LGMCTS: Language-Guided Monte-Carlo Tree Search for Executable Semantic Object Rearrangement

Haonan Chang, Kai Gao, Kowndinya Boyalakuntla, Alex Lee, Baichuan Huang, Harish Udhayakumar, Jingjin Yu, Abdeslam Boularias

***Abstract*— We present LGMCTS, a framework that uniquely combines language guidance with geometrically informed sampling distributions to effectively rearrange objects according to geometric patterns dictated by natural language descriptions. LGMCTS uses Monte Carlo Tree Search (MCTS) to create feasible action plans that ensure executable semantic object rearrangement. We present a comprehensive comparison with leading approaches that use language to generate goal rearrangements independently of actionable planning, such as StructFormer and StructDiffusion. We also present a new benchmark, called Executable Language Guided Rearrangement (ELGR) Bench, that contains tasks involving intricate geometry, and show limitations of task and motion planning (TAMP) solutions that are purely based on Large Language Models (LLM) such as Code as Policies and Progprompt on such tasks. Our findings advocate for the use of LLMs in generating intermediary representations rather than direct action planning in geometrically complex rearrangement scenarios, aligning with perspectives from recent literature. Our code and supplementary materials are accessible at https://github.com/changhaonan/LG-MCTS.**

I. Introduction

Everyday tasks, such as "Set up the kitchen", involve organizing objects based on verbal instructions, a process that is intuitive for humans but that presents a significant challenge for robots. The semantic rearrangement problem seeks to empower robots with the ability to reorganize a scene according to linguistic descriptions. This challenge necessitates that robots comprehend the task through natural language, and address the corresponding Task And Motion Planning (TAMP) problem effectively.

Traditionally, solving this problem requires formalizing semantic rearrangement in a symbolic representation, clearly defining the goal configuration or constraints, and using formal planners such as STRIPS and PDDL, or search-based planners like MCTS to devise a feasible plan. While this approach can yield optimal solutions, it demands expert-level knowledge to abstract a problem into a formal representation, making it less accessible to the average user.

To overcome this challenge, numerous recent studies have sought to tackle the problem directly from linguistic inputs and RGB-D observations [1]–[3]. One approach uses multi-modality transformers to establish a correlation between verbal descriptions and object positions using data generated from the simulation. Following work such as StructDiffusion [2] further improved this method by using a diffusion model to build the multi-modality solution. However, a common drawback of these methods is that they rely on an offline training stage, which makes them applicable only to trained object categories and spatial patterns.

The authors are with the Department of Computer Science, Rutgers University, 08854 New Brunswick, USA. This work is supported by NSF awards 1846043 and 2132972.

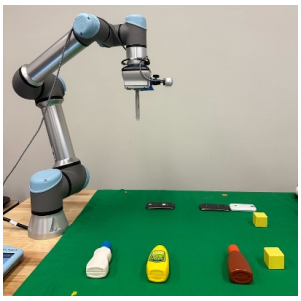

Fig. 1: Robotic Setup: a UR5e robot equipped with a RealSense D455 camera. The task is to re-arrange the objects, which are unknown to the robot, according to a natural language instruction.

With the advent of Large Language Models (LLMs), models such as GPT [4] and Llama [5], [6] have demonstrated impressive potential in understanding complex scenarios and exhibiting zero-shot planning capabilities. This has led researchers to explore the utilization of LLMs in solving language-based TAMP problems [7]–[9]. However, despite specific considerations for the feasibility of plans proposed by LLMs, it has been reported that these plans significantly lag in executability and completeness when compared to those crafted by a properly implemented traditional solver designed for the task [10]. This observation has naturally led researchers to seek methods that merge the user-friendliness of LLMs with the robustness of traditional TAMP algorithms such as PDDL, STRIPS, or MCTS. LLM-GROP [11] follows this approach in rearrangement, employing LLMs to parse user tasks from language into pairwise spatial relationship specifications and then calling a sampling-based task and motion planner [12] to generate the plan. A limitation of LLM-GROP is that it can only handle pair-wise relationships, and thus it cannot perform complex rearrangement tasks. AutoTAMP [13] uses LLMs to translate natural language into formal representations, and then invokes a planner to tackle the problem. AutoTAMP can solve a wide range of TAMP tasks, but it does not apply to general semantic rearrangement where the action space is not discrete and potentially large.

We present Language-Guided Monte-Carlo Tree Search (LGMCTS), a new technique for executable semantic object rearrangement. Like its predecessors, AutoTAMP and LLM-GROP, LGMCTS leverages LLMs for generating interme-

diate representations and employs a planner for formulating feasible plans. A key novelty of LGMCTS is the integration of parametric geometric priors for spatial relationship representations. LGMCTS facilitates more nuanced handling of complex geometric relationships among multiple objects, addressing scenarios that require organization beyond simple pairwise interactions such as configurations in lines or rectangles. Additionally, LGMCTS takes a holistic approach by simultaneously considering task planning (goal specification) and motion planning (execution order and intermediate steps). During planning, an obstacle relocation strategy is used to handle obstacles that may block the execution. This coordination ensures that plans are not only semantically coherent but also practically executable, offering a balanced consideration of goal achievement and operational efficiency.

To assess the efficacy of LGMCTS, we introduce the Executable Language-Guided Rearrangement (ELGR) benchmark, featuring over 1,600 varied language queries and robot execution checks. Our evaluations indicate that LGMCTS performs effectively on the ELGR benchmark, especially in comparison with Code as Policies and Progprompt in terms of feasibility and semantic consistency of the generated goals. LGMCTS also outperforms Structformer and StructDiffusion in goal generation on the Structformer dataset.

II. RELATED WORKS

A. Learning-based Semantic Rearrangement

The semantic rearrangement problem consists of devising a rearrangement plan that is both semantically congruent with a given language description and physically feasible. In recent years, this has gained increased traction, particularly as a pivotal application in language-driven robotics. CLIPort [1] took the initial step in this direction by merging CLIP features with a Transporter network. Yet, its design is limited to basic pick-and-place tasks. Structformer [3] advanced the field using a transformer model, by simulating rearrangements with hand-crafted rules and connecting language tokens to object poses. Leveraging Structformer's dataset, StructDiffusion [2] introduced a pose diffusion model to predict poses from language. Nonetheless, a common shortcoming amongst all these methodologies is their limitation to a single structure or pattern (e.g. circle, line) that they have been trained on, making composite patterns (e.g. rectangle + tower) a persistent challenge. Moreover, the rearrangement goals generated by these methods can be inexecutable.

B. LLM-driven Task And Motion Planning

Recent advancements in LLMs [4]–[6] have showcased impressive performance across a broad spectrum of tasks. There has been a growing interest in using LLMs for TAMP [7]–[9], [11], [14]–[30], owing to their few-shot and zero-shot reasoning ability [31]–[36]. Grounding language into a sequence of plannable tasks/actions without retraining LLMs was initially explored in [7]. Following this, SayCan [8] was proposed to facilitate the conversion of LLM-generated plans into robot-executable steps, though it struggled with addressing task execution failures. Inner Monologue [14] improved upon SayCan by incorporating real-time feedback to adjust plan post-execution, yet Inner Monologue is prone to generating suboptimal and infeasible plans. Instead of using LLMs to plan with predefined skills, other approaches such as Code as Policies [9] and Progprompt [15] leveraged LLMs for policy code generation, showcasing their potential in behavioral common sense and sequential policy logic. However, they do not show promising results on complex object rearrangement tasks requiring more nuanced spatial context. This is due to the inherent planning abilities of LLMs over long horizon tasks [10], [23].

Owing to the limitations of the aforementioned works, to offer a more reliable and interpretable planning process, recent works [10], [13], [23], [25], [28] emphasize translating natural language commands into intermediate representations that are interpretable by traditional TAMP algorithms. Text2Motion [18] uses LLMs to greedily plan a skill sequence combined with a geometric feasibility planner to ensure that the geometric dependencies are addressed. However, Text2Motion's hybrid LLM planner is less efficient in large spaces than planners such as MCTS. LLM-GROP [11] translates language instructions to symbolic spatial relationships with LLMs and employs a task and motion planner named GROP [12] to perform the rearrangement. Although GROP is optimized for efficiency and feasibility, LLM-GROP is limited by its focus on simple object rearrangements due to its treatment of multi-object semantic relationships as pairwise reasoning. AutoTAMP [13] employs LLMs for generating and validating STL representation from natural language and utilizes a formal STL planner [37] for generating optimal trajectories. Although effective across a spectrum of tasks, its applicability is limited in non-discrete action spaces, as is the case in semantic rearrangement tasks.

To address these limitations, we introduce LGMCTS, a new approach that incorporates parametric geometric priors and that is guided by MCTS's efficient exploration to provide a feasible TAMP solution for complex rearrangement tasks.

III. PRELIMINARIES

A. Problem Formulation

Semantic rearrangement is the task of rearranging a scene according to a series of natural language descriptions. One key insight here is that a goal described by language is usually a distribution rather than a single position. For example, "Put the mug at the right side of bowl" refers to a uniform distribution among a region that is right to the bowl. If we know the position distribution for each object, we just need to sequentially sample the poses for each object. The semantic rearrangement problem is then converted to a sequential sampling problem.

With this insight, we define the task of semantic rearrangement as follows. The robot is given as input a scene with objects from a set $O_S = \{o_1, o_2, \ldots, o_N\}$ and a command L, where L is a pure natural language command that implies a desired distribution list $F = \{f_i : p(o_i) \sim f_i | o_i \in O_R\}$, where $p(o_i)$ refers to the position of object o_i. Here, $O_R \subseteq O_S$ denotes the objects requiring an action, and f_i indicates the

desired pose distribution for each object. The objective is to identify an optimal action sequence, $A = (a_t)_{t=1}^{H}$, where each action a_t corresponds to moving an object o_i to a sampled position $p(o_i)$, with the objective to achieve a goal arrangement aligning L, i.e. $\prod_{o_i \in O_R} f_i(p_i) > 0$ and minimizing the number of action steps H. Noticeably, A includes not only movements of objects $o \in O_R$, but also those of distracting objects, denoted as O_D, with $O_D \subseteq O_S$.

B. Monte Carlo Tree Search (MCTS)

We provide here a brief reminder of the MCTS technique. A typical MCTS algorithm iteratively builds a search tree by performing the following four operations.

1) **Selection.** On a fully expanded node (all the children nodes have been visited), MCTS selects to explore the branch with the highest Upper Confidence Bound (UCB),

$$\arg\max_{a}\left(\frac{w(f(s,a))}{n(f(s,a))} + C\sqrt{\frac{\log(n(s))}{n(f(s,a))}}\right), \tag{1}$$

   where $f(s,a)$ is the child node of state s after action a, $w(\cdot)$ and $n(\cdot)$ are respectively cumulative rewards and the number of visits to a state.
2) **Expansion.** On a node that is not fully expanded, MCTS selects an action that has not been attempted yet.
3) **Simulation.** Given a node and a selected action, MCTS simulates a sequence of actions and receives a reward.
4) **Back-Propagation.** MCTS passes the terminal reward to ancestor nodes to update their cumulative expected rewards, which indicate the quality of the branch.

At each iteration, MCTS starts from the root node. When all the child nodes of the current node are visited, MCTS selects a child node with the UCB formula. When some child nodes of the current node are unvisited, MCTS expands by randomly selecting a new action and performing a simulation to reach a new child node. The new node returns a reward, which is back-propagated to all the ancestor nodes.

IV. Method

In a nutshell, LGMCTS starts by calling LLM to parse the language description L to a list of spatial distributions $F = \{f_i : p(o_i) \sim f_i | o_i \in O_R\}$. Then it uses an MCTS-based procedure to find a physically feasible action sequence A to rearrange the scene according to distributions F.

A. Language Parsing & Object Selection

During the language parsing stage, we parse L into $F = \{f_i : p(o_i) \sim f_i | o_i \in O_R\}$. Similar to previous methods [9], we conduct an automated prompt engineering to guide the LLM to perform the parsing.

The listing below showcases how prompt engineering is implemented. Essentially, the language model translates user requirements into structured goal configurations and constraints that guide task execution.

Consider the example depicted in Fig. 2. First comes the **system prompt** providing guidelines for the LLM to follow when interpreting user queries. Then, we need to provide an **object description** such as semantic labels, colors, and

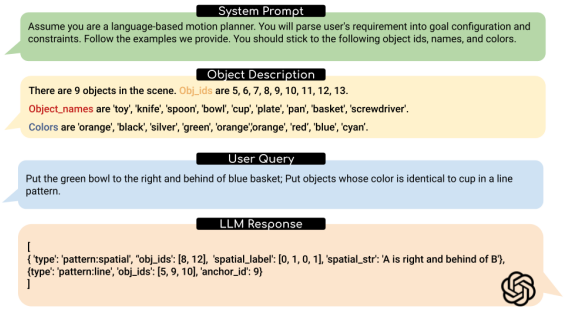

Fig. 2: An example of Language Parsing. We are using GPT-4 [4] in this work.

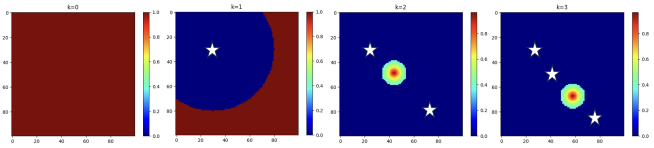

Fig. 3: Visualization of (x,y) prior for 'line' pattern. From left to right: $K=0$, $K=1$, $K=2$, $K=3$, where $K = |O_R^{sampled}|$, the number of sampled object poses. White star marks are sampled poses. When $K=0$, the pose can be sampled anywhere. When $K=1$, it needed to sampled outside a circle region. After that, all poses will be sampled along the line defined by the first two poses.

IDs for the objects in the scene. In practice, we use the Recognize Anything Model (RAM) [38], [39] for producing the semantic labels, and a color detector to determine the colors of the objects in the scene. A unique ID is assigned to each object. Finally, we provide a **user query** describing the rearrangement goal. Finally a structured answer is returned from the LLM. LLM's answers suggest how many patterns there are and which objects should be selected to form the pattern.

B. Parametric Geometric Prior

As mentioned in Sec. III-A, if we know the goal position distribution for each object, the semantic rearrangement problem can be converted to a sequential sampling problem. There are multiple details we need to pay attention to in this process. (1) The position distribution is not fixed but varies with the progress of sampling. For example, if we say object A, B, and C need to be put into a line, and we sample in the order of A, B, and C, then the distributions of A and B are actually unbounded, and C must be placed on the line defined by the positions of A and B. (2) The position distribution should be collision-free. (3) One can build a database for many different spatial distributions, but we need to have a flexible mechanism to associate LLM inferred distribution types with items in the database.

We show in the following how to compute distribution function $f_i(p_i|P_S, L)$ for each object o_i in the set O_R during the sequential pose sampling process. Here, p_i denotes the pose (x, y, θ) of an object o_i, P_S represents the current set of poses for all objects, and L serves as the language guidance for sampling poses. The set of poses P_S is updated

incrementally and reflects the progress of the sequential sampling process.

We compute $f_i(p_i|P_S,L)$ as an element-wise product of two components, a pattern prior function $f_{\text{prior}}(P_R,L)$ and a boolean function $f_{\text{free}}(p_i|P_S)$ of the workspace,

$$f_i(p_i|P_S,L) = f_{\text{prior}}(p_i|P_R,L) \times f_{\text{free}}(p_i|P_S) \tag{2}$$

In this equation, P_R stands for the set of poses for objects o_j belonging to O_R, defined as the objects concerned by instruction L. Function $f_{\text{free}}(p_i|P_S)$ is set to 1 for all poses p_i of object o_i that do not lead to a collision with the remaining objects with poses given by the set P_S, and to 0 otherwise. f_{free} is determined by running a 2D collision simulation using point cloud observations for each object o_i in the scene.

Our primary focus is to determine $f_{\text{prior}}(p_i|P_R,L)$. To this end, we employ an approach akin to the one described in [8]. We maintain a database comprising a collection of predefined prior functions. Each of these functions is linked with one or more Sentence-BERT embeddings, acting as keys. The function corresponding to the best matching key is selected, as follows,

$$f_{\text{prior}}(p_i|P_R,L) = f^k_{\text{prior}}(p_i|P_R) \text{ with } k = \underset{k\in\text{database}}{\text{argmax}}\,(\Theta_k \cdot \Theta(L))$$

Here, Θ_k is the k^{th} key in the database, and $\Theta(L)$ is the Sentence-BERT embedding generated from the language instruction L. In summary, the most suitable prior function from the database is selected based on the Sentence-BERT embedding of the given language instruction L.

We now delve into the definition of $f^k_{\text{prior}}(p_i|P_R))$ in the context of our work. We use a unique model for representing various distributions by employing parametric curves. A parametric curve can be expressed as $(x,y) = \gamma(t,\kappa)$, where t ranges from 0 to 1 and κ is a set of curve-defining parameters. In our work, κ is modeled as a function of two 2D positions, denoted as $\kappa(p_0,p_1)$. Therefore, for each pattern, we define two functions: γ and κ.

Given that pattern prior f_{prior} is used inside a sequential sampling process (check Section IV-C for more details), the prior distribution needs to be iteratively updated to capture the history of sampling. Consequently, we further categorize O_R (the subset of objects involved in given instruction L) based on whether the objects have been sampled in the current branch of the MCTS-Planner. Subsets $O_R^{sampled}$ and $O_R^{unsampled}$ denote the sampled and non-sampled objects, respectively. Thus, $O_R = O_R^{sampled} \cup O_R^{unsampled}$.

The probability $f^k_{\text{prior}}(p_i|P_{R_i}))$ of sampling a pose $p_i = (x_i,y_i,\theta_i)$ for next object $o_i \in O_{R_i}^{unsampled}$ is given as follows.

- If $O_R^{sampled} = \emptyset$ then $(x_i,y_i,\theta_i) \sim U$, suggesting that the first object can be placed arbitrarily.
- If $|O_R^{sampled}| = 1$ then $\sqrt{(x_i-x_0)^2+(y_i-y_0)^2} \leq \delta$ and $(x_i,y_i,\theta_i) \sim U$, imposing that the second object must be sampled uniformly at a position that is distanced from the first by at most δ.
- If $|O_R^{sampled}| > 1$ then $(x_i,y_i) = \gamma\big(\frac{|O_R^{sampled}|}{|O_R|}, \kappa(p_0,p_1)\big) + \varepsilon$ where $\varepsilon \sim G(0,\sigma)$, here G represents an Gaussian distribution of variance σ. $\theta = atan2(1,\gamma'(\frac{K}{N}))$, we use the angle of gradient represented as the rotation angle of the object.

Our parametric geometrical representation enables us to model any geometric shapes that can be written into the format of a parametric curve. In our current implementation, we defined shapes such as "line," "circle," "rectangle," "tower," "spatial:left," "spatial:right" and so on. Due to space constraints, we refrain from elaborating on the definitions of γ and κ for all these predefined patterns. Fig. 3 illustrates an example of a parametric geometric prior. Noticeably, we divide patterns into 'ordered' and 'unordered' based on if the pattern requires an execution sequence.

Note that in our work, language instruction L is typically composed of multiple instructions that deal with different subsets of objects. Specifically, L can be interpreted as a list $\{L_k\}$ of simple instructions L_k. For example, instruction L_1 refers to placing objects A, B and C in a line, and instruction L_2 refers to placing object A on the left of B. Each language instruction $L_k \in L$ is associated with a subset of objects $O_{R_i} \subset O_R$ that need to be manipulated to form the pattern described in L_k. In our example, $O_{R_1} = \{A,B,C\}$ and $O_{R_2} = \{A,B\}$. In the sequential sampling process described above in this section, we only presented the case of a single language instruction for simplicity, but the same process is used for sampling poses given by a complex instruction.

C. Monte-Carlo Tree Search (MCTS) for TAMP

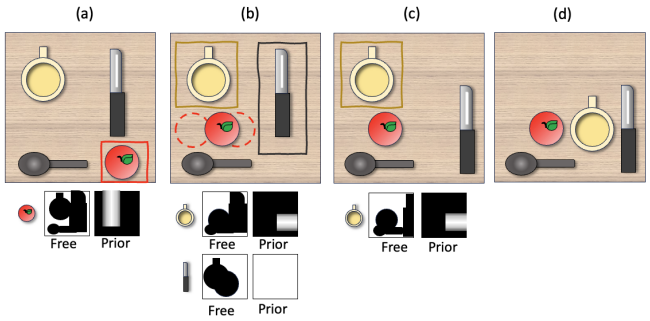

Fig. 4: A minimal example illustrates our MCTS-Planner's aim to arrange a table. The language description provided is: "Can you please put the apple on the top-side of the spoon? And I also want the cup to be to the right of the apple." The top row displays the current scene arrangement, while the bottom row shows the f_{prior} and f_{free} for the object being manipulated. In the probability figure, black represents 0, and white represents 1.

As previously mentioned, our rearrangement problem can be reformulated into a sequential task. In each step, we sample a pose p_i for each object o_i according to a pose distribution $f_{d_i}(p_i)$ selected by the LLM and computed by a parametric geometric prior. Once we complete $|D|$ steps, all objects will have been placed in their desired locations. Here, D represents the list of pose distributions that we need to consider during this task. However, this task cannot be executed in a naive sequential order, as the rearrangements made in previous steps may obstruct subsequent sampling.

Therefore, we propose a task and motion planner based on the Monte Carlo Tree Search (MCTS) algorithm to simultaneously arrange the task and address object relocation issues. The objective of the MCTS is to fulfill all object position requirements defined by $|D|$. In the MCTS-Planner, we maintain a tree where each node s in the tree comprises the current objects' poses, $\{p_1, p_2, ..., p_N\}$, and the remaining object position requirements D_s, where $D_s \subseteq D$.

Intuitively, at node s, we sample actions to progress towards reducing D_s. For each pose distribution $d_i \in D_s$, we attempt k actions for d_i. Each action is represented as (d_i, j), denoting the $j^{th}(j \leq k)$ attempt to progress in sampling d_i. If a collision-free goal pose for o_i within d_i is found, the action involves moving the target object o_i to a pose inside d_i. If not, the action will involve relocating an obstacle. This step is the *Simulation* stage in MCTS. Details are defined in Algo. 1. It is important to note that in ordered patterns, we will not consider sampling an object o_i into d_i if not all of d_i's prerequisites have been met. For instance, as depicted in Fig. 2, there is a pattern "A is on the right and behind B". A will only be rearranged after B has been rearranged.

Formally, MCTS-Planner is structured into four phases: selection, expansion, simulation, and back-propagation, as detailed in Section III-B. The selection, expansion, and back-propagation stages follow the methodology outlined by [40], as referenced in Section III-B. The reward for transitioning to a new state s is quantified by the reduction in the number of pose requirements, that is, $|D| - |D_s|$. This reward structure mirrors the approach in [40], aiming to steer the search towards branches that efficiently move objects to their goal poses. The simulation phase is elaborated in Algorithm 1. This phase begins with the MCTS state s and an attempted action (d_i, j). If the targeted object o_i is not graspable, a graspable obstacle is randomly selected from those above it, and a collision-free pose within the workspace is uniformly sampled for relocation (Lines 2-6). If o_i is graspable, an attempt is made to sample it within d_i (Line 8). Should the sampled position be deemed undesirable, i.e., $f_{d_i}(p) < \varepsilon$, an obstacle o within d_i is identified, and a collision-free pose in the workspace is uniformly sampled for its placement (Lines 10-14). The obstacle relocation strategy increases the robustness of our rearrangement planner, especially when the environment is cluttered. Although MCTS operates as an anytime search algorithm, our MCTS-Planner implementation returns the first solution it finds. Furthermore, we have proved that the MCTS-Planner is probabilistically complete within our specified framework in Proposition. 4.1.

Figure 4 presents a practical example of the MCTS-Planner at work. Owing to space constraints, we will only explore three simulation steps along the branch that yield a feasible solution. In this scenario, the user requests, "Can you please put the apple on the top-side of the spoon? And I also want the cup to be at the right of the apple." In response, the LLM generates the requirement distribution list $D = \{d_1, d_2\}$, where d_1 is for positioning the apple on the top-side of the spoon, and d_2 is for placing the cup to the right of the apple. Figure 4(a) illustrates the initial arrangement of objects. Given the dependency of d_2 on the apple's position, k actions will be sampled for d_1, but none for d_2 in this initial setup. Figure 4(b) depicts the outcome of an action (d_1, j), where the apple is relocated according to d_1. The dashed-line circles represent the other $k-1$ actions originating from the root node. After sampling d_1, we are left with $D_s = \{d_2\}$ as shown in Figure 4(b), and attempt is made to position the cup to the right of the apple. However, due to collisions, $f_{d_2}(\cdot) \leq \varepsilon$ throughout the workspace, leading to actions (d_2, j) that involve relocating either the apple or the knife. Figure 4(c) demonstrates the knife's relocation, maintaining $D_s = \{d_2\}$. Ultimately, Figure 4(d) showcases the final planning result.

Algorithm 1: Simulation

Input : s: An MCTS state,
(d_i, j): The action for this simulation.
Output: (o, p): a rearrangement action.

1 $o_i \leftarrow$ The sampling object in d_i;
2 **if** o_i *not graspable* **then**
3 $o \leftarrow$ Randomly choose a graspable object on top of o_i;
4 $p \leftarrow$uniformSampling(o, s);
5 **if** p **then return** (o,p);
6 **else return None**;
7 **else**
8 $p \leftarrow$sampling(o_i, d_i, s);
9 **if** $f_{d_i}(p) \geq \varepsilon$ **then return** (o_i,p);
10 **else**
11 $o \leftarrow$ Randomly choose an obstacle in d_i;
12 $p \leftarrow$uniformSampling(o, s);
13 **if** p **then return** (o,p);
14 **else return None**;

Proposition 4.1: MCTS-Planner is probabilistic complete.

Proof: For a semantic rearrangement task and the distribution list D, assume that there is a feasible action sequence A^* moving objects to a final arrangement A_f, such that $f_{d_i}(A_f[o_i]) \geq \varepsilon \ \forall o_i \in O_R$. Denote p as the probability that MCTS can find an action sequence satisfying the goal state criteria. We prove that as the number of samples k increases, p approaches 1.

First, we prove that there is an action sequence A_0^* whose actions can all be generated by Algorithm 1. Note that in Algorithm 1, a MCTS action (d_i, j) satisfies two rules: **R1**: If o_i is not graspable, we move away obstacles of o_i(Line 3); **R2**: If o_i is graspable, we move o_i into d_i or remove some obstacle in d_i for o_i (Line 11). We construct A_0^* by reordering and deleting actions in A^* as follows: 1) If an action satisfies **R1** and **R2**, we add the action to A_0^*. Otherwise, we delay the addition until it satisfies the rules; 2) If the action still cannot satisfy the requirements before we examine the next action in A^* for the same object, we delete the action. In this way, all the actions in A_0^* can be generated by Algorithm 1, and the final arrangement is also A_f. Let the rearrangement action (o, p) that A_0^* chooses at state s be $(A_0^*[s].o, A_0^*[s].p)$.

Let $r = \dfrac{min(C_1, C_2)}{|A_0^*|}$, where C_1 is the minimum distance between objects and their nearest obstacles in A_0^*, C_2 is the minimum distance between each pose $A_f[o_i]$ and their nearest

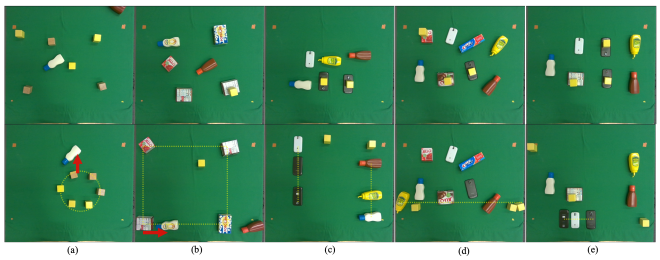

Fig. 5: Real world demonstration with a UR5 robot. The language instructions for the five scenes are: (a). "Move all blocks into a circle; while put the white bottle behind one block;" (b). "Put all boxes into a rectangle; and move the white bottle to the right of one box;" (c) "Move bottles into a line; and formulate all phones into another line;" (d) "Formulate all yellow objects into a line;" (e) "Set all phones into a line;". The top row images show the initial scenes, and the bottom ones show the results of using LGMCTS on the UR5. Dotted lines imply a shape pattern and red arrows indicate a spatial pattern (left, right, front, back). These real robot experiments show that LGMCTS can parse complex language instructions and also deal with infeasible start configurations as well as pattern composition.

Method	Line (4295)	Circle (3416)	Tower (1335)	Dinner (2440)
LFSP* [9], [15]	41.16%	51.75%	88.80%	27.05%
Structformer [3]	47.24%	62.64%	99.10%	28.36%
StructDiffusion [2]	61.49%	81.41%	98.95%	69.38%
LGMCTS (Ours)	**95.99%**	**95.25%**	**100%**	**100%**

TABLE I: Efficacy of LFSP, Structformer, StructDiffusion, and LGMCTS across diverse rearrangement tasks (task counts indicated) from the Structformer dataset. *Due to budget constraints, the LLM baseline LFSP are evaluated on 1150 (10%) randomly selected scenes of the Structformer dataset.

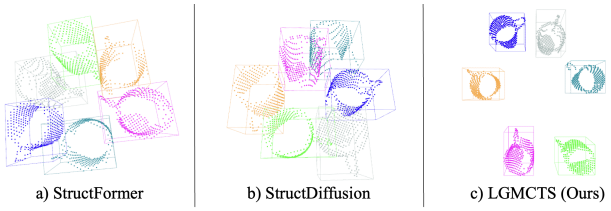

Fig. 6: Compared to Structformer and StructDiffusion, LGMCTS ensures a collision-free goal arrangement in all experiments.

position p, s.t. $d_i(p) < \varepsilon$. As k increases, the probability that $A_0^*[s].o$ is moved to the $r-$neighborhood of $A_0^*[s].p$ at state s approaches 1. Then, given a tolerance of pose offset r, the probability that all the intermediate states of A_0^* are in the MCTS tree approaches 1. When A_f is in the MCTS tree, the MCTS-Planner can find a solution after enough iterations. ■

V. EXPERIMENTS

We present a comprehensive evaluation of LGMCTS on: (1) its capability to produce collision-free and semantically correct goal poses, (2) the advantages of concurrently addressing pose generation and action planning, and (3) its performance in a real robotic system.

A. Baselines

We compare our approach with the following baselines.

Structformer [3]. It is a multi-modal transformer specifically designed for language-guided rearrangement tasks.

StructDiffusion [2]. It employs a diffusion model combined with a learning-based collision checker for pattern pose generation.

LLMs as Few-Shot Planners [9], [15]. We integrate *Code as Policies* and *Progprompt* into our evaluation pipeline, where the former generates policy code and the latter Pythonic code. As we cannot directly use the generated code, to streamline the input to our TAMP planner, our setup modifies the output as a sequence of actions (object IDs and their target poses). Initially, LLM processes complete scene details—including object names, IDs, textures, initial poses, and region boundaries for the rearrangement. We then instruct the LLM to take the natural language command and produce the optimal action sequence that contains an ordered list of object IDs and goal poses of the considered objects for rearrangement. However, for evaluating the Structformer dataset, we consider the structured goal specification pre-available in the dataset as the input to the LLM as it can infer the action plan from this intermediate representation. This approach avoids redundancy, as generating a natural language command would just restate the same specification for the goal rearrangement. In evaluating both datasets, we provide a few scenes as examples where the output format is clearly defined to the LLM with ground-truth optimal sequence. This baseline is named LLMs as Few-Shot Planners (LFSP).

Pose+MCTS. The Pose+MCTS (PMCTS) approach assumes that a collision-free and semantically aligned goal pose is provided. However, direct execution of this pose might be hindered if the target space is already occupied. To address this, we utilize MCTS to search for a viable plan to place objects in their predetermined goal poses. MCTS is only used as a motion planner. This method follows a two-step approach of using goal poses independently of task planning.

B. Structformer Dataset

We use the test set from the Structformer dataset to evaluate the goal pose generation ability. This dataset is composed of approximately $11,500$ rearrangement tasks, categorized into four patterns: line, circle, tower, and dinner. A rearrangement plan is considered successful if it adheres to language constraints and is collision-free, except in the "tower" task where collisions are inevitable. The "dinner" task is approached as a composition of patterns, involving the arrangement of items like plates, bowls, and utensils into a "tower" for plates and bowls, with other items lined up beside it. In both Structformer and StructDiffusion's experimental setup, object selection for rearrangement is based on the object's shape and size. Our evaluation setup does not involve object selection based on shape and size. Hence, to adapt them to our evaluation setup, we provide those two baselines with ground-truth object selection. Since the tasks already specify which objects to rearrange for the single pattern rearrangement, based on language instructions, we did not use the LLM parser in LGMCTS for this dataset.

As shown in TABLE I, LGMCTS demonstrated superior performance in all four rearrangement task categories, achieving remarkable success rates as follows: 95.99% for "line", 95.25% for "circle", and 100% for both "tower" and "dinner". LSFP performs the least among the baselines due to the inability of LLMs to produce goal patterns with high geometric fidelity. While StructDiffusion showed improvement over Structformer, it did not match LGMCTS's effectiveness. For a visual comparison, Fig. 6 illustrates LGMCTS's success in a "circle" task scene, highlighting its more actionable goal poses that contribute to higher rearrangement success rates. Conversely, Structformer and StructDiffusion tend to generate goal arrangements with a higher collision risk, leading to lower success rates. The evaluation of this dataset strictly assesses the accuracy of collision-free geometric patterns in goal poses, disregarding the executability of plans—a notable limitation of the dataset. This limitation is addressed through our proposed ELGR-Benchmark (refer to Section V-C). PMCTS method is introduced through our benchmark to solely verify the executability of plans. As PMCTS deals with motion planning using collision-free ground-truth poses, it was not included in further comparisons on this dataset due to its inherent access to goal poses.

C. ELGR-Benchmark

Existing datasets for semantic object rearrangement, such as Structformer, are limited in that they typically feature only one pattern per scene and do not include crowded scenarios. They also fail to address the challenge of feasibility, particularly when starting configurations are infeasible like one object being placed under another. To bridge these gaps, we introduce ELGR-Bench (**E**xecutable **L**anguage-**G**uided **R**earrangement **Bench**mark), which incorporates scenarios with infeasible starting configurations, including tasks that require unstacking and appropriate placement of unstacked objects before the actual rearrangement. Importantly, this new benchmark presents a novel task termed the "multi-pattern task", which requires multiple pattern goals to be satisfied during the rearrangement process. In this benchmark, we are considering common shapes such as "line", "circle", "rectangle" and "spatial" (left/right, front/behind, left/right + front/behind). For each scene, we randomly compose two of the aforementioned patterns and create the multi-pattern task. Success is measured based on the executability of the generated plan and its adherence to semantic requirements. ELGR-Bench builds upon the VIMA-Benchmark [41].

In our benchmark, we compared LGMCTS against two baselines: LFSP and PMCTS, excluding Structformer and StructDiffusion as they cannot handle composite geometric patterns. LFSP, leveraging an LLM, plans goal poses and action sequences simultaneously, while PMCTS, follows a two-step method using a given goal pose and then using MCTS for action planning. LGMCTS uniquely combines goal generation with action planning, aiming for more executable outcomes. As shown in TABLE II, LFSP demonstrates a 100% planning success rate through LLM's capability to generate action plans based on natural language commands and scene context. Nonetheless, over 50% of these plans are inexecutable, as indicated by the SR_{ep} scores. LGMCTS, however, manages a 90% success rate in generating action plans, with about 80% being executable. This performance not only underlines the limitations of LLMs in direct TAMP solving but also showcases LGMCTS's advantage over the two-step PMCTS approach, even when PMCTS is provided with accurate and feasible goal poses.

Method	SR_p	SR_{ep}
LFSP [9], [15]	**100%**	45.2%
Structformer [3]	n.a.	n.a.
StructDiffusion [2]	n.a.	n.a.
PMCTS	82.9%	74.1%
LGMCTS (Ours)	90.9%	**79.2%**

TABLE II: SR_{ep}, the executable plan success rate, reflects both planning success and the success of executing these plans, indicating if the final positions of objects meet the criteria set by language-based constraints. SR_p, a part of SR_{ep}, only tracks planning success, with PMCTS and LGMCTS capped at 10,000 planning steps. If planning with MCTS exceeds the limit, often due to dense object placement in the scene, the motion planning is considered a failure.

D. Real Robot Experiments

We qualitatively evaluated our system using a UR5e robot equipped with a D455 depth camera. The setup of the robot is shown in Fig.1. We employed the Recognize-Anything-Model (RAM) [38], [39] and an HSV-based color detector to detect object semantics and colors. Selected queries and their corresponding execution outcomes are presented in Fig.5. We considered five different language instructions involving various objects and initial configurations. For example, Fig.5(b) illustrates the experiment with "Put all boxes into a rectangle, and move the white bottle to the right of one box." This experiment involves pattern composition, requiring simultaneous consideration of "line" and "to the right of" constraint. Additionally, this scene presented an infeasible initial configuration, necessitating the removal of the yellow block before moving the gelatin box. Each experiment presented distinct challenges; for more details, refer to Fig. 5. These real-world robot experiments underscore the capabilities of LGMCTS in complex real-world settings.

VI. CONCLUSION

We introduced LGMCTS, a new framework for tabletop, semantic object rearrangement tasks. LGMCTS stands out by accepting free-form natural language input, accommodating multiple pattern requirements, and jointly solving goal pose generation and action planning. However, its main drawback is the extended execution time for complex scenes. Improving the tree search efficiency is a key research direction. Currently tailored for tabletop pick-place setups, future work should explore LGMCTS's adaptability to more complex rearrangement contexts.

REFERENCES

[1] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *5th Annual Conference on Robot Learning*, 2021.

[2] W. Liu, T. Hermans, S. Chernova, and C. Paxton, "Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects," in *Workshop on Language and Robotics at CoRL 2022*, 2022.

[3] W. Liu, C. Paxton, T. Hermans, and D. Fox, "Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6322–6329.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

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

[7] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[8] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on Robot Learning*. PMLR, 2023, pp. 287–318.

[9] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[10] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, "Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change)," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. [Online]. Available: https://openreview.net/forum?id=wUU-7XTL5XO

[11] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," *arXiv preprint arXiv:2303.06247*, 2023.

[12] X. Zhang, Y. Zhu, Y. Ding, Y. Zhu, P. Stone, and S. Zhang, "Visually grounded task and motion planning for mobile manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1925–1931.

[13] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," *arXiv preprint arXiv:2306.06531*, 2023.

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

[15] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.

[16] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009.

[17] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[18] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," in *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, 2023.

[19] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv preprint arXiv:2305.05658*, 2023.

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

[21] Z. Wu, B. Ai, and D. Hsu, "Integrating common sense and planning with large language models for room tidying," in *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.

[22] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling, "Pddl planning with pretrained large language models," in *NeurIPS 2022 foundation models for decision making workshop*, 2022.

[23] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[24] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The Eleventh International Conference on Learning Representations*, 2022.

[25] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating natural language to planning goals with large-language models," *arXiv preprint arXiv:2302.05128*, 2023.

[26] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar *et al.*, "Pre-trained language models for interactive decision-making," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 199–31 212, 2022.

[27] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," in *7th Annual Conference on Robot Learning*, 2023.

[28] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, "Leveraging pre-trained large language models to construct and utilize world models for model-based task planning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[29] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[30] T. Birr, C. Pohl, A. Younes, and T. Asfour, "Autogpt+ p: Affordance-based task planning with large language models," *arXiv preprint arXiv:2402.10778*, 2024.

[31] A. Zeng, M. Attarian, K. M. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. S. Ryoo, V. Sindhwani, J. Lee *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," in *The Eleventh International Conference on Learning Representations*, 2022.

[32] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[33] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[34] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[35] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *Transactions on Machine Learning Research*, 2022.

[36] A. Madaan, S. Zhou, U. Alon, Y. Yang, and G. Neubig, "Language models of code are few-shot commonsense learners," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 1384–1403.

[37] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.

[38] X. Huang, Y. Zhang, J. Ma, W. Tian, R. Feng, Y. Zhang, Y. Li, Y. Guo, and L. Zhang, "Tag2text: Guiding vision-language model via image tagging," *arXiv preprint arXiv:2303.05657*, 2023.

[39] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu *et al.*, "Recognize anything: A strong image tagging model," *arXiv preprint arXiv:2306.03514*, 2023.

[40] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3715–3722, 2020.

[41] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.