Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Embodied Multi-Agent Task Planning from Ambiguous Instruction

Xinzhu Liu*†, Xinghang Li*†, Di Guo*, Sinan Tan*, Huaping Liu*‡ and Fuchun Sun*
* Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China
† Equal Contribution.
‡ Corresponding Author. E-mail: hpliu@tsinghua.edu.cn

***Abstract*—In human-robots collaboration scenarios, a human would give robots an instruction that is intuitive for the human himself to accomplish. However, the instruction given to robots is likely ambiguous for them to understand as some information is implicit in the instruction. Therefore, it is necessary for the robots to jointly reason the operation details and perform the embodied multi-agent task planning given the ambiguous instruction. This problem exhibits significant challenges in both language understanding and dynamic task planning with the perception information. In this work, an embodied multi-agent task planning framework is proposed to utilize external knowledge sources and dynamically perceived visual information to resolve the high-level instructions, and dynamically allocate the decomposed tasks to multiple agents. Furthermore, we utilize the semantic information to perform environment perception and generate sub-goals to achieve the navigation motion. This model effectively bridges the difference between the simulation environment and the physical environment, thus it can be simultaneously applied in both simulation and physical scenarios and avoid the notorious sim2real problem. Finally, we build a benchmark dataset to validate the embodied multi-agent task planning problem, which includes three types of high-level instructions in which some target objects are implicit in instructions. We perform the evaluation experiments on the simulation platform and in physical scenarios, demonstrating that the proposed model can achieve promising results for multi-agent collaborative tasks.**

I. Introduction

In real life, a group leader may release an ambiguous instruction, which contains his intention but lacks the implementation details. Nevertheless, intelligent group members may analyze the instruction to extract the intention and utilize their knowledge or shared-mental-mind with the leader to execute the necessary operational details to accomplish the task. Such a collaboration mechanism is also highly expected for human-robot collaboration. For example, a human would give robots an instruction in which the process of completing the task is obvious to the human himself. However, the overall instruction given to robots is likely ambiguous for them to understand as some information is implicit in the instruction, such as *"Put the book and newspaper away"*. Although human knows that the book and newspaper are most likely to be put on the bookshelf, or the drawer if there is no bookshelf found, robots may not know where to put the book and newspaper directly from the instruction, let alone collaborating to complete this task. Therefore, it is necessary for the robots to jointly reason the operation details and perform the embodied multi-agent task planning given the ambiguous instruction (Fig. 1).

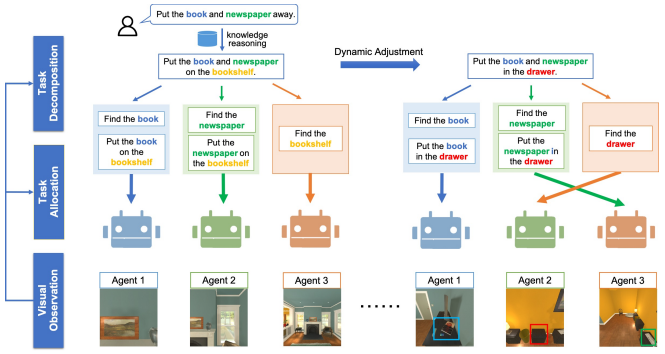

Fig. 1. An overview of the embodied multi-agent task planning from ambiguous instruction. Given a high-level instruction, several sub-tasks are generated and allocated to a group of agents. The agents explore the environment and implement the sub-tasks. With the change of the visual observation during the exploration process, the task decomposition and task allocation processes are also adjusted dynamically.

In the multi-agent task planning scenario, a complex task can be decomposed in multiple possible ways, and the decomposed sub-tasks are allocated to multiple agents for the execution [24]. Therefore, the task planning includes task decomposition, which focuses on the problem of *what to do* [21], task allocation which focuses on the problem of *who does what* [3], and task scheduling which focuses on the problem of *how to arrange tasks in time* [43]. Among them, the task decomposition is the problem of decomposing a complex task into simpler ones, down to the level of actionable tasks [21, 28]; task allocation is the problem of determining which robot should execute which task in order to achieve the overall system goal [3], and task scheduling is the problem of sequencing tasks for execution [17]. The above problems have been extensively investigated in diversified works of literature, most of which transform the given task into a well-defined optimization problem that requires a clear, structured, and complete instruction [2, 16]. In practical scenarios, task planning is highly coupled with the human-robot interaction and the perception of the environment. It should be dynamically adjusted due to the vagueness of the interaction and dynamics of the environment. In this work, we formulate such a problem as embodied multi-agent task planning from ambiguous instruction, which exhibits the following key challenges:

1) **Ambiguous Instruction** Due to the incompleteness and ambiguity of the given instruction, it is necessary to use

external knowledge sources (such as domain knowledge in a specific field, knowledge graph, and industry rules) to reason and clarify the instruction. Based on the clarified instruction, combined with the characteristics of the robot, the given task is required to be decomposed into specific sub-tasks that the robot can perform. For example, the task *"Put the book and the newspaper away"* should be clarified to be *"Put the book and the newspaper on the bookshelf"* with the visual perceptions of robots and the knowledge that books and newspapers are always placed on the bookshelf. Then it should be initially decomposed into the sub-tasks of *"Find the book"*, *"Find the newspaper"* and *"Find the bookshelf"*.

2) **Dynamic Task Decomposition** Since the initial visual perceptions of multiple robots are limited, the reasoning information based on the initial state may not be correct. Therefore, it is necessary to dynamically adjust the instruction reasoning and task decomposition with updated visual perceptions during the continuous execution process of agents. For example, the task *"Put the book and the newspaper away"* is clarified to be *"Put the book and the newspaper on the bookshelf"* with the initial visual perceptions. After several steps of exploration, agents find that there is no bookshelf but a drawer in the current scene based on their newly obtained visual perceptions. The book and newspaper can also be put in the drawer. Then agents need to re-reason the implicit information in the given instruction and obtain the new clarified instruction *"Put the book and the newspaper in the drawer"*. Afterward, the subsequent decomposition and allocation processes are performed based on the clarified results.

3) **Dynamic Task Allocation** Based on the specific decomposed sub-tasks, the sub-tasks need to be allocated to multiple robots considering the robots' perception and motion abilities so that each robot is assigned to a corresponding sub-task. More importantly, in the specific execution process, because of the ambiguity of instructions and the dynamic nature of the environment, robots are required to dynamically adjust their allocated sub-tasks according to the environment perception information. For example, the three decomposed tasks *"Find the book"*, *"Find the newspaper"* and *"Find the bookshelf"* are allocated to Agent 1, Agent 2 and Agent 3 respectively based on their initial visual environment information. After several steps, if Agent 1 finds that it is actually closer to the bookshelf based on its obtained observations, they need to re-allocate the sub-tasks and Agent 1 would change to perform *"Find the bookshelf"*, and Agent 2 would change to *"Find the book"* accordingly.

To tackle the above issues, we propose an embodied multi-agent task planning framework demonstrated in Fig.1 which utilizes external knowledge sources, and dynamically perceives environment information to parse the high-level ambiguous instructions, dynamically allocates the decomposed sub-tasks and completes the distributed navigation tasks. In this framework, multiple agents are able to leverage the advantages of their embodiment attribute to dynamically and automatically adjust the instruction parsing results and efficiently complete the task. The main contributions are summarized as follows:

1) **Multi-agent embodied task planning framework:** A multi-agent task planning framework is proposed to solve the multi-agent collaborative mission, which utilizes external knowledge sources, and dynamically perceived visual information to resolve the high-level instructions, and dynamically allocates the decomposed tasks.

2) **Sim&Real learning method for embodied task planning:** A dynamic task allocation model is developed based on multi-agent collaboration. We utilize the semantic information to perform the environment perception and generate sub-goals to achieve navigation motion. This model effectively bridges the difference between the simulation environment and the physical environment, thus it can be simultaneously applied in both simulation and physical scenarios and avoid the notorious sim2real problem.

3) **Evaluation and validation:** We build a benchmark dataset to validate the embodied multi-agent task planning problem, which includes three types of high-level instructions in which some target objects are implicit. We perform the evaluation experiments both in the AI2-THOR [22] platform and physical scenarios including *Easy* and *Hard* settings, which demonstrate that the proposed model can achieve promising results for multi-agent collaborative tasks.

II. Related Work

A. Embodied Multi-Agent Collaboration

Recently, embodied intelligence tasks which integrate perception and action (sometimes language) to perform navigation [41, 25], exploration [7], object search [39], question answering [12], remote embodied visual referring expression [29], task completion with language instructions [4] etc., are progressively proposed and accelerate the fusion of communities of machine learning, computer vision and robotics. Most of them are validated in simulation environment and some real-world experiments also emerges [1].

On the other hand, in multi-agent collaboration scenarios, the complicated collaborative mission, in which multiple agents are required to decompose a specific task and execute the sub-tasks in a distributed manner, has attracted a surge of attentions in domains of search and rescue [38, 18], exploration [10, 40], as well as industrial manufacturing [15]. Furthermore, multi-agent systems also witness remarkable progress in planning, perception, localization, and control [36].

Inspired by the success of the multi-agent system, there have been some researches on multi-agent embodied tasks in visual simulation environments. *FurnLift* [19] and *FurnMove* [20] tasks have been proposed to learn a multi-agent decision policy for two agents to collaboratively lift or move large objects at the same time. A centralized 3D reconstruction method is developed to solve the multi-agent question answering task, in which multiple agents explore the scene jointly to answer the given question [34]. Modified multi-agent reinforcement learning and memory-augmented communication module are

TABLE I
COMPARISON BETWEEN OUR WORK AND OTHER RELATED WORK WITH LANGUAGE INSTRUCTION

Work	Ambiguous Instructions	Task Decomposition	Dynamic Task Allocation	Embodied Multi-Agent	Vision Perception	Real-World Experiments
S2R-VLN [1]	✗	✗	✗	✗	✓	✓
REVERIE [29]	✗	✗	✗	✗	✓	✗
HLSM-ALFRED [4]	✗	✓	✗	✗	✓	✗
Virtualhome [28]	✓	✓	✗	✗	✗	✗
VGP-WV [21]	✓	✓	✗	✗	✗	✗
ProScript [30]	✓	✓	✗	✗	✗	✗
LMCR [8]	✓	✗	✗	✗	✓	✓
MA-EQA [34]	✗	✗	✗	✓	✓	✗
FurnMove [20]	✗	✗	✗	✓	✓	✗
CollaVN [37]	✗	✗	✗	✓	✓	✗
Ours	✓	✓	✓	✓	✓	✓

utilized to solve the multi-agent visual navigation task [37]. A centralized spatial coordination planner is developed to solve the multi-agent exploration tasks to improve the exploration efficiency [40]. In all multi-agent tasks mentioned above, the task each agent needs to complete is determined without the process of dynamic decomposition and allocation. In our work, the given task is required to be dynamically decomposed and assigned according to the visual perception of each agent. More importantly, our work provides a promising method to bridge the difference between the simulation and real-world.

B. Dynamic Task Decomposition and Task Allocation

In terms of task decomposition, the given task is required to decompose into executable sub-tasks [32, 27].

Particularly, in some tasks with instructions as input, the language feature is utilized to decompose the task. A commonsense instruction reasoning approach is proposed to help the agent to reason the missing information in the instruction with the environment observation and complete the instruction [8]. Pre-trained language models are utilized to decompose the abstract instruction into sub-tasks [21] or sub-programs [28] and predict their temporal orders [30] with language instructions.

In terms of task allocation, multiple tasks need to be allocated to multiple agents to obtain maximum benefit. Negotiation-based methods including time-constraint negotiation [23] and game theory-based negotiation [11] are classical methods to solve the task allocation problem. Auction-based algorithm [5], consensus-based bundle algorithm [42] as well as cross-entropy temporal logic optimization [3] have also been proposed to solve this problem. Besides, dynamic task allocation extends the general task allocation task to the dynamic environment setting, and the allocation results need to be modified with the change of the environment. The policy search algorithm of reinforcement learning is used to study the autonomous task sequencing challenge, in which the agent swarm can sequence tasks whose execution orders are unknown [17]. Stochastic conflict-based allocation model [9] and adaptive allocation approach for heterogeneous agents [13, 14] are proposed to solve the challenge of dynamic multi-agent task assignment under environment uncertainty and temporal constraints. All the above methods regard task allocation as an optimization problem that do not consider the impact of the visual perception of different agents. Our work is different from other relevant works that have language instructions as input, and the comparison between these works is illustrated in Table I.

III. PROBLEM FORMULATION

In this work, we focus on multiple embodied agents which aim to decompose the task indicated by the high-level ambiguous instruction into several sub-tasks, collaboratively allocate sub-tasks and cooperate to complete the entire task with egocentric visual observations.

Concretely speaking, we consider embodied agents $\{D^{(1)}, D^{(2)}, \cdots, D^{(N)}\}$, where N is the number of agents. All of the agents share an action space $\mathcal{A}$, from which each agent can take an action to perform at each time instant. The state representation of the i-th agent is represented $S_t^{(i)}$.

Given a high-level language instruction $\mathcal{L}$, which contains the main intention of the person but lacks necessary operation details to ensure the task to be performed, what we hope to address is to analyze the intention embedding in the instruction, complete the operation details using external knowledge source, and perform the task planning for practice execution. So, we should first transform the original instruction $\mathcal{L}$ as structural representation $\mathcal{P}$, which may contain incomplete or missing items due to the ambiguity of the instruction $\mathcal{L}$.

Therefore, at each time instant t, we have to resort to the available external knowledge source $\mathcal{K}$ and the agents' perception information to complete the necessary operation details and decompose the whole task into sub-tasks $\mathcal{T}_t = \{T_t^{(1)}, T_t^{(2)}, \cdots, T_t^{(K)}\}$ with temporal ordering constrains, where $T_t^{(k)}$ denotes a specific sub-task, and K is the number of sub-tasks. This procedure can be represented as

$$\mathcal{T}_t = \mathrm{TD}(\mathcal{P}, \mathcal{S}_t)$$

where $\mathcal{S}_t = \{S_t^{(1)}, S_t^{(2)}, \cdots, S_t^{(N)}\}$ are the collected state information from the agents. Obviously, the above representations formulate a perception-aware task decomposition problem, and it results in a list of sub-tasks that should be performed. The next step is to allocate them to the embodied agents. The task allocation results can be represented as $\mathcal{V}_t = \{V_t^{(1)}, V_t^{(2)}, \cdots, V_t^{(N)}\}$, which can be derived as

$$\mathcal{V}_t = \mathrm{TA}(\mathcal{T}_t, \mathcal{S}_t).$$

Once the goal of each agent at time t is determined, a multi-agent search strategy can be used to perform as

$$a_t^{(i)} = \pi^{(i)}(S_t^{(i)}, S_t^{(i^-)}, V_t^{(i)}) \tag{1}$$

for $i = 1, 2, \cdots, N$, where $S_t^{(i^-)}$ is the state information the other agents which can be transferred via communication.

According to Eq.(1), the main concern in this work is essentially a multi-agent visual semantic search problem. Different from existing VSN work such as [37, 26], in our work, the goal of each agent $V_t^{(i)}$ is time-varying, and it is determined from an ambiguous instruction, which makes this problem more challenging.

IV. Dataset Construction

To evaluate the embodied task planning in the general environment is extremely difficult, and a specific scenario benchmark simulation dataset is desired to validate the proposed methods. In this work, we restrict ourselves to deal with the ambiguous instructions from which we may infer the receptacles for some objects. We use such practical scenarios to construct a benchmark dataset.

There exist lots of simulation environments such as Matterport3D [6] and Habitat [31] simulator. Though existing datasets in these simulators could provide photo-realistic visual perception for the navigation, none of them supports manipulating actions that are required to evaluate the embodied multi-agent task planning from ambiguous instruction. To develop a benchmark, we resort to ALFRED [33] dataset in AI2-THOR simulator[22]. ALFRED is a recently developed benchmark for interpreting grounded instructions for everyday tasks. Based on ALFRED dataset, reasonable and abstract high-level instructions can be generated in which the target object is implicit and need to be inferred. However, ALFRED does not provide ambiguous instructions, and it cannot be directly used for multi-agent exploration.

In this work, we use the instructions in ALFRED to extract necessary information and construct a new benchmark with AI2-THOR, which supports multi-agent collaboration. Concretely speaking, we extract five fundamental types of behaviors *heat*, *cool*, *clean*, *put*, and *throw* in the high-level instructions. In addition, we extract the relationships between object properties and behaviors in ALFRED, such as *bread can be heated* and *book cannot be heated or cleaned*. We also preserve the placement constraints in ALFRED between small objects and receptacles, which imposes the constraints on whether a certain type of objects could be placed on or inside another type of receptacle objects. For example, the reasonable receptacles of *book* can be *bookshelf*, *desk* and *countertop* rather than *garbage can*. This provides solutions to design reasonable ambiguous instructions.

Then, we use the AI2-THOR environment to design our tasks. The constructed dataset consists of three types of multi-agent tasks indicated by high-level instructions, in which the target objects and operation information may be implicit and need to be reasoned during the process of task execution. As illustrated in Fig.2, the three types of tasks are generated based on the fundamental actions, object properties, and placement constraints. When generating high-level instructions for each type of tasks, we select each type of action, preserve objects meeting object properties constraints for the specific action, and obtain the most likely reasonable receptacles for the specific pairs of action and object to form the triple $(action, object, receptacle)$. In particular, we sort the reasonable receptacles in a descending order according to the frequency of the triples with the specific action and object pair appeared in ALFRED instructions. For the fixed action and object, the candidate triples are utilized to generate the high-level instructions.

For the task Type I, the high-level instructions contain one action and one object. Instructions are generated directly based on obtained triplets as well as several language rules, and implicit receptacles do not appear explicitly in instructions. For instance, in the instruction *"Heat the bread"*, the receptacle *microwave* needs to be reasoned.

For the task Type II, the high-level instructions include one action and two objects that can be executed by that action. We select two objects with a common receptacle for a specific action from candidate triples and generate high-level instructions in a similar way, such as *"Heat the tomato and the bread"*.

For the task Type III, the high-level instructions contain two actions and two objects, in which two actions have a temporal order, and the object processed by the first action is also the target object of the second action *put*. After the first action is completed, the operated object needs to be placed on another receptacle, so we filter to get receptacles of the second action according to the placement constraints. The instructions are generated using "and" to connect two action instructions and the receptacle of the first action is implicit. For instance, in the instruction *"Heat the bread and put it on the countertop"*, the *bread* needs to be put on the *counterTop* after heated in the *microwave*.

It is noted that the receptacles are implicit in all three types of tasks, and they need to be inferred when the task is processed. Meanwhile, all three types of tasks can be decomposed into sub-tasks that are executed by multiple agents.

To facilitate evaluating the dynamic task allocation strategy, we utilize the object mask to perform data argumentation. For the sorted receptacles, we mask out the top three objects in the scene respectively and select another object of the highest frequency to form new triples and generate new tasks of the three types. The correct inferred receptacles are different from

TABLE II
DATASET SPLITS

	Train	Validation(Unseen)	Test(Unseen)
#Tasks	4025	728	926
#Scenes	80	20	20
#Demonstrations	75260	9465	22310

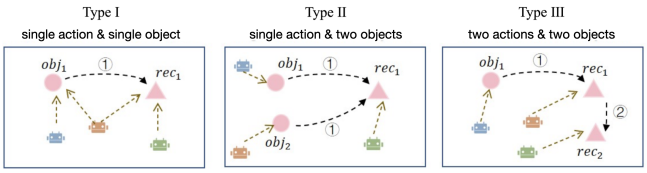

Fig. 2. Three types of instructions in our dataset.

the task without masked objects, but the generated high-level instructions are the same since the receptacles are implicit in instructions. When the agent cannot see the top 1 masked receptacle during the task process, it needs to dynamically adjust the inferred results to find other possible receptacles. For example, in the instruction *"Put the book away"*, the most likely receptacle *Drawer* is masked, the agent needs to re-reason the receptacle to be *Cabinet* after finding no *Drawer* in the scene. Meanwhile, it is noted that to make the generated dataset more natural and flexible, we extend the original dataset to a larger one, in which we utilize natural language processing methods to generate richer daily activity instructions through synonym generation, syntax analysis, and sentence pattern conversion.

The expert demonstrations are also generated, which consist of the task decomposition results, sub-task allocation results, and actions to be taken at each step for every agent. The task decomposition results are generated with specific language rules and the ground-truth task triples. We consider the situation where three agents are in the environment, and the locations of the three agents are initialized under the constrains that the distance between multiple agents is greater than 1m and the distance between agents and the same target object is also greater than 1m, ensuring allocation results reasonable. The allocation results are generated with the task decomposition results and agents' initialization locations. The navigation actions are generated by the shortest path algorithm in the sub-goal level with the rotation.

From the above description, we can see that though the developed benchmark is dependent on AI2-THOR, it is significantly different with ALFRED. Finally, We divide the full dataset, including 120 scenes, into three non-overlapping splits, namely Train, Validation, and Test. Train split contains 80 scenes, while Validation and Test split contain 20 unseen scenes each. The corresponding information is shown in Table II. In addition, more details of the dataset is presented in the Appendix A and we will release this dataset soon.

V. MODELS

The *Embodied Multi-Agent Task planning* requires the agents to reason a set of sub-tasks from the high-level instruction and to dynamically correct and re-allocate the sub-tasks in the exploring process. Considering these challenges, we propose a hierarchical multi-agent framework to solve the task. Our framework consists of four essential modules: scene encoder module, task planning module which concludes the task decomposition and task allocation, action module and communication module, as is shown in Fig. 3.

A. Scene Encoder

The scene encoder module firstly takes the RGB and depth observations as input to obtain the semantic map, and a pre-trained method [35] is utilized to generate the semantic map feature vector. The *Swin Transformer* is used to obtain the semantic point cloud from the agent's current RGB and depth observations. Then we voxelize the semantic point cloud with a voxel size of $0.125m \times 0.125m$ and a semantic map of $C \times M \times M$ is obtained, where C indicates the number of object types, M denotes the length of the semantic map. We maintain the egocentric semantic map that is within four meters around the agent, so the size of i-th agent's semantic map $mp_t^{(i)}$ is $C \times 64 \times 64$ at step t. Specifically, for step t, the $i-th$ agent's semantic map $mp_t^{(i)}$ is generated by merging its current local semantic map $lp_t^{(i)}$ and the semantic map $mp_{t-1}^{(i)}$ from previous step.

Then, a scene encoder $ScEr$ is trained with a pre-training method to generate the feature vector $sm_t^{(i)}$ from the semantic map $mp_t^{(i)}$. The feature vector is supposed to retain the region information in the semantic map. The scene encoder is composed of a series of convolution layers. To train the scene encoder, we give a query to see if a specific type of object exists in a sub-region of the map. The scene encoder is trained to answer this query correctly to get the reasonable semantic map feature vector. With the pre-trained scene encoder $ScEr$, we could have the semantic map feature vectors as $sm_t^{(i)} = ScEr(Merge(mp_{t-1}^{(i)}, lp_t^{(i)}))$, which are the input for the task planning and action modules.

B. Task Planning

The task planning module is composed of task decomposition part and task allocation part. The task decomposition part takes the high-level instruction and the semantic map feature vectors as input for receptacle reasoning and sub-task sequences generation. The task allocation part allocates the sub-task sequence to each agent conditioned on its semantic map information.

1) Task decomposition: Given a high-level instruction, the task decomposition part firstly extracts the key actions and objects in the instruction, and the implicit receptacles in the instruction are predicted based on the semantic map feature vectors. At step t, we concatenate the encoding of the action and objects from the instruction to generate the task embedding k_t. And then the task embedding k_t and the current

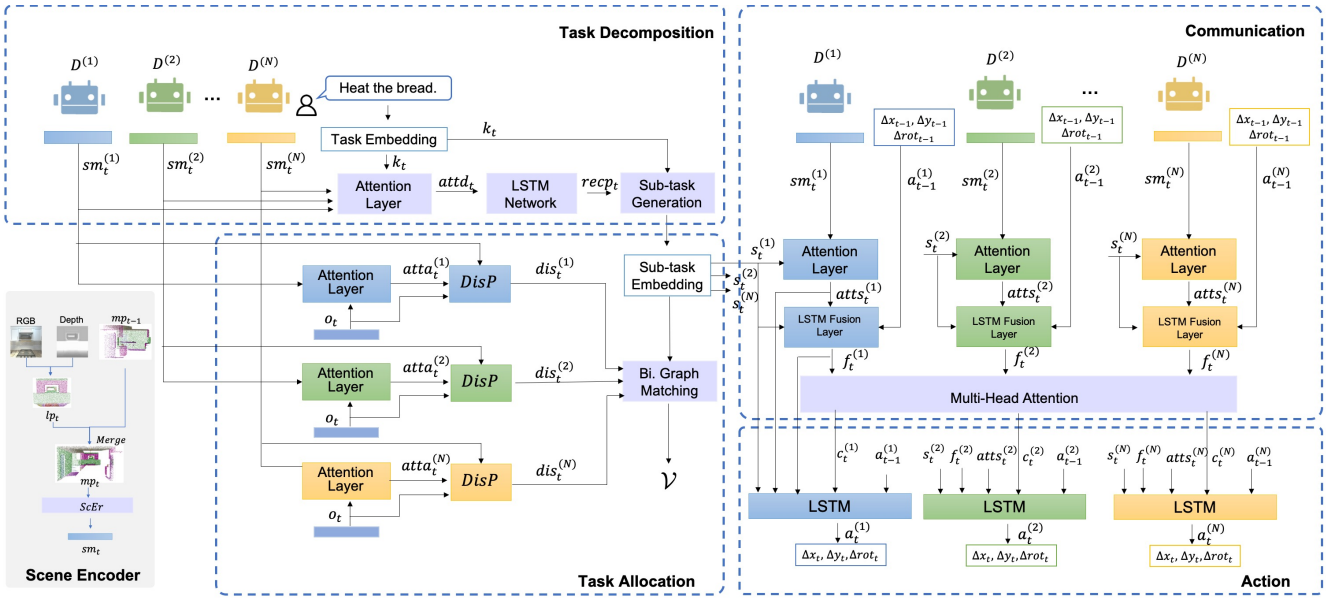

Fig. 3. An overview of the hierarchical multi-agent framework for the embodied multi-agent task planning.

semantic map feature vectors $sm_t^{(i)}, i = 1, 2, \cdots, N$ from all agents are fed into an attention layer and the attention $attd_t$ is obtained. We use a LSTM network to predict the implicit receptacle as $recp_t = F_d(LSTM(k_t, attd_t))$, where F_d denotes the receptacle prediction network. Having the receptacle $recp_t$, the given incomplete instruction can be completed. Then, the decomposed sub-tasks as well as their corresponding temporal order are generated based on the type of the instruction.

2) Task allocation: The task allocation part predicts the distance from the agents' current location to target objects in the scene given its egocentric semantic map feature vector $sm_t^{(i)}$. At step t, given the embedding of the target objects o_t, we firstly calculate the attention $atta_t^{(i)}$ of o_t on semantic map vector $sm_t^{(i)}$. Then the distance $dis_t^{(i)}$ from the agent's current location to target objects is predicted as follows,

$$\begin{aligned} dis_t^{(i)} &= DisP(sm_t^{(i)}, LSTM(atta_t^{(i)}), o_t) \\ dis_t^{(i)} &= \{dis_t^{(i)}(1), dis_t^{(i)}(2), \cdots, dis_t^{(i)}(H)\} \end{aligned} \tag{2}$$

where H is the number of target objects, $DisP$ denotes the distance predictor which consists of linear layers. After the distances are predicted, the task allocation part utilizes the method similar to the bipartite graph matching algorithm to allocate each sub-task to each agent based on their predicted distances so that the total length executed when multiple agents complete the task is the smallest. And the result of task allocation is denoted as $\mathcal{V} = F_a(dis_t^{(1)}, dis_t^{(2)}, \cdots, dis_t^{(N)})$, where F_a denotes the task allocation model. The task allocation method requires that each sub-task is assigned to at least one agent for execution.

It is noted that the task decomposition and task allocation results can be adjusted dynamically. After each step t, new visual observations can be obtained after the agents execute actions based on their assigned sub-tasks. Then with their updated semantic map feature vectors, the task planning module will also be updated correspondingly.

C. Communication

Having the sub-tasks allocated, the communication module exchanges the information among agents. The agent can utilize the message from other agents to facilitate itself for the following interaction module.

For each agent $D^{(i)}, i = 1, 2, ..., N$, we utilize the triple $(action, object, receptacle)$ to represent the corresponding sub-task for each agent, and *"blank"* is used to denote a vacant item in the sub-task, such as $(find, bread, blank)$ for the sub-task "Find the bread". Then we encode the triple to get the sub-task embedding $s_t^{(i)}$ assigned to agent D^i at step t. The sub-task $s_t^{(i)}$ and the agent's semantic map feature vector $sm_t^{(i)}$ are firstly fed into an attention layer and the attention $atts_t^{(i)}$ is obtained. The action information of the agent is represented by the sub-goal. And the sub-goal of the agent in the previous step $t-1$ includes the movement distance in its egocentric x and y axis $\Delta x_{t-1}^{(i)}$ $\Delta y_{t-1}^{(i)}$, and the rotation angle $\Delta rot_{t-1}^{(i)}$. We encode the previous sub-goal to get the action embedding $a_{t-1}^{(i)}$. And then the attention $atts_t^{(i)}$, the corresponding sub-task $s_t^{(i)}$, and the sub-goal of the agent in the previous step $a_{t-i}^{(i)}$ are fed into a LSTM fusion layer and the fused feature $f_t^{(i)}$ is obtained.

If we view the multi-agents as a sequence of agents, then the communication between each pair of agents is similar to self-attention mechanism, which is widely used in transformer-based methods. Therefore, we use the multi-head attention

layer for the information communication, which takes the fused feature $f_t^{(i)}$ from multiple agents as input, and calculates the attention of each agent's state information on other agents' states separately to generate the communication embedding as

$$(c_t^{(1)}, c_t^{(2)} \cdots, c_t^{(N)}) = F_{mha}(f_t^{(i)}, f_t^{(2)} \cdots, f_t^{(N)})$$

where F_{mha} denotes the multi-head attention network. Through attention mechanism, each agent can selectively utilize the beneficial information from others for the following action module. Meanwhile, we modify the traditional multi-head attention structure with the Softmax activation function to make the communication module more convenient to extend to the multi-agent system with a larger number of agents.

D. Action Module

The action module leverages the communication information, semantic information, sub-task information and previous sub-goal to predict the next sub-goal for the agent. After obtaining the sub-goal, the low-level action generation policy is employed to generate the navigation actions for the agent to execute in the environment.

We train a sub-goal predictor F_p to generate the next sub-goal $a_t^{(i)}$, which includes the egocentric movement distance in the x and y axis, namely $\Delta x_t^{(i)}$ and $\Delta y_t^{(i)}$, the rotation angle $\Delta rot_t^{(i)}$, and the probability of "Stop" action $stop_t^{(i)}$. Given $f_t^{(i)}$, $s_t^{(i)}$, $a_{t-1}^{(i)}$, $atts_t^{(i)}$ and c_t^i, the next sub-goal $a_t^{(i)} = F_p(LSTM(f_t^{(i)}, s_t^{(i)}, a_{t-1}^{(i)}, atts_t^{(i)}, c_t^{(i)})$ is generated. If the agent performs "stop" action, the model judges whether the target object is found. If it is found, a specific manipulation action such as "Pick" is performed and the current sub-task is switched to the next one. If the current sub-task is the last one, then the entire task is completed.

To navigate to the sub-goal, in the simulation environments, a low-level action strategy based on the shortest path algorithm is utilized to make agents navigate from the current sub-goal to the next sub-goal. In real-world experiments, agents can navigate from the current location to the predicted sub-goal based on their own localization, obstacle avoidance, and path planning system. In this way, we can bridge the difference between the simulation and real-word to the most.

VI. Performance Validation

A. Experiment Settings

To comprehensively evaluate the proposed method, we design two settings *Easy* and *Hard* using the developed dataset. The main difference lies in the layout of the room.

1) Easy: All of the target objects and agents are in one single room. In this setting, multiple agents are initialized in random positions.

2) Hard: The target objects and multiple agents are distributed in different rooms, and some agents need to go from one room to another to complete the specified task. This is an extension to AI2-THOR environment. In this setting, multiple target objects are distributed in different rooms and multiple agents are initialized in random rooms.

We train the models only in *Easy* setting and evaluate the performance in both *Easy* and *Hard* settings to assess the generalization ability of models.

B. Evaluation Metrics

We utilize two metrics to compare the performance of all models: Success Rate (SR), Success weight by Path Length (SPL). If all sub-tasks are completed successfully under the temporal order constrains, the given task is considered successful. SR is the ratio of successful tasks, which is defined as $SR = \frac{1}{N_{task}} \sum_{i=1}^{N_{task}} R_i$ where $R_i = 1$ if the i-th task is successful, otherwise $R_i = 0$, and N_{task} is the number of tasks. SPL is denoted as $SPL = \frac{1}{N_{task}} \sum_{i=1}^{N_{task}} R_i \frac{L_i}{max(D_i, L_i)}$, where L_i denotes the minimum number of steps for multi-agent to complete the task, and D_i is the actual steps.

C. Comparison of Task Planning

We conduct experiments to evaluate the effectiveness of our task planning module, including task decomposition and task allocation parts.

In the task decomposition part, we are interested in the difference between Fixed Task Decomposition (FTD) and Dynamic Task Decomposition (DTD). The former provides task decomposition results directly from the external knowledge base, while DTD will further dynamically adjust the results using the perception information. In addition, to show the performance gap to the upper bound, we also introduce the Oracle Task Decomposition (Oracle-TD) method, which uses the ground truth of the task decomposition.

In the task allocation part, we first design two baseline methods: For *Random* method, the agents randomly choose the task allocation for execution. For *Init.* method, the agents initially use some simple rules which satisfy the allocation constrictions to form the allocation results and keep them unchanged during the running period. We consider two cases of our proposed task allocation method: *Fixed Task Allocation (FTA)* and *Dynamic Task Allocation (DTA)*: The former calculates the allocation results for only once and the latter dynamically adjust the allocation using the perception information. Finally, we replace the calculation method in *Fixed Task Allocation (FTA)* and *Dynamic Task Allocation (DTA)* with the ground truth results to form the ceiling methods *Oracle-FTA* and *Oracle-DTA*.

Therefore, there are a total of $3 \times 6 = 18$ combinations considering different task decomposition and allocation ways, and we evaluate them in two types of difficulty setting with three types of tasks, respectively. The results are shown in Table III, from which we draw the following conclusions:

1) *Dynamic task decomposition significantly performs better than fixed task decomposition:* The reason is that FTD simply reasons receptacles based on natural language without considering visual perceptions, while DTD dynamically updates the decomposition results according to agents' perception information. An exception appears in Type III, in which the receptacles required for inference in the type *two action & two objects* are relatively simple, and can be accurately reasoned

TABLE III
COMPARATIVE RESULTS IN TASK PLANNING MODULE

Settings	Task Allocation	Task Decomposition																	
		Type I						Type II						Type III					
		Oracle-TD		FTD		DTD		Oracle-TD		FTD		DTD		Oracle-TD		FTD		DTD	
		SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)
Easy	Random	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Init	60.3	12.0	14.2	3.1	35.6	7.1	13.7	3.2	2.4	0.8	8.7	2.2	22.0	5.7	22.0	5.7	22.0	5.7
	FTA	77.4	15.9	17.3	4.4	44.9	9.7	26.4	6.0	3.8	0.9	14.7	3.6	22.0	6.1	22.0	6.1	22.0	6.1
	DTA	**78.2**	**16.7**	18.9	**5.1**	**45.1**	**10.3**	**29.6**	**7.3**	**4.6**	**1.1**	**17.3**	**4.4**	**27.0**	**6.8**	**27.0**	**6.8**	**27.0**	**6.8**
	Oracle-FTA	78.2	16.2	17.7	4.4	46.5	9.7	31.3	7.5	5.3	1.4	17.3	4.5	31.0	7.2	31.0	7.2	31.0	7.2
	Oracle-DTA	81.9	18.2	20.0	5.5	49.0	11.3	35.6	9.6	5.5	1.4	18.8	5.3	35.0	8.9	35.0	8.9	35.0	8.9
Hard	Random	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Init	31.6	19.7	11.5	6.5	13.9	7.6	6.7	2.3	2.6	0.2	4.0	1.3	3.0	0.8	3.0	0.8	3.0	0.8
	FTA	46.9	28.2	18.1	11.8	18.6	12.0	10.7	3.7	2.7	0.9	8.0	3.2	4.5	0.8	4.5	0.8	4.5	0.8
	DTA	**50.2**	**28.9**	**19.6**	**12.5**	**22.0**	**13.2**	**13.3**	**5.1**	**5.3**	**2.3**	**9.3**	**3.9**	**13.4**	**2.9**	**13.4**	**2.9**	**13.4**	**2.9**
	Oracle-FTA	55.0	30.7	20.6	13.9	21.5	13.5	20.0	6.7	9.3	3.2	14.7	5.3	11.9	3.1	11.9	3.1	11.9	3.1
	Oracle-DTA	55.5	30.5	21.5	13.4	23.0	13.7	21.3	7.0	12.0	4.0	17.3	5.8	14.9	3.8	14.9	3.8	14.9	3.8

directly based on fixed decomposition. Therefore, the success rates of different decomposition models in this type of tasks are the same. We also observe that the oracle decomposition method Oracle-TD performs best in each type of tasks because it uses the true decomposition results for the subsequent process. This reflects that the task decomposition significantly affect the subsequent task allocation and navigation process, so it has great influence on the success rate of the whole tasks. The results also show that there still exists gap between Oracle-TD and our DTD, which means the embodied task decomposition is challenging and still has large space for further research.

2) *Dynamic task allocation significantly performs better than fixed task allocation:* It is found that the dynamic allocation methods always have advantages over the specific fixed ones. Due to the mutual influence of task allocation and navigation, agents perform navigation based on the allocated subtasks, and agents would obtain new distance prediction results with the movement and new visual observations. Dynamic allocation methods can dynamically update the allocation results based on new perceptions at each step, which is more reasonable than the fixed strategy.

3) *The embodied task planning achieves the best results except Oracle:* Agents have less visual perceptions about the scene at the beginning of the task, then the receptacle reasoning and distance prediction become more accurate with the continuous interaction and perception with the scene, which can generate more accurate allocation results and improve the accuracy of navigation and task completion. The results demonstrate the importance and effectiveness of the embodied task planning module.

D. Qualitative Results

We demonstrate the successful sample of multi-agent task planning in hard settings, which is shown in Fig.4. Agents perform *"Clean the plate and put it on the desk"* in different rooms. At 3-th step, Agent 1 and Agent 3 dynamically adjust the task allocation results with the updated perceptions of the scene. After Agent 2 finishing its sub-tasks *Clean the plate in the sink*, it switches to *Put the plate on the desk*, goes into the bedroom with the help of the effective information from Agent 3, and finally completes the task successfully. This successful sample demonstrates that the embodied dynamic task allocation is important and effective in the multi-agent task planning process. Some more success and failure examples are illustrated in Appendix C (in simulator) and D (in real scenario).

TABLE IV
COMPARATIVE RESULTS FOR MULTI-AGENT V.S. SINGLE-AGENT

Settings	Methods	Type I		Type II		Type III	
		SR(%)	SPL(%)	SR(%)	SPL(%)	SR(%)	SPL(%)
Easy	VSN	13.2	2.9	2.0	0.2	4.0	1.3
	VSN-SP	15.0	3.0	4.0	0.5	9.5	2.5
	Single-Agent	46.3	10.4	10.1	2.3	22.0	8.7
	Multi-Agent w/o Com.	77.8	16.0	29.8	7.0	26.5	6.0
	Multi-Agent	**78.2**	**16.2**	**31.3**	**7.5**	**31.0**	**7.2**
Hard	VSN	8.1	3.8	0.0	0.0	0.0	0.0
	VSN-SP	11.0	5.7	0.0	0.0	1.5	0.2
	Single-Agent	23.9	10.6	1.3	0.8	4.5	1.6
	Multi-Agent w/o Com.	52.7	26.4	18.7	**8.0**	7.5	1.9
	Multi-Agent	**55.0**	**30.7**	**20.0**	6.7	**11.9**	**3.1**

E. Multi-Agent v.s. Single-Agent

Before closing this section, we present results to show the advantages of utilizing multi-agent in these scenarios, and therefore further verify the roles of the embodied task planning since it is only useful for multi-agent. To this end, we compare the results of our multi-agent task planning framework in three types of tasks with the following methods.

For single-agent cases, we consider three methods. The most typical one is *VSN-SP* in which scene prior is incorporated into this model except for the RGB observations and the previous action as input[39]. This method reduces to *VSN* if we remove the scene prior module. Further, we set the number of the agent

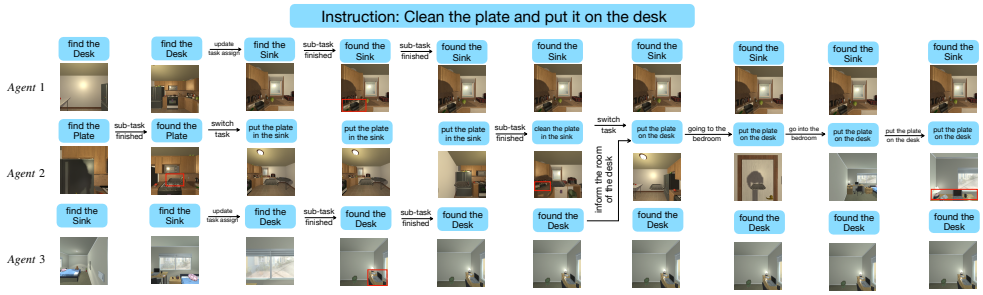

Fig. 4. Qualitative results of the *Hard* setting in the Simulation environment.

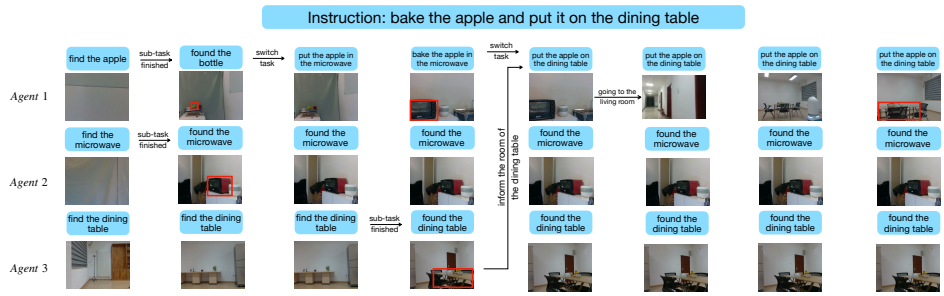

Fig. 6. Qualitative results of the *Hard* setting in the Real-World scenarios.

Fig. 5. The scenario setup in the Real-World experiments.

to one in our method and form the method *Single-Agent* which does not have the task planning and communication module. The core difference between *Single-Agent* and *VSN-SP*, *VSN* is that the former uses the sub-goal planning while the latter ones use the low action generation policy.

For multi-agent cases, we consider two methods. The first one is named *Multi-Agent w/o com*, which does not have the communication module for the agents to exchange information during the collaborative navigation, and its rest is the same as our multi-agent framework. The other one *Multi-Agent* is our standard and complete version.

To perform a fair comparison between multi-agent and single-agent baselines, we use the oracle task decomposition and allocation results, and the SPL is calculated based on the low-level actions. The comparison results are shown in Table IV, from which we make the following observations:

1) Even for single-agent case, the proposed method *Single-Agent* performs better than *VSN* and *VSN-SP*. This illustrates the roles of the semantic map coding for the visual perception and sub-goal planning for the action generation.

2) The *Multi-Agent* method performs much better than *Single-Agent* methods. This shows the advantages of the multi-agent in such a scenario and the roles of the task planning.

3) Type III has higher requirements for collaboration, and our model has significant improvement compared with the model without communication. The comparison results between *Multi-Agent* and *Multi-Agent w/o Com.* demonstrate the effectiveness of the communication module.

VII. Real-World Experiments

As we claim, the proposed learning method reduces the gap between simulation and real-world by using the semantic encoder to represent the visual perception and the sub-goal to facilitates navigation. To verify this point, we establish a real-world scenario in a practical office building, which is also an unseen environment but the navigation map is available. This scenario consists of Room1(30m^2), Room2(100m^2) and the corridor(35m long) connecting them. Following similar guidelines, we design the *Easy*(single room) and *Hard* settings(cross room), which are shown in Fig.5. We deploy three customized mobile robots as the embodied agents. Since we use the sub-goal navigation policy, the developed method can be directly transferred to the real robots. However, it is noted that our robotic platform does not have manipulator and therefore we assume that the object is virtually manipulated when it is found by the object detector, which is consistent with the setting in [29].

In Fig.6 we show the results for *Hard* setting. We also show the qualitative result of the *Easy* setting in Appendix D. The results of both tasks in detail are shown in the attached video. From the evaluation experiments in the real-world scenes, it is found that the encoded semantic map and the sub-goal prediction can help shrink the gap in visual and action domains between simulation and real-world.

VIII. Conclusions

In this paper, we establish the benchmark and develop methods for the challenging problem of embodied multi-agent task planing from ambiguous instructions. An important merit of this work is that the multiple agents could jointly reason the implicit meaning in the ambiguous instruction and perform the autonomous task planning for navigation. The proposed method has been validated in the simulation and real-world environments and the experimental results verify that the dynamic adjustment for the task decomposition and task allocation is vital for embodied agents.

Due to the limitation of the simulator, we can only investigate this problem in restricted scenarios. In the future work, we would like to extend this work to more complicated real scenarios and further improve the multi-agent collaboration capability of dealing with the ambiguous instruction.

Acknowledgments

This work was supported in part by the National Natural Science Fund for Distinguished Young Scholars under Grant 62025304.

References

[1] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021.

[2] Barbara Arbanas, Antun Ivanovic, Marko Car, Matko Orsag, Tamara Petrovic, and Stjepan Bogdan. Decentralized planning and control for uav–ugv cooperative teams. *Autonomous Robots*, 42(8):1601–1618, 2018.

[3] Christopher Banks, Sean Wilson, Samuel Coogan, and Magnus Egerstedt. Multi-agent task allocation using cross-entropy temporal logic optimization. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7712–7718. IEEE, 2020.

[4] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. *arXiv preprint arXiv:2107.05612*, 2021.

[5] Martin Braquet and Efstathios Bakolas. Greedy decentralized auction-based task allocation for multi-agent systems. *IFAC-PapersOnLine*, 54(20):675–680, 2021.

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020.

[8] Haonan Chen, Hao Tan, Alan Kuntz, Mohit Bansal, and Ron Alterovitz. Enabling robots to understand incomplete natural language instructions using commonsense reasoning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1963–1969. IEEE, 2020.

[9] Shushman Choudhury, Jayesh K Gupta, Mykel J Kochenderfer, Dorsa Sadigh, and Jeannette Bohg. Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Autonomous Robots*, pages 1–17, 2021.

[10] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 43(2):485–501, 2019.

[11] Rongxin Cui, Ji Guo, and Bo Gao. Game theory-based negotiation for multiple robots task allocation. *Robotica*, 31(6):923–934, 2013.

[12] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2018.

[13] Yousef Emam, Siddharth Mayya, Gennaro Notomista, Addison Bohannon, and Magnus Egerstedt. Adaptive task allocation for heterogeneous multi-robot teams with evolving and unknown robot capabilities. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7719–7725. IEEE, 2020.

[14] Yousef Emam, Gennaro Notomista, Paul Glotfelter, and Magnus Egerstedt. Data-driven adaptive task allocation for heterogeneous multi-robot teams using robust control barrier functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9124–9130. IEEE, 2021.

[15] Catriona Eschke, Mary Katherine Heinrich, Mostafa Wahby, and Heiko Haman. Self-organized adaptive paths in multi-robot manufacturing: reconfigurable and pattern-independent fibre deployment. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4086–4091. IEEE, 2019.

[16] Barbara Arbanas Ferreira, Tamara Petrović, and Stjepan Bogdan. Distributed mission planning of complex tasks for heterogeneous multi-robot teams. *arXiv preprint arXiv:2109.10106*, 2021.

[17] Lorenzo Garattoni and Mauro Birattari. Autonomous task sequencing in a robot swarm. *Science Robotics*, 3(20), 2018.

[18] Chao Huang and Rui Liu. Robot inner attention modeling for task-adaptive teaming of heterogeneous multi robots. *arXiv preprint arXiv:2006.15482*, 2020.

[19] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6689–6699, 2019.

[20] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander

Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *European Conference on Computer Vision*, pages 471–490. Springer, 2020.

[21] Peter A Jansen. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. *arXiv preprint arXiv:2009.14259*, 2020.

[22] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[23] Yan Kong, Minjie Zhang, and Dayong Ye. A negotiation-based method for task allocation with time constraints in open grid environments. *Concurrency and Computation: Practice and Experience*, 27(3):735–761, 2015.

[24] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[25] Jeffrey L Krichmar, Tiffany Hwu, Xinyun Zou, and Todd Hylton. Advantage of prediction and mental imagery for goal-directed behaviour in agents and robots. *Cognitive Computation and Systems*, 1(1):12–19, 2019.

[26] Xinzhu Liu, Di Guo, Huaping Liu, and Fuchun Sun. Multi-agent embodied visual semantic navigation with scene prior knowledge. *IEEE Robotics and Automation Letters*, pages 1–1, 2022. doi: 10.1109/LRA.2022.3145964.

[27] James Motes, Read Sandström, Hannah Lee, Shawna Thomas, and Nancy M Amato. Multi-robot task and motion planning with subtask dependencies. *IEEE Robotics and Automation Letters*, 5(2):3338–3345, 2020.

[28] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[29] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020.

[30] Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. proscript: Partially ordered scripts generation via pre-trained language models. *arXiv preprint arXiv:2104.08251*, 2021.

[31] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019.

[32] Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663. PMLR, 2018.

[33] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[34] Sinan Tan, Weilai Xiang, Huaping Liu, Di Guo, and Fuchun Sun. Multi-agent embodied question answering in interactive environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 663–678. Springer, 2020.

[35] Sinan Tan, Mengmeng Ge, Di Guo, Huaping Liu, and Fuchun Sun. Self-supervised 3d semantic representation learning for vision-and-language navigation. *arXiv preprint arXiv:2201.10788*, 2022.

[36] Hoa Van Nguyen, Hamid Rezatofighi, Ba-Ngu Vo, and Damith C Ranasinghe. Multi-objective multi-agent planning for jointly discovering and tracking mobile objects. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7227–7235, 2020.

[37] Haiyang Wang, Wenguan Wang, Xizhou Zhu, Jifeng Dai, and Liwei Wang. Collaborative visual navigation. *arXiv preprint arXiv:2107.01151*, 2021.

[38] Hongling Wang, Chengjin Zhang, Yong Song, and Bao Pang. Master-followed multiple robots cooperation slam adapted to search and rescue environment. *International Journal of Control, Automation and Systems*, 16(6):2593–2608, 2018.

[39] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[40] Chao Yu, Xinyi Yang, Jiaxuan Gao, Huazhong Yang, Yu Wang, and Yi Wu. Learning efficient multi-agent cooperative visual exploration. *arXiv preprint arXiv:2110.05734*, 2021.

[41] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.

[42] Farouq Zitouni, Saad Harous, and Ramdane Maamri. A distributed approach to the multi-robot task allocation problem using the consensus-based bundle algorithm and ant colony system. *IEEE Access*, 8:27479–27494, 2020.

[43] Robert Zlot and Anthony Stentz. Market-based multi-robot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.

Appendix

A Dataset Details

The detailed information about the three types of tasks in the constructed dataset is shown in Fig. A.1. For each type of tasks, We show the different action types, their instructions and sub-tasks tuples information.

Figure A.1: The detailed information about the constructed dataset of three types of tasks.

B Training Methodology

Since both the task planning module and the action module take semantic map feature vector as input, and the training process of the two modules is relatively independent, we train this two module

separately. We utilize the supervised learning algorithm to train the task planning module and the imitation learning algorithm to train the action model.

The task decomposition part in the task planning module predicts the implicit receptacles, and the task allocation part predicts the distance between the agent to the target objects in the scene. The ground truth of the implicit receptacles can be obtained from the true task triple. The true distance between the agent's location and all objects can be obtained from the AI2-THOR simulator. During the training process, the task allocation part is trained to predict the distance from the agent's location to all objects. During the inference process, the agents only predict the distance to target objects. The loss function of the task planning module is defined as follows,

$$\begin{aligned} Loss_{task} &= \alpha Loss_{recep} + \beta Loss_{dis}, \\ Loss_{dis} &= Loss_{seen} + \lambda Loss_{unseen} \end{aligned} \tag{1}$$

where $Loss_{recep}$ denotes the entropy loss between the predicted receptacles and the true receptacles, $Loss_{dis}$ denotes the sum of MSE loss of distance to objects. $Loss_{dis}$ contains the regression loss of the predicted distance to seen objects $Loss_{seen}$ and the regression loss of the distance to unseen objects $Loss_{unseen}$. α, β and λ are the hyper-parameters to control the training process. In our implementation, we set α and β to 1 and λ to 0.1.

The action module task predicts the next sub-goal specified by the grid of 0.25 meters given the specific sub-task embedding and semantic map feature vectors. The sub-goal is at most 1 meter from the original position on the egocentric x and y axes, that is, four grids in the simulator. Meanwhile, each sub-goal also contains the rotation angle, which is specified in units of 90 degrees, that is, the rotation in the new sub-goal can take the value of 0, 90, 180, 270. The sub-goal navigation demonstrations are generated by the heuristic shortest path algorithm. For each sub-task, we find the viewpoints in which the agent can interact with the target object and select several viewpoints with the largest bounding box area in the RGB observations as the target location. Then we generate the shortest path in the sub-goal level. Since the movement in egocentric x and y axes as well as rotation degrees are relatively independent during navigation, we predict the number of moving grids in the two directions respectively and the rotation angle. We also predict the probability of "Stop" action to monitor the progress of the sub-task. The loss function of the action module is defined as follows,

$$Loss_{sub-goal} = \gamma Loss_{x_{loc}} + \delta Loss_{y_{loc}} + \theta Loss_{rot} + \phi Loss_{stop} \tag{2}$$

where $Loss_{x_{loc}}$ denotes the cross entropy loss between the number of predicted sub-goal movement grids in egocentric x axis direction and that in demonstrations, $Loss_{y_{loc}}$ denotes the cross entropy loss between the number of predicted sub-goal movement grids in egocentric y axis and that in demonstrations, $Loss_{rot}$ indicates the cross entropy loss between the predicted rotation angle and that in demonstrations, $Loss_{stop}$ denotes the binary of cross entropy loss of the binary classification problem whether the agent stops or not. γ, δ, θ and ϕ are the hyper-parameters to control the training process. In our implementation, we set γ, δ, θ and ϕ to 1.

C Qualitative Results in Simulator

We show the successful sample of the *Easy* setting in the simulation environment in Fig. C.1.Agents execute the setting *"Put the cut away"* in the same room. The implicit receptacle is reasoned to be "cabinet", and decomposed sub-tasks are allocated to three agents based on their visual perception at the first step. Then at 3-rd step, Agent 1 and Agent 3 change their sub-tasks because of their newly obtained visual perceptions which can help them predict the distance to target objects more accurately. At 5-th step, after Agent 2 finishes its sub-task, it switches the sub-task and completes the whole task with the help of communication messages from Agent 1 and Agent 3.

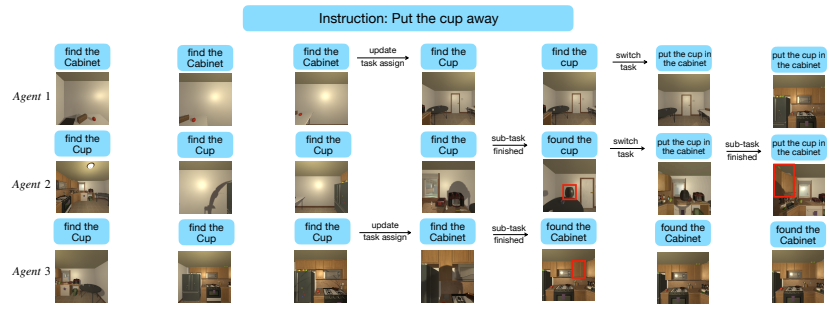

Figure C.1: Qualitative results of the *Easy* setting in the simulation.

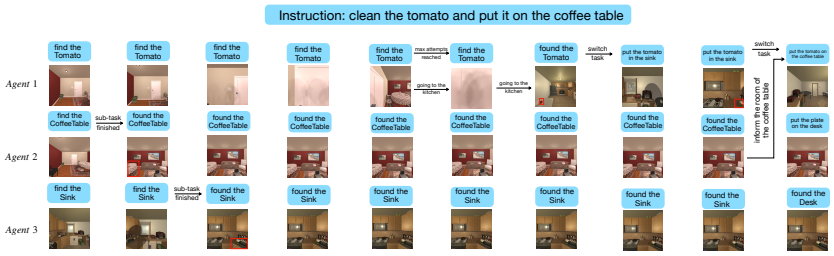

Figure C.2: The failure case in the simulation environment.

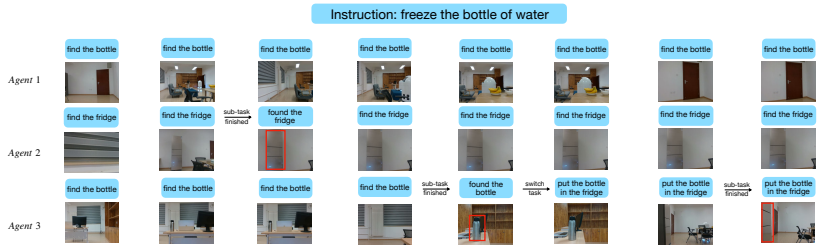

Figure D.1: Qualitative results of the *Easy* setting in the real-world scenarios.

In Fig. C.2, we show a failure case, in which the instruction is *"Clean the tomato and put it on the coffee table"*. Agent 1 and Agent 2 are in the living room and Agent 3 is in the kitchen. The optimal task allocation result is that Agent 3 washes the tomato in the sink and brings it to the living room, but in the inference process, Agent 3 chooses to find the sink at the beginning resulting in its early stop. The sub-task assigned to Agent 1 is *"find the tomato"*, and only when it reaches the maximum step limit in living room can it realize to go to the kitchen to find the tomato because of the lack of perception information in the kitchen causing by the early stop of Agent 3. Agent 1 needs to wash the tomato in the kitchen and then back to living room to put it on the coffee table, but it exceeds the maximum task steps, resulting in the task failure. The cases indicate that in *Hard* setting, the task allocation results at the first step are important for the task completion. If the allocation is reasonable at first, the overall task is more likely to be completed in fewer steps.

D Qualitative Results in Real World

We show the qualitative performance in the *Easy* setting in the real-world scenario in the same room in Fig. D.1. Agents perform the task *"Freeze the bottle of water"*. Agent 3 finds the bottle of water, then finds the fridge with the help of communication information from Agent 1 and Agent 2, and finally completes the task successfully.