LEMMA: Learning Language-Conditioned Multi-Robot Manipulation

Ran Gong, Xiaofeng Gao, Qiaozi Gao, Suhaila Shakiah, Govind Thattai, and Gaurav S. Sukhatme, *Fellow, IEEE*

***Abstract*—Complex manipulation tasks often require robots with complementary capabilities to collaborate. We introduce a benchmark for LanguagE-Conditioned Multi-robot MAnipulation (LEMMA) focused on task allocation and long-horizon object manipulation based on human language instructions in a tabletop setting. LEMMA features 8 types of procedurally generated tasks with varying degree of complexity, some of which require the robots to use tools and pass tools to each other. For each task, we provide 800 expert demonstrations and human instructions for training and evaluations. LEMMA poses greater challenges compared to existing benchmarks, as it requires the system to identify each manipulator's limitations and assign sub-tasks accordingly while also handling strong temporal dependencies in each task. To address these challenges, we propose a modular hierarchical planning approach as a baseline. Our results highlight the potential of LEMMA for developing future language-conditioned multi-robot systems.**

***Index Terms*—Data Sets for Robot Learning, Natural Dialog for HRI, Multi-Robot Systems.**

I. Introduction

THERE is growing interest in connecting human language to robot actions, particularly in single-agent systems [1], [2], [3], [4], [5]. However, there remains a research gap in enabling multi-robot systems to work together in response to language input.

Recent vision and language tasks have primarily focused on navigation and object interactions [4], [6], [7]. However, the lack of physical manipulation in these works makes the settings oversimplified. Although some recent studies, such as [1], [5], address vision and language object manipulation in single-robot settings, the language instructions provided specify only short-term goals, neglecting long-term objectives. [8] attempt to address these limitations by exploring long-horizon planning with manipulation for individual robots. Nevertheless, there remains a need to investigate multi-robot systems capable of accomplishing a broader range of long-horizon tasks while following language instructions.

Learning policies for multi-robot systems introduces distinct challenges, including diverse capabilities arising from physical constraints such as the location and reach of different robots. Moreover, task planning heavily depends on the spatial and physical relations between the objects and robots, in addition to the geometries of the objects. To ensure suitable task assignments, an awareness of each robot's specific physical capabilities is needed.

To tackle the language-conditioned vision-based multi-robot object manipulation problem, we have developed LEMMA, a benchmark that contains 8 types of collaborative object manipulation tasks with varying degrees of complexity. Some tasks require the robot to use tools for object-object interactions. For each task, the object poses, appearances, and robot types are randomized, requiring object affordance estimation and robot capability understanding. To enable multi-task learning, each task is paired with an expert demonstration and several language instructions specifying the task at different granularities. As a result, LEMMA introduces a diverse range of challenges in multi-robot collaboration, including physics-based object manipulation, long-horizon task planning, scheduling and allocation, robot capability and object affordance estimation, tool use, and language grounding. Each aspect poses distinct challenges and is crucial for a multi-robot system that follows human instructions to complete tasks. To evaluate existing techniques on LEMMA, we further provide several baseline methods and compare their performance to each other. We assess task performance by utilizing the latest language-conditioned policy learning models. Our results indicate that current models for language-conditioned manipulation and task planning face significant challenges in LEMMA, especially when dealing with complex human instructions.

We make the following contributions:

- We design eight novel collaborative object manipulation tasks involving robots with different physical configurations implemented in Nvidia Omniverse - Isaac Sim.
- We provide an open-source dataset comprising 6,400 expert demonstrations and natural language instructions, including human and high-level instructions.
- We implement a modular hierarchical planning approach as a baseline, which integrates language understanding, task planning, task allocation, and object manipulation.

II. Related Work

Language Conditioned Manipulation: Recent research has shown a growing interest in connecting human language to robot actions [2], [3], [5], [6], [9], [10], [11], [12]. However,

Manuscript received 18 April 2023; accepted 21 August 2023. Date of publication 7 September 2023; date of current version 15 September 2023. This letter was recommended for publication by Associate Editor L. Rozo and Editor A. Faust upon evaluation of the reviewers' comments. This work was supported by Amazon Alexa AI. *(Corresponding author: Xiaofeng Gao.)*

Ran Gong is with the Center for Vision, Cognition, Learning, and Autonomy, UCLA, Los Angeles, CA 90025 USA (e-mail: nikepupu@ucla.edu).

Xiaofeng Gao, Qiaozi Gao, Suhaila Shakiah, and Govind Thattai are with the Amazon Alexa AI, San Jose, CA 95129 USA (e-mail: xfgao@g.ucla.edu; qiaozikl@gmail.com; ssshakia@amazon.com; gowin.thattai@gmail.com).

Gaurav S. Sukhatme is with the Amazon Alexa AI, San Jose, CA 95129 USA, and also with the Department of Computer Science, USC Viterbi School of Engineering, Los Angeles, CA 90089 USA (e-mail: gaurav@usc.edu).

Project website: https://lemma-benchmark.github.io

Digital Object Identifier 10.1109/LRA.2023.3313058

TABLE I
COMPARISON WITH OTHER BENCHMARKS.

Benchmark	Alfred[6]	MQA[21]	Calvin[2]	M-EQA[13]	Ravens[22]	Vlmbench[3]	CH-MARL[23]	TBP [16]	EMATP[14]	LEMMA
Language	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓
Multi-task	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓
Manipulation	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓
Multi-agent	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓
Tool Use	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Temporal Dep.	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓

LEMMA evaluates the performance of language-conditioned multi-agent object manipulation in long-horizon tasks. Multi-task: using a multi-task setting. Language: language instructions to specify goal. Manipulation: physical object manipulation. Multi-agent: requiring multiple agents for task completion. Tool use: requiring the robot to use a tool to interact with other objects. Temporal Dep: temporal dependency between sub-tasks.

these studies typically use a single-robot setting. In contrast, our work emphasizes multi-robot collaboration. In particular, our task settings are designed to require the collaboration of two robots to successfully complete the task. In contrast, in [13], [14], a single agent can still finish the task although multiple agents are available. In our setting, due to the limited workspace reachability of each robot, each target object can only be manipulated by one robot initially, making collaboration necessary to achieve the task goals.

Visual Multi-Agent Collaboration: Visual multi-agent collaboration has attracted attention in recent embodied AI research [13], [14], [15], [16], [17], [18], [19]. However, among the works that involve object manipulation, simplified non-physics based atomic actions are often employed as an abstraction for manipulation. These works often use a magic glove to attach an object to the gripper as long as hand-crafted conditions are met. For example, an object within 15 cm to the gripper can be automatically snapped to the gripper [20]. While this simplification does not affect learning robot task planning, it is unsuitable for learning low-level manipulation policies. We do not make such a simplification here instead using a gripper to interact with objects physically. Additionally, most previous works study the collaboration problem in a single-task setting. In contrast, our work uses a multi-task setting requiring the comprehension of a textual description to understand the goal.

Bimanual Robot Manipulation: There is a rich set of literature on bimanual robot manipulation [24], [25], [26], [27], [28], [29], [30]. These works address important problems in dual-arm coordination with a focus on coordinated control and collision avoidance. However, there is less exploration of multi-robot task planning and allocation for long-horizon tasks with strong temporal dependencies, along with workspace management. In addition, these works typically do not involve vision and language inputs, especially for the recognition of the physical limitations of different robots from vision input. More importantly, previous research usually employs robot arms of the same type. In contrast, our work considers the settings of both heterogeneous and homogeneous robot arms.

Visual Robot Task and Motion Planning: Traditionally, most works in this area use search over pre-defined domains for planning, which require extensive domain knowledge and accurate perception. Moreover, they often scale poorly with an increasing number of objects. Another line of work involves generating task and motion plans given scene images [31], [32]. In contrast, in our settings, the model uses both RGBD images and textual descriptions as input for multi-task learning. Most recently, [33] generated robot policies in the form of code tokens using large language models (LLMs). Nonetheless, they only focus on single-agent task planning. We compare and contrast our benchmark with existing works in Table I.

III. PROBLEM FORMULATION

Assume a robot system comprised of N robots that is tasked to complete a complex manipulation task, the goal of which is specified by a language instruction $x_L = \{x_l\}_{l=1}^{L}$, which is a sequence of L word tokens. The full task can be decomposed into M sub-tasks $d_M = \{d_m\}_{m=1}^{M}$. d_M represents the full task, and d_m represent each sub-task in d_M. Given the language instruction x_L, our goal is to find a valid and optimal sub-task allocation. We define q_{im} and c_{im} as the quality and cost, respectively, for allocating robot i to work on sub-task m. Then the combined utility for the sub-task is:

$$u_{im} = \begin{cases} q_{im} - c_{im}, & \text{if robot } i \text{ can execute sub-task m} \\ -\infty. & \text{otherwise} \end{cases}$$

We define the assignment of sub-task m to robot i as

$$v_{im} = \begin{cases} 1, & \text{robot } i \text{ is assigned to sub-task m} \\ 0. & \text{otherwise} \end{cases}$$

with $\gamma_m = i$ being an assignment variable for each sub-task m, indicating that sub-task m is assigned to robot i.

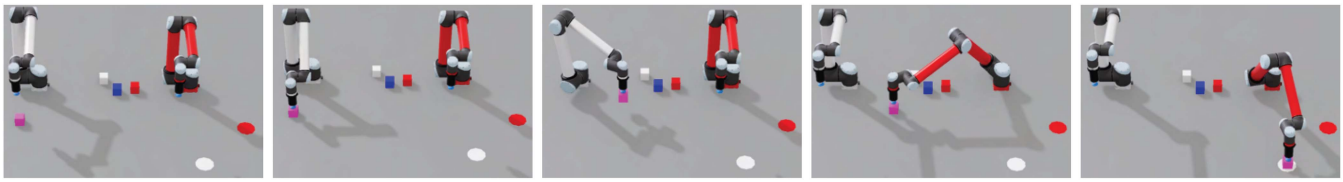
The goal is to maximize the utility of the full manipulation task under a time constraint. Defining the execution time for task m by robot i as τ_{im}, and the maximum time allowed to execute the task as T_{max}, we can express the task decomposition and assignment problem as follows:

$$\arg\max_{v} \sum_{i=1}^{N} \sum_{m=1}^{M} u_{im} v_{im} \tag{1}$$

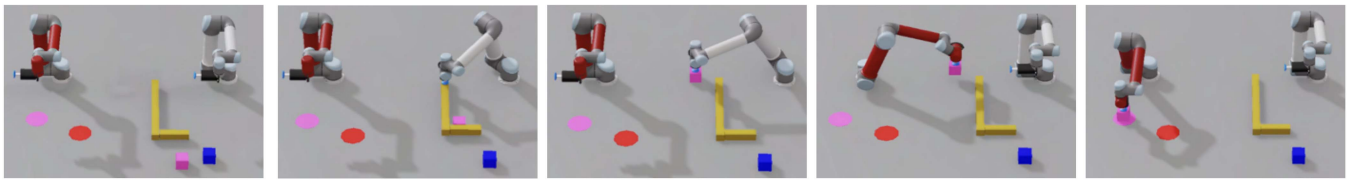
Subject to:

$$\begin{aligned} \sum_i \sum_m \tau_{im} v_{im} &\leq T_{max} \\ \sum_i v_{im} &\leq 1 \qquad \forall m \in M \\ v_{im} &\in \{0,1\} \quad \forall i \in N, \forall m \in M \end{aligned}$$

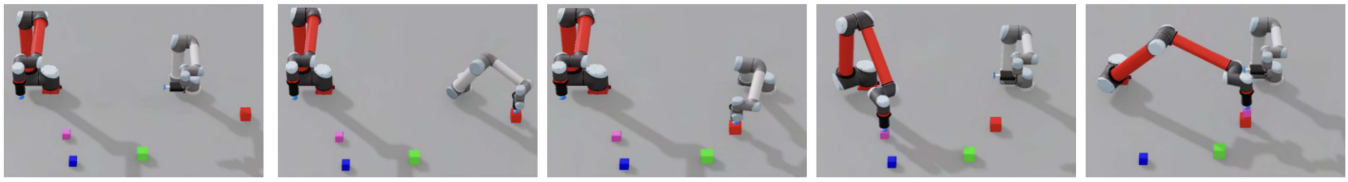
As pointed out by [34], this problem cannot be solved in polynomial time. In this work, we tackle this problem by learning from expert demonstrations, so that each sub-task can be assigned to a capable robot to ensure successful task execution. With the sub-task d_m and its assignment γ_m, an object manipulation policy $\pi(s^{t+1}|s^t, o_N^t, x_L, d_m, \gamma_m)$ can be used to move a specific object from its current pose s^t to its target pose s^{t+1}, given the observations $o_N^t = \{o_i^t\}_{i=1}^{N}$ and language instruction



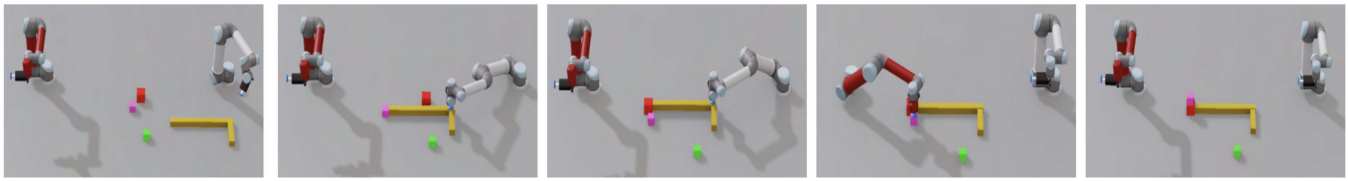
(a) Pass. Instruction: Place the pink cube on the white pad. Robots: UR10 and UR10.



(b) Hook. Instruction: Place the pink cube on the pink pad. Robots: UR5 and UR5.



(c) Stack. Instruction: Place the pink cube on top of the red cube. Robots: UR5 and UR10.



(d) Poke2stack. Instruction: Place the pink cube on top of the red cube. Robots: UR5 and UR5.

Fig. 1. Expert demonstrations and high-level instructions of tasks in LEMMA . We use minimal instructions that specify the goal, thus, it is possible that two different tasks may have the same instruction. E.g., tasks in (c) and (d) have the same instruction, but (d) requires robots to make use of the tool to poke the blocks so that the red robot can reach them. Note that the pair of robots involved in each demonstration can be different. The pair of robots in homogeneous settings (e.g., a, b, and d) have the same reach, while in heterogenous cases the reach can be different for each robot (e.g., c).

x_L. In this work, the observation O_N consists of robot joint configurations, RGBD images associated with each robot, and camera parameters.

IV. LEMMA Benchmark

We introduce LEMMA to address the language-conditioned multi-robot manipulation problem. This benchmark is designed to evaluate a system's ability to dynamically perform task allocation and object manipulation in a tabletop environment. LEMMA sets itself apart from existing language-conditioned robotic manipulation benchmarks, such as [2], [3], [22], in several key aspects:

- All tasks in LEMMA feature strong temporal dependencies, making the execution order of sub-tasks critically important. Out-of-order execution will result in task failure.
- LEMMA tasks are exclusively multi-agent based. Due to the robots' reachable space limitations, it is impossible to complete tasks in LEMMA using a single agent.
- As illustrated in Fig. 1, robots are provided with only minimal high-level instructions. This requires the model to have a deep understanding of the environment and plan a sequence of actions accordingly to reach the goal specified by the instruction. Using a language classifier to determine the task type and a template to form task plans, as in [35], is inadequate, as multiple tasks may share the same language instructions.
- LEMMA allows robots with different physical configurations to collaborate on manipulation tasks. Two types of robots are provided in LEMMA, namely UR10 and UR5.

A detailed comparison between LEMMA and other related benchmarks is shown in Table I.

A. Task Settings

In LEMMA, we focus on tasks that require a multi-robot system to complete, given a single instruction and visual observations from top-down cameras placed above each robot. Specifically, we consider a two-robot setting under centralized control. The high-level goal is specified by a single natural language instruction such as "Place the red block on top of the white pad". However, this instruction may not provide all the necessary details on how to complete the task. To thoroughly assess system performance in language-conditioned multi-robot collaboration, we have designed 8 tasks of varying difficulty. The tasks are

TABLE II
EIGHT TYPES OF TASKS IN LEMMA

Task Type	Pass	Pass2	Stack	Stack2	Poke	Poke&Stack	Hook	Hook&Stack
Objects Categories	Cube, Pad	Cube, Pad	Cube	Cube	Cube, Pad, Stick	Cube, Stick	Cube, Pad, Stick	Cube, Stick
Using Tools	✗	✗	✗	✗	✓	✓	✓	✓
Passing Tools	✗	✗	✗	✗	✗	✗	✗	✓
Number of Objects	3-6	6-8	2-5	4-6	4-6	3-5	3-5	3-6
Number of Sub-tasks	2	4	2	4	3	5	4	7

Tasks require 2-7 sub-tasks, with each sub-task requiring the robot to pick up an object, move to a location and put it down. Some tasks require the robots to use tools. In addition, the most difficult task, Hook&Stack, requires passing the tool from one robot to the other.

implemented in a simulated tabletop environment in NVIDIA Omniverse Isaac-Sim. Task statistics can be found in Table II.

Pass: The first robot is required to pick up a cube of a designated color in its own reachable workspace and place it within a shared workspace, i.e. a space that is reachable by both robots. Following this, the second robot must pick up the designated cube and position it on top of a specified pad in its reachable workspace.

Stack: The first robot picks up a designated cube in its reachable workspace and places it in the shared workspace. Subsequently, the second robot picks up a different designated cube and stacks it on top of the first cube.

Poke: Initially, the cube is unreachable by either robot. One robot has access to a tool and must use it to poke the designated cube into the other robot's reachable workspace. The second robot then picks up the cube and places it in the target pad in its reachable workspace.

Hook: Initially, the cube is unreachable by either robot. However, one of the robots can use a tool to hook the cube into its own reachable space. After that, the robot need to move it to the shared workspace so that the other robot can pick it up and position it on the target pad.

Pass2: As an extension of Pass, the task requires the robots to pass two different objects to each other and place them in their respective target locations.

Stack2: As an extension of Stack, the task requires the robots to construct two block towers, each requires two cubes of different designated colors.

Poke&Stack: The task requires one robot to use a tool to poke two designated cubes into the other robot's workspace. The other robot then places one cube on top of the other.

Hook&Stack: The task requires one robot to use a tool to hook a designated cube into its own reachable space and pass the tool to the second robot, allowing it to perform its hook action on the second cube. The first robot then moves the first hooked cube to the shared workspace so that the second robot can pick it up and stack it on the second block.

B. Action Space and Observation Space

The default action space of each robot is the end-effector position. Nevertheless, alternative control mechanisms such as joint positions, joint velocities, and joint torques can also be accommodated for controlling 6 degrees-of-freedom robots. Due to physical constraints, each robot has a unique reachable workspace, allowing it to only interact with a limited set of objects in the task.

To obtain visual observations, we place a fixed RGBD camera above each robot. Each camera has a limited field of view and cannot capture all the objects. We follow [1] to process the vision input: we first generate the scene point cloud based on all RGBD images and camera parameters, and use the point cloud to obtain the top-down orthographic RGBD reconstruction of the scene.

C. Task Generation

For diversity among the tasks, we use a rejection sampling mechanism to generate task instances. Each task instance specifies the initial environment configurations and goal conditions. To ensure that the task completion requires both robots, we make sure that the initial location of the target object falls into the reachable workspace of only one robot. In addition to the target object, we also add some distractor objects to make the task more challenging:

1) Specify each robot's type, location, and color. There are two robot types, UR5 and UR10, respectively. Each robot has two different colors, red and white. Colors and types are randomly assigned to each robot.
2) Sample the goal condition of the task, including the colors of the target objects and their goal locations, according to the task type. E.g., the goal condition for the *Stack pink on red* task is satisfied when a pink block is on top of a red block, as shown in Fig. 1(c). There are five available colors for blocks and pads, including pink, red, white, blue, and green. The color of the tool is always yellow.
3) Sample the initial locations of target objects, so that each object is only within the reachable space of a single robot.
4) Sample the colors and locations of distractor objects while making sure the colors of the target objects are unique.

The above sampling mechanism is repeated until one valid task instance is generated.

D. Expert Demonstration

Given a task specification, we use an oracle task and motion planner to create expert demonstrations. Based on the initial configurations and goal conditions of the task, the oracle creates a task plan consisting of a sequence of sub-tasks d_M, and the allocated robot γ_m for each sub-task d_m. The sub-task follows a pick-and-place procedure, specifying the target object to pick up and the target pose at which to place the object. Once the task plan is generated, an RMPflow motion planner is utilized to generate a motion plan based on ground truth object locations at each time step. All the motion plans are executed by the designated robot, and the corresponding RGBD images and camera poses are recorded to form the expert demonstration data.

TABLE III
EXAMPLE HIGH-LEVEL INSTRUCTION AND CROWD-SOURCED HUMAN LANGUAGE INSTRUCTION TEMPLATES TO SPECIFY MANIPULATION TASKS IN LEMMA

Task Type	High-Level Instruction	Human Instruction
Pass	place the *pick-color* cube on top of the *place-color* pad	pick a *pick-color* cube from one side, put it at the center and put it on the *place-color* circle
Pass2	place the *pick-color1* cube on top of the *place-color1* pad and place the *pick-color2* cube on top of the *place-color2* pad	place the *pick-color1* cube on the *place-color1* pad and *pick-color2* cube on the *place-color2* pad in the opposite direction
Stack	place the *pick-color* cube on top of the *place-color* cube	take *place-color* block and place it in center under the *pick-color* block
Stack2	place the *pick-color1* cube on top of the *place-color1* cube and place the *pick-color2* cube on top of the *place-color2* cube	assemble two block towers by placing the *place-color1* cube under the *pick-color1* cube and the *place-color2* cube under the *pick-color2* cube
Poke	place the *pick-color* cube on top of the *place-color* pad	use the L object to push the *pick-color* block to the other robot and place it on the *place-color* pad
Poke&Stack	place the *pick-color* cube on top of the *place-color* cube	grab the brown L to push the *pick-color* and *place-color* cubes closer to the other robot, so it can grab and place the *pick-color* cube on top of the *place-color* cube
Hook	place the *pick-color* cube on top of the *place-color* pad	fetch the *pick-color* cube using the L shaped object and place it on top of the *place-color* pad
Hook&Stack	place the *pick-color* cube on top of the *place-color* cube	use the grippers to pick up the *pick-color* block from the hook and stack it on the *place-color* block

Pick-color and *place-color* represent the color of objects being picked up and placed on.

E. Language Instructions

We assign high-level instruction and a human instruction to each task instance. For high-level instruction, we manually define a template for each task and lexicalize the template using the goal of the task. For human instruction, we first crowd-sourced 500 templates on Amazon Mechanical Turk and selected 80 valid templates that are not too verbose and uniquely specify the goal given the visual observation, 10 for each task type. For each task instance, we then sample a template from the template pool and lexicalize it to generate human instruction. Some examples of the instruction templates can be found in Table III.

In summary, each datapoint in LEMMA contains the following information: 1) a manipulation task, specified by the initial configurations and goal conditions, 2) an expert demonstration, including camera poses and RGBD images at each timestep, 3) a high-level instruction and a human instruction specifying the task. As a result, we generate 800 data sessions for each task type, with a total of 6400 sessions. For each task type, we keep 700 sessions in the *training* set, 40 in the *validation* set, and 60 in the *test* set.

F. Evaluation Metrics

We have adopted success rate as the evaluation metric in LEMMA . Task success is defined as 1 if the task goal-conditions are met at the end of the episode, and 0 otherwise. The time constraint of each episode T_{max} is set to 100 seconds. The task specified by the instruction *Place the red block on the green circle*, for example, is considered successful if, at the end of the episode, the red block is on top of the green circular pad.

V. BASELINE MODELS

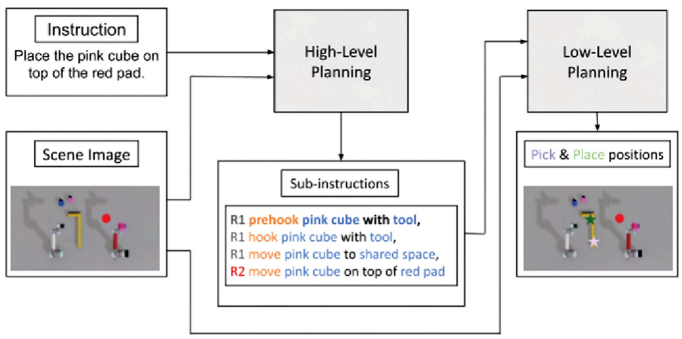
Consider the problem of learning multi-agent task and motion planning with two robots given a single language instruction and visual observations. The problem can be decomposed into two sub-problems: (a) multi-robot task planning and task allocation, and (b) single-robot planning given the assigned sub-task. To this end, we design a modular baseline model which includes a high-level planner for deciding which sub-task a robot should work on and a low-level planner for generating pick and place locations for the gripper given the assigned sub-task. Fig. 2(a) shows the overall architecture of our baseline model. We follow [1] to use the fused point clouds generated from the RGB and depth images of two cameras. Then we use the top orthographic projection of the point cloud as the visual input.

A. Action Primitives

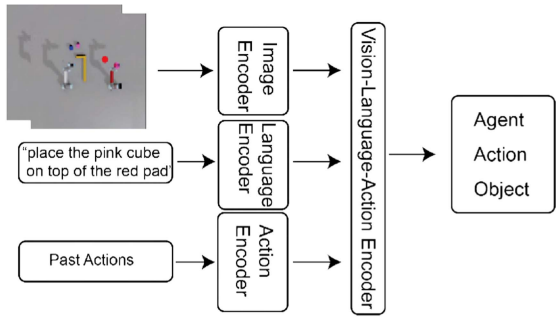
We define six action primitives: *move, prehook, hook, prepoke, poke* and *stop*. The non-stop action primitives follow generalized pick-and-place settings. *Move* is the standard pick-and-place primitive, requiring the robot to pick up the object and move it to a specified location. *Prehook* and *prepoke* require the robot to pick up the tool and align it with the target object to prepare for *hook* and *poke* respectively. The required height of the gripper is different for each primitive. For *hook* and *poke*, the height of the gripper after picking is set at 5 cm, enabling the tool to come into contact with the target object. For other primitives, the height of the gripper is set to 30 cm to avoid contact with other objects between the pick and place actions. The sequence of action primitives required to complete the task depends on the relative poses of target objects, such as cubes, tools, and pads. As a result, the same language instruction can correspond to completely different sequences of primitive actions, depending on the specific arrangement of these objects.

B. Multi-Robot Task Planning and Task Allocations

The multi-robot collaboration task can be decomposed into a sequence of sub-tasks, each can be completed by a single robot. The sub-task allocation can be represented by a tuple $(d_m^t, g_m^t, \gamma_m^t)$. $g_m^t = (e, p, q)$ represents the entities to specify the sub-task d_m^t, including the action primitive e, the pick entity p and place entity q. E.g. $(\text{Move}, \text{Red Cube}, \text{Shared Space})$ indicates moving a red cube on top of the shared workspace between two robots. γ_m^t is the sub-task assignment indicating which robot is to perform the task. $(d_m^t, g_m^t, \gamma_m^t)$ are further lexicalized using templates to form the sub-instruction specifying the sub-task and



(a) Overall architecture of the baseline model.



(b) The architecture of the Episodic Transformer model used as the high-level planner.

Fig. 2. Our baseline model involves a high-level task planning module and a low-level planning module. The high-level planning module takes as input the top-down fused scene image and human instruction to generate sub-instructions that assign the corresponding sub-tasks to robots based on their limitations. Each sub-task is specified by action primitives (in orange) and objects (in blue). The low-level planning module uses the sub-instruction and top-down projection of the scene to generate pick and place locations (visualized as purple and green stars respectively) of the gripper in the scene.

its allocation (Fig. 2(a)). In practice, we use Episodic Transformer [36], a vision and language task planner to generate the sub-instructions. The approach is a language-conditioned visual task planning method that employs a transformer architecture for long-horizon planning. As shown in Fig. 2(b), ET uses the historical visual and language information in the entire episode to capture long-term dependencies between actions. It leverages the transformer architecture to first separately encodes image histories, language instructions, and past action histories, and then perform cross-attention across modalities to decode robot assignment, action primitives, and the target object separately.

The choice between neural-based open-loop planning and closed-loop planning is often debatable. Open-loop planning cannot adapt to errors made in the planning process. However, it is more stable to train since the training distribution is often more aligned with the testing distribution. Here we consider both open-loop planning and close-loop planning for our benchmark and compare their performance. Note the original Episodic Transformer is closed-loop only and requires new observation at each time step to plan the next action (i.e. *Single-step*). We modify the algorithm to use only the initial visual observation by providing it as input repeatedly during the loop to plan the whole sub-instruction sequences (i.e. *Multi-step*).

C. Single Agent Object Manipulation and Grounding

In this work, we use CLIPort [1] as the low-level planner. Formally, at each time step, the algorithm focuses on learning a goal-conditioned policy π that produces actions $\mathbf{a}^t$ based on the current visual observation o^t and a language instruction $x_L^t = \{x_l^t\}_{l=1}^L$. The visual observation o^t is an orthographic top-down RGB-D reconstruction of the scene, where each pixel corresponds to a point in 3D space. As shown in Fig. 2(a), in our use case, each input language instruction $x_L^t = (g_m^t, \gamma_m^t)$ specifies a sub-task being allocated to robot γ_m^t at time step t. As a result, the goal-conditioned policy is defined as

$$\pi\left(o_N^t, x_L^t\right) = \pi\left(o_N^t, g_m^t, \gamma_m^t\right) \rightarrow \mathbf{a}^t = \left(\mathscr{T}_{\text{pick}}, \mathscr{T}_{\text{place}}\right) \in A,$$

where the actions $\mathbf{a} = (\mathscr{T}\text{pick}, \mathscr{T}\text{place})$ denote the end-effector poses for picking and placing respectively. CLIPort is designed for tabletop tasks, with $\mathscr{T}\text{pick}, \mathscr{T}\text{place} \in \mathbf{SE}(2)$.

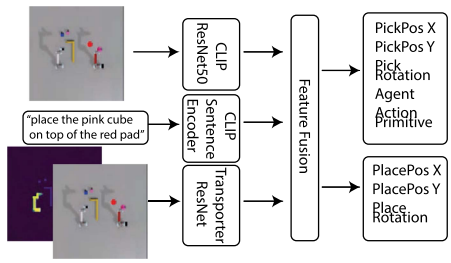


Fig. 3. Multi-Agent Cliport Model architecture. The output is extended to include robot assignment and action primitive for each predicted action.

D. Multi-Agent Cliport

Since the sub-instruction already contains the sub-task allocation γ_m^t and action primitive e, the CLIPort module used as our low-level planner only needs to predict the pick and place location. To compare with this modular approach, we present a modified version of the original CLIPort module (i.e. M-CLIPort) to perform task planning and task allocation explicitly in an end-to-end fashion. We extend the output to a higher dimensional vector to predict the robot assignment and action primitive to use in addition to pick and place locations, as shown in Fig. 3.

VI. Experiments

A. Evaluation Workflow

We train the CLIPort module for 300 K steps on the training set and save a checkpoint every 20 K steps. Then we perform checkpoint selection on the validation split for both the high-level planner and the low-level planner. For the low-level planner, we use the ground-truth sub-instructions as input for checkpoint selection. For the high-level planner, we train the Episodic Transformer model for 30 epochs and save a checkpoint at the end of each epoch. We choose the best checkpoint based on the accuracy of predicted task plans on the validation split. We report the performance of a combination of best-performing high-level and low-level checkpoints on the test set. Since the physics and rendering in Isaac-sim are not deterministic, we

TABLE IV
PERFORMANCE ON THE TEST SET WITH **HIGH-LEVEL** INSTRUCTIONS

Task Type	Pass	Pass2	Stack	Stack2	Poke	Poke&Stack	Hook	Hook&Stack	Avg
M-CLIPort	34.33±2.19	0.00±0.00	10.67±0.82	0.00±0.00	15.33±4.64	6.00±0.82	21.67±4.34	1.67±1.50	11.21±0.76
Single-step	87.00±0.67	27.00±1.25	41.67±0.00	30.00±3.86	28.00±3.86	10.33±1.24	86.70±2.36	35.00±2.36	43.21±0.45
+ GTA	**100.00±0.00**	**84.67±2.67**	**91.33±0.67**	72.00±0.67	46.67±0.00	32.33±1.70	**98.33±0.11**	**73.33±2.36**	**74.83±0.44**
Multi-step	83.06±1.15	28.06±1.78	28.89±0.79	31.11±2.48	30.83±2.31	9.17±1.27	82.78±1.57	24.72±4.66	39.83±0.84
+ GTA	**100.00±0.00**	83.89±3.00	90.00±0.00	**74.72±1.78**	**55.28±0.62**	**35.56±1.24**	96.94±0.62	58.34±1.93	74.34±0.56

M-CLIPort: Multi-agent Cliport. Single-step: high-level planning generates each sub-instruction based on the new observation. Multi-step: high-level planning generates the complete sub-instruction sequences from the initial observation. GTA: replacing the robot task allocation results from either single-step or multi-step planning by the ground truth while preserving the predicted action primitives and objects.
The bold values represent the best average performance.

TABLE V
PERFORMANCE ON THE TEST SET WITH **HUMAN** INSTRUCTIONS

Task Type	Pass	Pass2	Stack	Stack2	Poke	Poke&Stack	Hook	Hook&Stack	Avg
M-CLIPort	25.83±1.86	0.00±0.00	20.00±2.36	0.00±0.00	4.58±1.38	1.25±1.38	3.75±0.72	0.00±0.00	6.93±0.47
Single-step	44.44±2.29	1.67±0.00	21.67±1.36	0.00±0.00	13.88±1.24	2.50±0.83	25.56±1.57	9.17±1.60	14.86±0.26
+ GTA	63.33±0.00	**5.56±1.24**	59.44±1.24	**1.39±0.62**	**25.28±0.62**	**9.17±1.27**	**30.00±0.00**	**19.72±0.62**	26.74±0.20
Multi-step	58.33±1.18	0.00±0.00	38.54±0.99	0.00±0.00	7.71±2.92	2.92±1.10	21.46±0.99	3.96±1.16	16.61±0.39
+ GTA	**81.89±0.55**	5.00±1.44	**70.42±0.72**	0.00±0.00	15.00±0.00	8.33±1.67	23.33±0.00	14.17±1.87	**27.27±0.30**

The bold values represent the best average performance.

evaluate all tasks for 10 runs and report the means and standard deviations.

B. Experiment Results

1) High-Level Instructions: The results shown in Table IV compare language-conditioned policies with different task-planning modules. The M-CLIPort fails for all tasks with longer horizons. In comparison, our modular hierarchical planning approach works reasonably well for most tasks but fails for very long-horizon tasks (i.e. poke&stack and hook&stack). As an ablation, we further supply ground truth task allocation to the planning module (i.e. GTA). The results show that system performance can be greatly improved with ground truth task allocation. This indicates that task allocation is quite challenging since it requires understanding the reachable workspace of robots to determine which robot should be assigned to a certain sub-task.

2) Human Instructions: The results shown in Table V demonstrate the system performance under human instructions. Human instructions are more complex than high-level instructions since they feature different input lengths, levels of detail, and word choices. Our results show there is a significant gap between the performance of high-level instructions and human instructions, showing that current models are not capable of handling the increased complexity in language. Among all the results, models perform significantly worse on Stack2 and Pass2 with human instructions. This discrepancy is likely due to the fact that the orders of objects vary in human instructions for these tasks (e.g. Place the red cube under the white cube" vs "Place the white cube on top of the red cube"), as demonstrated in Table III, which poses greater challenges in language understanding. In addition, for tasks requiring tool use, human instructions do not always involve the tools, e.g. one instruction for poke is "push the red block toward the other robot and put it on the blue circle". In these cases, the model often fails to predict the correct object to manipulate.

3) Further Analysis: To provide more insight into the factors affecting task performance, we further show a breakdown of performance under different settings, including robot types and the number of distractors in the scene. We observe that in general, the task performance decreases with an increasing number of distractor objects (Table VI). As for the robot types, the collaborations among two UR10 s exhibit significantly higher performance using the hierarchical planning model given high-level instructions (Table VII). This is probably due to the fact that UR5s have a smaller reachable space compared to UR10 s, making it more critical to accurately predict the reachable workspace of each robot.

TABLE VI
IMPACT OF DISTRACTORS (SINGLE-STEP PLANNING)

No. Distractors	0	1	2
Pass	**91.67±0.00**	87.06±2.35	81.18±3.50
Pass-human	**75.00±0.00**	53.92±2.19	37.25±5.55
Stack	**46.15±0.00**	40.00±3.08	16.92±2.86
Stack-human	16.67±2.87	**28.33±2.36**	15.38±6.28
Poke	**33.08±7.14**	30.00±4.44	17.50±2.50
Poke-human	**16.03±2.64**	12.96±2.62	11.46±6.67
Hook	**91.30±3.89**	83.16±3.94	84.44±7.37
Hook-human	26.09±2.51	**32.46±3.62**	17.59±2.07

The bold values represent the best average performance.

TABLE VII
IMPACT OF ROBOT TYPES

Robot Type	UR5&UR5	UR5&UR10	UR10&UR10
Multi-step	38.60±3.25	36.32±0.47	**46.89±1.69**
Multi-step-human	15.71±0.59	**19.16±0.83**	13.06±0.53
Single-step	37.44±1.00	41.05±1.30	**51.94±1.17**
Single-step-human	**16.81±0.40**	15.57±0.33	11.94±0.96
M-CLIPort	**13.50±0.84**	11.27±0.51	9.10±2.23
M-CLIPort-human	6.62±0.93	**8.52±1.25**	4.48±1.06

The bold values represent the best average performance.

VII. CONCLUSION, LIMITATIONS AND FUTURE WORK

In this letter, we presented LEMMA, the first public benchmark for language-conditioned multi-robot tabletop manipulation. LEMMA combines the problems of language grounding, task planning, task allocation, tool use, capability estimation, long-horizon manipulation, and multi-modal scene understanding.

All these subproblems of LEMMA pose significant challenges for existing algorithms. We plan to open-source the simulation environment, the generated dataset, the baseline models, and other tools used during the project's development stage, and we hope our benchmark can inspire new methods in these areas. In this work, we assume tasks can be easily decomposed into independent sub-goals and we restrict the object manipulation problem to SE2. Extending it to SE3 with a more diverse set of tasks and objects will be an important future direction. In our setting robots have different capabilities if they have different reachable spaces. Of course, robots can differ in multiple other ways such as payload, gripper types, etc. We use instructions generated by templates, which are not as diverse as human natural language instructions. Leveraging LLMs to produce more diverse instructions can be a promising way to increase linguistic diversity. This work did not specifically focus on bimanual robot control, and one important future direction is to explore challenges inherent to bimanual control including dual-arm coordination and collision avoidance with real robot experiments. However, this study is the first of its kind to study vision-based language-conditioned multi-robot collaboration and our simplifications serve as a reasonable starting point for research in this area.

REFERENCES

[1] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and where pathways for robotic manipulation," in *Proc. 5th Conf. Robot Learn.*, 2022, pp. 894–906.

[2] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7327–7334, Jul. 2022.

[3] K. Zheng, X. Chen, O. C. Jenkins, and X. E. Wang, "VLMBENCH: A compositional benchmark for vision-and-language manipulation," in *Proc. Neural Inf. Process. Syst. Track Datasets Benchmarks*, 2022, 665–678.

[4] P. Anderson et al., "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3674–3683.

[5] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proc. Conf. Robot Learn.*, 2023, pp. 785–799.

[6] M. Shridhar et al., "ALFRED: A benchmark for interpreting grounded instructions for everyday tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10740–10749.

[7] X. Gao, Q. Gao, R. Gong, K. Lin, G. Thattai, and G. S. Sukhatme, "DialFRED: Dialogue-enabled agents for embodied instruction following," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10049–10056, Oct. 2022.

[8] W. Huanget al., "Inner monologue: Embodied reasoning through planning with language models," in *Proc. Conf. Robot Learn.*, 2023, pp. 1769–1782.

[9] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," in *Proc. Robot.: Sci. Syst.*, 2021, pp. 1–18.

[10] A. Zenget al., "Socratic models: Composing zero-shot multimodal reasoning with language," in *Proc. 11th Int. Conf. Learn. Representations*, 2022, pp. 1–35.

[11] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor, "Language-conditioned imitation learning for robot manipulation tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 13139–13150.

[12] S. Nairet al., "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *Proc. Conf. Robot Learn.*, 2022, pp. 1303–1315.

[13] S. Tan, W. Xiang, H. Liu, D. Guo, and F. Sun, "Multi-agent embodied question answering in interactive environments," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 663–678.

[14] X. Liu, X. Li, D. Guo, S. Tan, H. Liu, and F. Sun, "Embodied multi-agent task planning from ambiguous instruction," in *Proc. Robot.: Sci. Syst.*, 2022, pp. 1–14.

[15] U. Jain et al., "A cordial sync: Going beyond marginal policies for multi-agent embodied tasks," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 471–490.

[16] U. Jain et al., "Two body problem: Collaborative visual task completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6689–6699.

[17] H. Wang, W. Wang, X. Zhu, J. Dai, and L. Wang, "Collaborative visual navigation," 2021, *arXiv:2107.01151*.

[18] B. Chen, S. Song, H. Lipson, and C. Vondrick, "Visual hide and seek," in *Proc. Artif. Life Conf.*, 2020, pp. 645–655.

[19] X. Liu, D. Guo, H. Liu, and F. Sun, "Multi-agent embodied visual semantic navigation with scene prior knowledge," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3154–3161, Apr. 2022.

[20] A. Szotet al., "Habitat 2.0: Training home assistants to rearrange their habitat," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 251–266.

[21] Y. Deng et al., "MQA: Answering the question via robotic manipulation," in *Proc. Robot.: Sci. Syst.*, 2020, pp. 1–10.

[22] A. Zenget al., "Transporter networks: Rearranging the visual world for robotic manipulation," in *Proc. Conf. Robot. Learn.*, 2021, pp. 726–747.

[23] V. Sharma, P. Goyal, K. Lin, G. Thattai, Q. Gao, and G. S. Sukhatme, "CH-MARL: A multimodal benchmark for cooperative, heterogeneous multi-agent reinforcement learning," 2022, *arXiv:2208.13626*.

[24] Y. Chen et al., "Towards human-level bimanual dexterous manipulation with reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 5150–5163.

[25] K. Takata, T. Kiyokawa, I. G. Ramirez-Alpizar, N. Yamanobe, W. Wan, and K. Harada, "Efficient task/motion planning for a dual-arm robot from language instructions and cooking images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12058–12065.

[26] S. Stavridis, P. Falco, and Z. Doulgeri, "Pick-and-place in dynamic environments with a mobile dual-arm robot equipped with distributed distance sensors," in *Proc. IEEE-RAS 20th Int. Conf. Humanoid Robots*, 2021, pp. 76–82.

[27] C. Smith et al., "Dual arm manipulation–A survey," *Robot. Auton. Syst.*, vol. 60, no. 10, pp. 1340–1353, 2012.

[28] H. Zhang, P.-J. Lai, S. Paul, S. Kothawade, and S. Nikolaidis, "Learning collaborative action plans from youtube videos," in *Proc. Int. Symp. Robot. Res.*, 2019, pp. 208–223.

[29] S. Stepputtis, M. Bandari, S. Schaal, and H. B. Amor, "A system for imitation learning of contact-rich bimanual manipulation policies," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 11810–11817.

[30] P. Lertkultanon and Q.-C. Pham, "A certified-complete bimanual manipulation planner," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 3, pp. 1355–1368, Jul. 2018.

[31] D. Driess, J.-S. Ha, and M. Toussaint, "Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image," in *Proc. Robot: Sci. Syst. Found.*, 2020, pp. 1–10.

[32] D. Driess, J.-S. Ha, and M. Toussaint, "Learning to solve sequential physical reasoning problems from a scene image," *Int. J. Robot. Res.*, vol. 40, no. 12-14, pp. 1435–1466, 2021.

[33] I. Singh et al., "ProgPrompt: Generating situated robot task plans using large language models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 11523–11530.

[34] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[35] S. Y. Min, D. S. Chaplot, P. K. Ravikumar, Y. Bisk, and R. Salakhutdinov, "FILM: Following instructions in language with modular methods," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–17.

[36] A. Pashevich, C. Schmid, and C. Sun, "Episodic transformer for vision-and-language navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 15942–15952.