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 introduce a novel approach to the executable semantic object rearrangement problem. In this challenge, a robot seeks to create an actionable plan that rearranges objects within a scene according to a pattern dictated by a natural language description. Unlike existing methods such as StructFormer and StructDiffusion, which tackle the issue in two steps by first generating poses and then leveraging a task planner for action plan formulation, our method concurrently addresses pose generation and action planning. We achieve this integration using a Language-Guided Monte-Carlo Tree Search (LGMCTS). Quantitative evaluations are provided on two simulation datasets, and complemented by qualitative tests with a real robot. Our code and supplementary materials are accessible at https://github.com/changhaonan/LG-MCTS.**

I. INTRODUCTION

In daily life, tasks like "Set up the kitchen" require arranging objects according to language cues, an intuitive process for humans but a complex challenge for robots. The semantic rearrangement problem aims to enable robots to reconstruct scenes based on linguistic descriptions. This necessitates the seamless integration of scene understanding, linguistic reasoning, and action planning, involving multiple disciplines such as robotics and natural language processing.

Consider the task: "Set the dinnerware for dinner and place a candle in front of a plate." A robot must identify the items constituting 'dinnerware' and their correct arrangement, while also handling real-world constraints like initial object coverings and spatial obstacles. The command introduces two key constraints: 1) 'dinnerware' must be arranged appropriately for dinner, and 2) a candle must be placed in front of a plate. This example highlights the problem's complexity.

One approach employs a multi-modality transformer [1]–[3] to learn the mapping between language and object poses through simulated arrangements and rule-based language descriptions. However, this method has limitations. It assumes that language descriptions map to precise ground-truth poses, which is often unrealistic. It is also less adaptable to free-form linguistic inputs, as it performs best with descriptions similar to its training data.

Recent research employs diffusion models to capture how language tokens map to spatial distributions. DALL-E-Bot [4] uses text-to-image to generate target images and derive poses, while StructDiffusion [5] employs a diffusion model conditioned on language and point-cloud embeddings.

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.

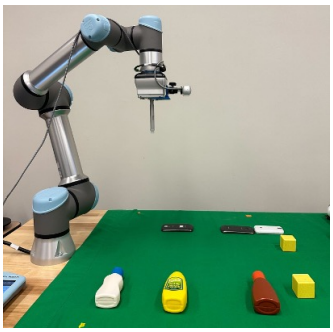

Fig. 1: Robot Setup. We use a UR5e robot equipped with a RealSense D455 camera.

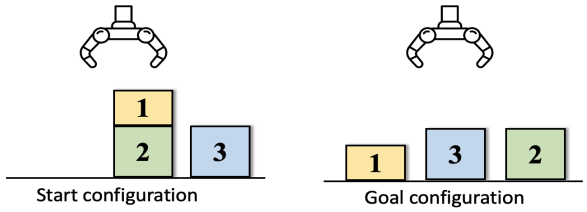

Fig. 2: Illustration of infeasible start and goal configuration. In this scene, block 2 is in an infeasible start configuration. And the goals of blocks 3 and 2 are infeasible without an excessive number of intermediate steps.

Both approaches show promise but have limitations: DALL-E-Bot can be unstable and distracted by extraneous objects, whereas StructDiffusion is constrained to known training patterns and lacks zero-shot adaptability.

Moreover, recent developments emphasize direct robot control through Large Language Models (LLMs) [6]–[8] and prompt techniques such as Chain of Thoughts (COT) [9]. The 'Code-as-policies' [10] approach shows remarkable zero-shot capabilities and could theoretically address the semantic rearrangement problem, leveraging LLMs' strength in pattern comprehension and object selection.

Current methods such as StructFormer and Code-as-policies often fail to produce collision-free arrangements, whereas StructDiffusion incorporates a learning-based collision checker. However, collision-free does not mean executable, as shown in Fig. 2. For example, block 3 obstructs block 2's target position, complicating block 1's motion. This highlights the challenge of distinguishing between collision-free and executable goals, as some may be the former but not the latter, or may require excessive steps to execute.

This study presents the **L**anguage-**G**uided **M**onte-**C**arlo **T**ree **S**earch (LGMCTS) technique, designed specifically

for executable semantic object rearrangement. LGMCTS leverages LLMs to interpret free-form language and consider object placements as probability distributions, not exact points. We frame the challenge as a sequential sampling problem, where each object's pose is drawn from a distribution influenced by language and the current scene state. The approach incorporates distracting objects, enabling plans that meet language criteria while also being executable.

The primary contributions of this study are: 1) Introducing a novel approach that concurrently addresses semantic rearrangement goal generation and action planning 2) Presenting a unique method that facilitates zero-shot multi-pattern composition for semantic object rearrangement 3) Establishing a new benchmark tailored for executable semantic object rearrangement.

II. RELATED WORKS

A. 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 [2] advanced the field using a transformer model, simulating rearrangements with hand-crafted rules and connecting language tokens to object poses. Leveraging StructFormer's dataset, StructDiffusion [5] introduced a pose diffusion model to predict poses from language, enhancing performance. Nonetheless, a common shortcoming amongst these methodologies is their limitation to a single structure or pattern that they have been trained on, making composite patterns a persistent challenge. Meanwhile, the rearrangement goals generated by these methods might be inexecutable.

B. LLM-driven Robot Control

Recent advancements in large language models (LLMs) [6]–[8] have showcased stellar performance across a broad spectrum of tasks. This has led to a growing interest in LLM-driven robotics. SayCan [11] was among the first to integrate LLMs into robotic task planning, which resulted in impressive context comprehension and behavioral decision-making. Subsequently, Code-as-policies [10] merges LLM code generation with API planning, demonstrating remarkable zero-shot capabilities across various robotic tasks. Nonetheless, LLM-driven robotics faces several challenges. LLMs, while semantically adept, seem to lack a true physical scene understanding, leading to plans that, although meaningful, can be unfeasible.

C. Rearrangement Planning

In rearrangement problems, determining the sequence of tasks presents a significant challenge. Several studies have tackled this by encoding collisions between initial and target arrangements into a graph [12]–[14]. These representations then transform the problem into established graph problems. Additionally, recent advances [15], [16] have adopted the Monte Carlo Tree Search (MCTS) for long-horizon planning. A common attribute among most prehensile planners is the need for specific goal states [17]. However, dictating a goal state in semantic rearrangement can restrict the solution space, leading to potential planning failures. To address this, our proposed LGMCTS planner calls for goal state distributions from a language model, which act as constraints for individual goal poses.

III. PRELIMINARIES

A. Problem Formulation

The task of semantic rearrangement can be succinctly defined as follows. Given a scene with objects represented by $O_S = \{o_1, o_2, \ldots, o_N\}$ and a command L, where L is a pure natural language command that implies a desired distribution list $D = \{d_i : p(o_i) \sim f_{d_i} | o_i \in O_R\}$, $p(o_i)$ refers to the position of object o_i. Here, $O_R \subset O_S$ denotes the objects designated for rearrangement, and d_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 goal to achieve a goal arrangement aligning L, i.e. $\prod_{o_i \in O_R} f_{d_i}(p_i) > 0$ and minimizing the action steps H. Noticeably, A not only includes movement of objects $o \in O_R$, but also distracting objects, denoted as O_d, with $O_d \subset O_S$.

B. Monte Carlo Tree Search

A typical MCTS algorithm iteratively performs the following four operations:

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

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

   where $f(s,a)$ is the child node of state s after action a, $w(\cdot)$ and $n(\cdot)$ are 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 the selected action, MCTS simulates and gets rewards.
4) **Back-Propagation.** MCTS passes the acquired reward to ancestor nodes to update the quality evaluation of the branch.

In 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 doing a simulation to reach a new child node. The new node returns a reward, which is be back-propagated to all the ancestor nodes.

IV. Method

A. Notation Vocabulary

TABLE I: Notation and Definitions

Symbol	Meaning	Definition
L	Language input	
D	Parsed distribution set	
o_i	Individual object	
d_i	Pose distribution for o_i	
O_R	Objects to be rearranged	
L_i	Language guidance for sampling o_i	
O_{R_i}	Subset of objects associated with L_i	
$w(f(s,a))$	Total reward accumulated for action a in state s	Eq. (1)
$n(f(s,a))$	Number of times action a in state s is taken	Eq. (1)
C	Exploration constant	Eq. (1)
$\log(n(s))$	Logarithm of the number of times state s has been visited	Eq. (1)
$f_{d_i}(p_i\|P_S,L_i)$	Conditional distribution for o_i	Eq. (2)
p_i	Pose of object o_i	
P_S	Set of current poses for all objects	
f_{prior}	Pattern Prior function	Eq. (3)
f_{free}	Free space function	
F_k	k^{th} Key in database	Eq. (4)
F_{L^i}	Sentence-BERT embedding of L^i	
γ	Parametric curve function	
κ	Parameters related with a parametric curve	
P_{R^i}	Poses for all objects in O_{R^i}	
P_{R_i}	Poses of sampled objects in O_{R^i}	
N	Total number of objects associated with a pattern	
K	Number of sampled objects	
S_A	MCTS action set	Algorithm 1

B. Language Parsing & Object Selection

Since the input L is a pure natural language command, we must first parse L into $D = \{o_i : d_i | o_i \in O_R\}$. Here, we utilize a large language model (LLM) to parse the language into some structured representation. Similar to previous methods [10], we conduct a prompt engineering to guide the LLM to parse L into D. 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.

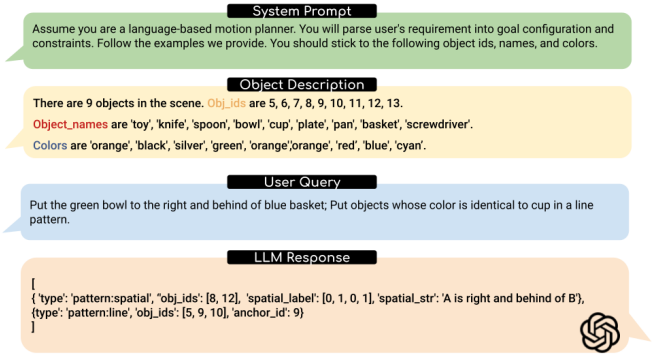

Fig. 3: An example of Language Parsing. We are using GPT-4 [6] in this work.

Consider the example depicted in Fig. 3. Here, the system prompt provides guidelines for the LLM to follow when interpreting user queries. For producing the descriptor labels such as name, color, and ID for the objects in the scene, we use the Recognize Anything Model (RAM) [18], [19] for producing the semantic labels, and a color detector for associating colors to the objects in the scene.

Following the relay of user instructions to the LLM, our requirements for the model can be summarized in two main steps: 1) Identifying an ensemble of pertinent objects (O_R) and 2) Correlating them with their respective sampling pattern descriptions and relevant objects to form these patterns. More precisely, for each object $o_i \in O_R$, there is a corresponding sampling pattern description L_i and a subset of objects $O_{R_i} \subset O_R$ associated with L_i.

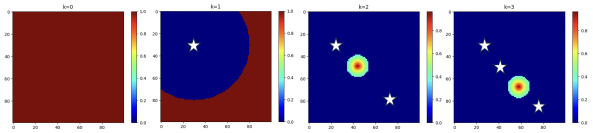

Fig. 4: Visualization of (x,y) prior for 'line' pattern. From left to right: $K=0$, $K=1$, $K=2$, $K=3$. White star marks are sampled poses. When $K=0$, the pose can be sampled anywhere. When $K=1$, it needed to sampled outside a circle region. After that all poses will be sampled along the line defined by the first two poses.

Through extensive testing, we observed that instead of directly instructing the LLM to prescribe a sampling pattern for individual objects, it is more robust to guide it to discern the number of patterns present in the instruction and to identify the respective objects within these patterns (Fig. 3).

C. Distribution Generation

In this section, our aim is to construct a conditional distribution $f_{d_i}(p_i|P_S,L_i)$ for each object o_i in the set O_R. 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_i serves as the language guidance for sampling poses for o_i.

To facilitate this, we compute $f_{d_i}(p_i|P_S,L_i)$ as an element-wise product of two components: a pattern prior function $f_{\text{prior}}(P_{R^i},L_i)$ and a boolean function $f_{\text{free}}(P_S)$ of the workspace,

$$f_{d_i}(p_i|P_S,L_i) = f_{\text{prior}}(P_{R^i},L_i) \times f_{\text{free}}(P_S) \tag{2}$$

In this equation, P_{R^i} stands for the set of poses for objects o_j belonging to O_{R^i}. $f_{\text{free}}(P_S)$, indicating the free space, can be determined by running a 2D collision simulation using point cloud observations for each object o_i.

Our primary focus is to determine $f_{\text{prior}}(P_{R^i},L_i)$. To this end, we employ an approach akin to the one described in [11]. 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 most closely matching key is selected, as follows,

$$f_{\text{prior}}(P_{R^i},L_i) = f^k_{\text{prior}}(P_{R^i}); \quad k = \underset{k\in\text{DB}}{\text{argmax}}(F_k \cdot F_{L^i}) \tag{3}$$

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

We now delve into the definition of $f^k_{prior}(P_{R^i})$ in the context of LGMCTS. We aim to provide a unified approach to defining patterns by employing different 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 LGMCTS, κ is modeled as a function of

two 2D positions, denoted as $\kappa(p_0, p_1)$. 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-D for more details), we want distribution to capture the history of sampling. Consequently, we further categorize O_{R^i} based on whether the objects have been sampled in MCTS-Planner. The sets $O^s_{R_i}$ and $O^n_{R_i}$ denote the sampled and non-sampled objects, respectively.

A predefined sampling function f for o_i takes three parameters into account: 1) $N = |O_{R_i}|$ represents the total number of objects forming this pattern 2) $K = |O^s_{R_i}|$ indicates the number of objects already sampled 3) $P^s_{R_i}$ refers to the poses of the sampled objects $O^s_{R_i}$.

The sampling function f has three distinct cases based on K:

1) When $K = 0$, $(x_i, y_i, \theta_i) \sim U$, suggesting that the first object can be placed arbitrarily.
2) For $K = 1$, $(x_i, y_i, \theta_i) \sim U$ and $\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \geq \delta$, imposing that the second object must be distanced from the first by at least δ.
3) When $K \geq 2$, $(x_i, y_i) = \gamma\left(\frac{K}{N}, \kappa(p_0, p_1)\right) + \varepsilon$ 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.

In our current implementation, we have defined patterns 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. However, to enhance clarity for the readers, we offer illustrative figures of the 'lines' pattern in Fig. 4 to shed light on the process of the prior generation.

Noticeably, we divide patterns into 'ordered' and 'unordered' based on if the pattern requires an execution sequence.

D. Monte-Carlo Tree Search (MCTS) for Task Planning

We propose a task planner based on the Monte Carlo Tree Search (MCTS) algorithm to move objects to desired collision-free poses. The distribution list D indicates the preference distributions of the goal poses. We define pose p_i as a desired pose of the object o_i if its probability $f_{d_i}(p_i)$ is higher than a threshold ε. Our MCTS-Planner seeks an action sequence by maintaining a search tree. In MCTS-Planner, each state in the tree represents an arrangement of O: $\{p_1, p_2, ..., p_N\}$, and a list of remaining pose distributions D. The MCTS action set S_A of a state defines a finite number of branches that we expand the state with, which is computed as shown in Algorithm 1. Each MCTS action is labeled as (d_i, j), representing the j^{th} attempt to make progress in sampling d_i. k is the number of attempts we try for each d_i. Note that in ordered patterns, we will not consider sampling an object o_i into d_i if there is an object o_j in the constraints of d_i still away from the goal pose (Line 3). For example, in Fig. 3, there is a pattern "A is on the right and behind B". A will not be sampled to its goal distribution if B is still away from its goal.

Algorithm 1: Get Action Set

Input : s: An MCTS state,
k: The number of attempts for each sampler.
Output: S_A: Action set

```
1 S_A ← ∅;
2 for d_i ∈ s.D do
3     if availableToSample(d_i) then
4         for j ← 0 to k − 1 do S_A.add(d_i, j);
5 return S_A
```

The simulation stage of MCTS-Planner is presented in Algorithm 2. The algorithm consumes the MCTS state s and the attempted action (d_i, j). If the corresponding object o_i is not graspable, we randomly choose a graspable obstacle on top and uniformly sample a free space to place it (Line 2-6). If o_i is graspable, we try to sample it in d_i. (Line 8) If the sampled position is not preferred: $f_{d_i}(p) < \varepsilon$, we find an obstacle o in d_i and uniformly sample a free space to place it (Line 10-14). Similar to the formulation in [15], in LGMCTS, the reward of a node s is the number of finished samplers. That is, $|root.D| - |s.D|$. While MCTS is an anytime search algorithm, in our implementation, MCTS-Planner returns the first found solution. Finally, we prove that MCTS-Planner is probabilistic complete under our setting.

Algorithm 2: Simulation

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

```
1  o_i ← The sampling object in d_i;
2  if o_i not graspable then
3      o ← Randomly choose a graspable object on top of o_i;
4      p ←uniformSampling(o, s);
5      if p then return (o,p);
6      else return None;
7  else
8      p ←sampling(o_i, d_i, s);
9      if f_{d_i}(p) ≥ ε then return (o_i,p);
10     else
11         o ← An obstacle in d_i;
12         p ←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 2. Note that in Algorithm 2, 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

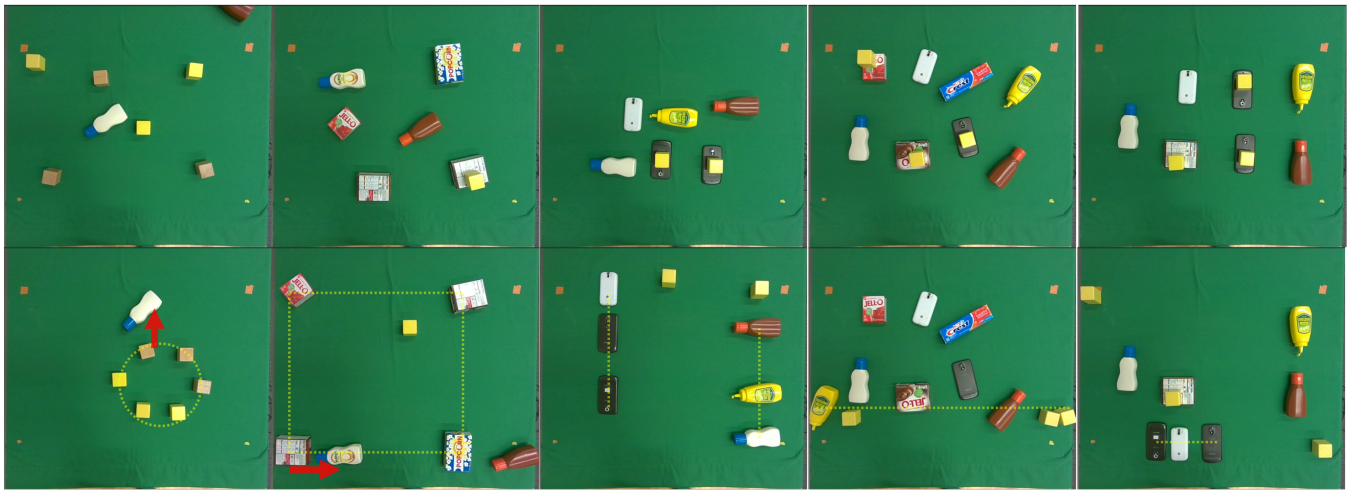

Fig. 5: Results with a real UR5 robot. The language instructions for the five scenes are: (a). "Move all blocks into a circle; while put the white bottle in front of one block;" (b). "Put all boxes into a rectangle; and move the white bottle to the left of one box;" (c) "Move bottles into a line; and formulate all phones into another line;" (d) "Formulate all yellow objects into a line;" (e) "Set all phones into a line;". 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.

	Line (4295)	Circle (3416)	Tower (1335)	Dinner (2440)
StructFormer	47.24%	62.64%	99.10%	28.36%
StructDiffusion	61.49%	81.41%	98.95%	69.38%
LGMCTS-T (ours)	**95.99%**	**95.25%**	**100%**	**100%**

TABLE II: Efficacy of StructFormer, StructDiffusion, and LGMCTS across diverse rearrangement tasks (task counts indicated) from the StructFormer dataset

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 2, 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 $r = \frac{min(C_1, C_2)}{|A_0^*|}$, 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 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

In this part, 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 3) The stability of the LLM parse module, and 4) LGMCTS's performance in actual robotic systems.

A. Baselines

We compare our approach with the following baselines and LGMCTS variants:

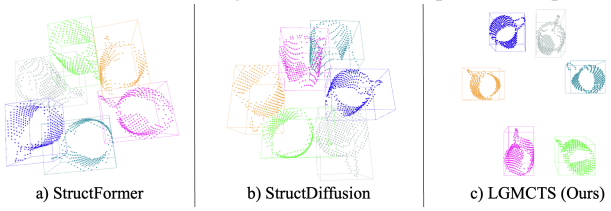

Fig. 6: Compared to StructFormer and StructDiffusion, LGMCTS ensures a collision-free goal arrangement in a qualitative comparison.

StructFormer [2]: StructFormer is a multi-modal transformer architecture specifically designed for language-guided rearrangement tasks.

StructDiffusion [5]: Recognized as the state-of-the-art, StructDiffusion employs a diffusion model combined with a learning-based collision checker for pattern pose generation.

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.

LGMCTS-T: This is a variation of LGMCTS that uses ground-truth data for object and pattern language selection. Given that O_R and L_i are directly provided, LGMCTS-T functions as LGMCTS without the LLM parser.

LGMCTS-L: This represents the full LGMCTS system. It employs the LLM to interpret input from natural language and subsequently produce an action plan.

B. Datasets

StructFormer Dataset [2]: 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 regarded as successful if and only if it meets the language constraints while containing no collision. It is worth noting we do not apply collision checking for the 'tower' task, for collision is not avoidable in that specific task. We solve the 'dinner' pattern by treating it as a pattern composition. It involves rearranging objects such as plates, bowls, forks, spoons, cups, and knives. Within this pattern, the plate and bowl are arranged to form a 'tower,' while the remaining objects are positioned adjacent to this tower forming a 'line'.

LGR-Benchmark Existing datasets for semantic object rearrangement tasks, like StructFormer, have limitations. They typically support only a single pattern per scene and lack crowded scenarios. Moreover, they often overlook the feasibility challenge, especially scenarios like infeasible starting configurations where one object might be placed under another from the outset. Addressing these gaps, we introduce LGR-Bench (**L**anguage-**G**uided **R**earrangement **Bench**mark). This new benchmark presents a novel task termed the "multi-pattern task", which requires multiple pattern goals to be satisfied during the rearrangement process. We also incorporate scenarios with infeasible starting configurations, where objects may initially be stacked. In each scene, we randomly select two patterns from "line", "circle", and "spatial". This LGR-Benchmark is modified from VIMA-Benchmark [3].

C. Semantic Pattern Pose Generation

In this assessment, we draw comparisons between StructFormer, StructDiffusion, and LGMCTS-T. We decided against using LGMCTS-L because, in the StructFormer Dataset, object selection demands an insight into the object's shape and size, which is not the focus of LGMCTS. To ensure an equitable comparison, we provide ground-truth object selections to both StructFormer and StructDiffusion. In this context, LGMCTS showcased exemplary performance across all four rearrangement task categories. As evidenced in TABLE II, LGMCTS posted outstanding success rates: 95.99% for the 'line' pattern, 95.25% for 'circle,' and a perfect 100% for both the 'tower' and 'dinner' patterns. On the other hand, although StructDiffusion improved upon StructFormer's results, it did not rival the success of our approach. A unique characteristic of StructDiffusion is its employment of a collision checker that filters out samples with collisions, providing some clarity to its enhanced performance over StructFormer. For a more tangible comparison, Fig. 6 displays a scene from the circle task, illuminating the distinctions between LGMCTS and the existing benchmarks.

D. Benefit of Joint Modeling

Contrary to previous methodologies like StructFormer and StructDiffusion, which viewed pose generation and action planning as separate challenges, LGMCTS concurrently addresses both pattern generation and action planning. We posit that this integrated approach will render our goal poses more executable than other 'correct' alternatives. To substantiate this idea, we juxtapose the performance of LGMCTS-T and PMCTS using the LGR-Benchmark, providing PMCTS with ground-truth goal poses. The comparative data is presented in TABLE III. Beyond assessing the success rates of planning and execution, we also evaluate the overall success rate and the number of actions suggested by each planning mechanism. Our findings highlight the superiority of LGMCTS over the two-step solutions, even when they are provided with a semantically accurate and collision-free goal pose.

	SR_p	SR_e	SR_a	Steps
PMCTS	82.9%	86.2%	74.1%	6.15
LGMCTS-T (ours)	**97.3%**	**93.4%**	**92.8%**	**5.99**

TABLE III: SR_p represents the success rate of planning. Both PMCTS and LGMTCS-T were set with an identical planning step limit, specifically, 10,000 steps. SR_e denotes the success rate of execution, assessed post-execution of the plan to determine if the outcome aligns with the language-derived constraints. SR_e is the overall success rate. The term 'steps' here refers to the average number of steps in the plan returned by each planner. Smaller Steps mean the action can executed faster.

	SR_p	SR_e	SR_a	Acc_{LLM}
LGMCTS-L	90.9%	83.1%	79.2%	89.3%

TABLE IV: Performance of LGMTC-L on LGR-Benchmark. For Acc_{LLM}, we compare the object selection and pattern selection, if they are the same, we return true.

E. Language Parse Stability

We provide an evaluation of our whole pipeline, i.e. LGMCTS-L on LGR-Benchmark. Apart from the success rate, we also present the accuracy of LLM instruction parse. The result is shown in TABLE IV. From the result, we can find the LLM parsing module of LGMTS has high stability. While the performance of LGMCTS-L lags behind that of LGMCTS-T, it still surpasses that of PMCTS.

F. Physical Robot Experiment

We qualitatively evaluated our system using a UR5e robot outfitted with a D455 depth camera as shown in Fig. 1. We employed the Recognize-Anything-Model (RAM) [18], [19] and an HSV-based color detector to detect object semantics and colors. Selected queries and their corresponding execution outcomes can be viewed in Fig. 5. These real-world robot experiments highlight the capabilities of LGMCTS in intricate 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 its Monte-Carlo 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," *ArXiv*, vol. abs/2109.12098, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237396838

[2] 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.

[3] 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," *arXiv preprint arXiv:2210.03094*, 2022.

[4] I. Kapelyukh, V. Vosylius, and E. Johns, "Dall-e-bot: Introducing web-scale diffusion models to robotics," *IEEE Robotics and Automation Letters*, 2023.

[5] W. Liu, T. Hermans, S. Chernova, and C. Paxton, "Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects," *arXiv preprint arXiv:2211.04604*, 2022.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] K. E. Ferguson and P. Turnbull, *Oh, Say, Can You See?: The Semiotics of the Military in Hawaii*. U of Minnesota Press, 1999, vol. 10.

[12] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[13] K. Gao, S. W. Feng, B. Huang, and J. Yu, "Minimizing running buffers for tabletop object rearrangement: Complexity, fast algorithms, and applications," *The International Journal of Robotics Research*, p. 02783649231178565, 2023.

[14] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1961–1967.

[15] 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.

[16] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, "Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9433–9440.

[17] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[18] 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.

[19] 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.