2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

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

Haonan Chang, Kai Gao, Kowndinya Boyalakuntla, Alex Lee, Baichuan Huang,
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, including Structformer, StructDiffusion, and Code as policies. We also present a new benchmark, the Executable Language Guided Rearrangement (ELGR) Bench, containing tasks involving intricate geometry. With the ELGR bench, we 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 using LLMs to generate 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://lgmcts.github.io/.**

I. Introduction

Tasks like "Set up the kitchen" are intuitive for humans but challenging for robots. The semantic rearrangement problem aims to equip robots with the ability to reorganize scenes based on verbal instructions, integrating natural language understanding with Task And Motion Planning (TAMP). Traditionally, this requires formalizing tasks into symbolic representations and using planners like STRIPS [1], PDDL [2], or MCTS [3]. While effective, this demands expert knowledge, limiting its use.

Recent works use multi-modality transformers and diffusion models to map verbal descriptions to object positions directly from linguistic inputs and RGB-D data [4]–[6]. However, these methods are constrained by offline training and are limited to predefined object categories and patterns.

Large Language Models (LLMs) like GPT [7] and Llama [8] have shown promise in understanding complex scenarios and zero-shot planning, prompting research into their use for language-based TAMP problems [9]–[11]. However, plans generated by LLMs often lack the executability and completeness of those from traditional solvers [12]. This has led to efforts to combine the user-friendliness of LLMs with the robustness of TAMP algorithms like PDDL, STRIPS, or MCTS.

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.

LLM-GROP [13] uses LLMs to convert language tasks into pairwise spatial relationships, followed by a sampling-based TAMP planner, but is limited to handling only pairwise relationships. AutoTAMP [14] translates natural language into formal representations and employs a planner to solve the TAMP task but it solves general semantic rearrangement involving non-discrete, large action spaces.

We introduce Language-Guided Monte-Carlo Tree Search (LGMCTS) for executable semantic object rearrangement, using LLMs for intermediate representations and a planner for feasible plans. Unlike AutoTAMP and LLM-GROP, LGMCTS incorporates parametric geometric priors to handle complex spatial relationships among multiple objects, enabling more nuanced configurations beyond pairwise interactions such as lines or rectangles. LGMCTS integrates task and motion planning with an obstacle relocation strategy, ensuring plans are both semantically coherent and executable.

We also present the Executable Language-Guided Rearrangement (ELGR) benchmark, featuring over 1,600 language queries. LGMCTS excels on ELGR, outperforming Code as Policies, Progprompt, Structformer, and StructDiffusion in both goal feasibility and semantic consistency.

II. Related Works

A. Learning-based Semantic Rearrangement

The semantic rearrangement problem involves creating a rearrangement plan that aligns with a given language description and is physically feasible [15]. This has become crucial in language-driven robotics. CLIPort [4] initiated this by integrating CLIP features with a Transporter network but was limited to basic pick-and-place tasks. Structformer [6] advanced the approach using a transformer model to simulate rearrangements and link language to object poses. Building on Structformer, StructDiffusion [5] employed a pose diffusion model to predict poses from language. However, these methods are limited to single structures or patterns they are trained on, struggling with composite patterns. Moreover, they can produce inexecutable rearrangement goals.

B. LLM-driven Task And Motion Planning

Recent LLM advancements [7], [8] have shown strong performance in various tasks, sparking interest in their use for TAMP [9]–[11], [13], [16]–[30] due to their few-shot and zero-shot reasoning abilities. Early efforts like grounding language into task sequences without retraining LLMs showed potential, as seen in SayCan [10], which converted LLM

plans into robot-executable steps but struggled with task execution failures. Inner Monologue [16] improved this by using real-time feedback but still generated suboptimal plans. Other approaches, like Code as Policies and Progprompt [17], leveraged LLMs for policy code generation, demonstrating potential in behavioral logic but falling short on complex object rearrangement tasks due to LLMs' limitations in long-horizon planning [12], [25].

To enhance reliability and interpretability, recent works [12], [14], [25], [26], [28] have focused on translating language commands into representations suitable for traditional TAMP algorithms. Text2Motion [20] combines LLMs with a geometric feasibility planner but is less efficient in large spaces than MCTS-based planners. LLM-GROP [13] and AutoTAMP [14] use LLMs for generating task representations and employ planners for rearrangement, yet they are limited by simplistic object relationships and non-discrete action spaces. To overcome these challenges, we introduce LGMCTS, which integrates parametric geometric priors and utilizes MCTS's exploration efficiency to solve complex rearrangement tasks effectively.

III. Preliminaries

A. Problem Formulation

Semantic rearrangement involves organizing a scene based on natural language descriptions. A key insight is that a language-described goal represents a distribution of possible positions rather than a specific one. For example, "Put the mug on the right side of the bowl" implies a uniform distribution over the area to the right of the bowl. Knowing the position distribution for each object allows us to sequentially sample their poses, effectively converting the semantic rearrangement problem into 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 $\mathscr{F} = \{f_i : p(o_i) \sim f_i | o_i \in O_R\}$, where $p(o_i)$ refers to the position of object o_i. Here, $O_R \subseteq O_S$ denotes the objects requiring an action based on L, 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 [3] 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),

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

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

A. Language Parsing & Object Selection

During the language parsing stage, we parse L into $\mathscr{F} = \{f_i : p(o_i) \sim f_i | o_i \in O_R\}$. Similar to previous methods [11], we conduct an automated prompt engineering to guide the LLM to perform the parsing. Fig. 1 showcases how prompt engineering is implemented. Essentially, the language model translates user requirements into structured goal configurations and constraints that guide task execution.

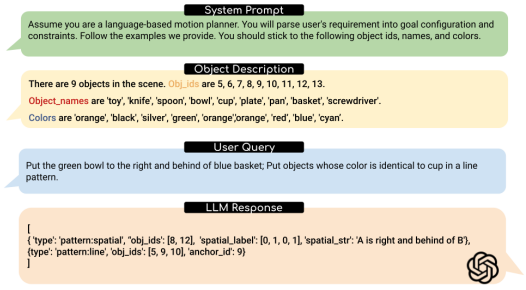

Fig. 1: An example of language parsing. We are using GPT-4 [7] in this work.

Consider the example in Fig. 1. It begins with the **system prompt** that guides the LLM in interpreting user queries. Next, an **object description** is provided, including semantic labels, colors, and unique IDs for the objects in the scene. We use the Recognize Anything Model (RAM) [31], [32] for semantic labels and a color detector for identifying object

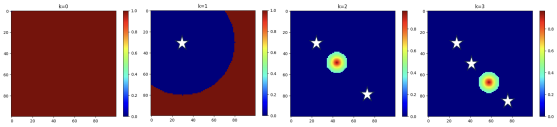

Fig. 2: Visualization of (x,y) prior for 'line' pattern. From left to right: $K=0$, $K=1$, $K=2$, $K=3$, where $K=|O_R^{sampled}|$, the number of sampled object poses. White star marks are sampled poses. When $K=0$, the pose can be sampled anywhere. When $K=1$, it needed to sampled outside a circle region.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

colors. Finally, a **user query** describing the rearrangement goal is given, and the LLM returns a structured response, indicating how many patterns exist and which objects should be selected to form them.

B. Parametric Geometric Prior

As mentioned in Section III-A, If we know the goal position distribution for each object, the semantic rearrangement problem reduces to a sequential sampling problem. Key considerations include: (1) The position distribution changes with the sampling progress. For instance, when placing objects A, B, and C in a line, the distributions for A and B are unbounded, while C must align with A and B. (2) The position distribution must avoid collisions. (3) A database of spatial distributions can be built, but a flexible mechanism is needed to match LLM-inferred distribution types with items in the database.

We show in the following how to compute the pose 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 list of current poses for all objects in the scene, and L is the natural language command.

We compute $f_i(p_i|P_S,L)$ as an element-wise product of two components, a pattern prior function $f_{\text{prior}}(p_i|P_R,L)$ 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 is the list of current poses of all objects in O_R. We note that P_R is a subset to P_S. The function $f_{\text{free}}(p_i|P_S)$ is set to 1 if p_i is collision-free from the remaining poses in P_S, and to 0 otherwise. f_{free} is determined by running a 2D collision simulation using point cloud observations for each object $o_i \in O_S$.

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 [10]. 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,

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

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 $\Theta(L)$.

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 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 $f^K_{\text{prior}}(p_i|P_R))$ of sampling a pose $p_i = (x_i,y_i,\theta_i)$ for next object $o_i \in O_R^{unsampled}$ is given as follows.

- If $|O_R^{sampled}|=0$ then $(x_i,y_i,\theta_i) \sim U$, suggesting that the first object can be placed arbitrarily.
- If $|O_R^{sampled}|=1$ then $\sqrt{(x_i-x_0)^2+(y_i-y_0)^2} \le \delta$ 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 $(x_i,y_i) = \gamma(\frac{|O_R^{sampled}|}{|O_R|}, \kappa(p_0,p_1)) + \varepsilon$ where $\varepsilon \sim G(0,\sigma)$, here G represents a Gaussian distribution with mean zero and variance σ, and $\theta = atan2(1,\gamma'(\frac{K}{N}))$. θ represented the rotation angle of the object.

Our parametric geometric representation can model any shape expressible as a parametric curve, including shapes like "line," "circle," "rectangle," "tower," "spatial:left," and "spatial:right." Due to space constraints, detailed definitions of γ and κ for these patterns are omitted. Fig. 2 provides an example of a parametric geometric prior. We categorize patterns as "ordered" or "unordered" based on the necessity of a specific 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 $\{L_i\}$ of sub-instructions L_i . For example, L can be a composite instruction: $L=\{L_1,L_2\}$, where L_1 refers to placing objects A, B, and C in a line, and L_2 refers to placing object A on the left of B. Each sub-instruction $L_i \in L$ is associated with a subset of objects $O_{R_i} \subseteq O_R$. In our example, $O_{R_1}=\{A,B,C\}$ and $O_{R_2}=\{A,B\}$. In the sequential sampling process described before, we 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

As previously mentioned, our rearrangement problem can be formulated as a sequential task. In each step, we sample a pose p_i for each object $o_i \in O_R$ according to a pose

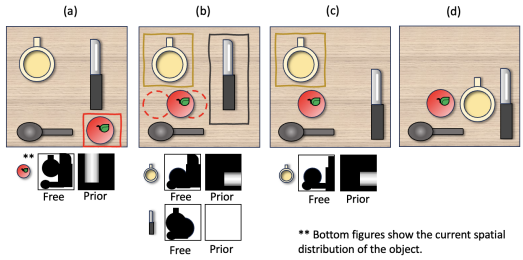

Fig. 3: A minimal example illustrates our MCTS-Planner's aim to arrange a table. The language description provided is: "Can you please put the apple behind the spoon? And I also want the cup at the right of the apple." The top row displays the current scene arrangement, while the bottom row shows the f_{prior} and f_{free} for the object being manipulated. $f = f_{prior} \times f_{free}$. In spatial distribution figures, black represents probability 0, and white probability 1.

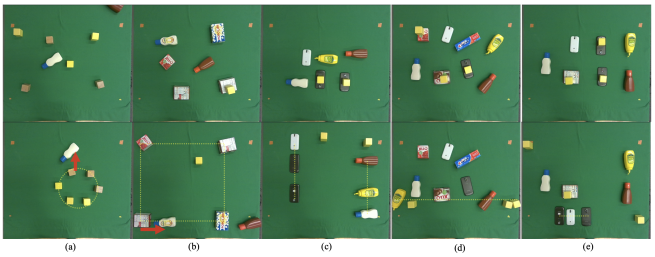

Fig. 4: Real world demonstration with a UR5e robot. The language instructions for the five scenes are: (a) "Move all blocks into a circle; while put the white bottle behind one block;" (b) "Put all boxes into a rectangle; and move the white bottle to the right of one box;" (c) "Move bottles into a line; 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 UR5e. 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.

distribution f_i selected by the LLM and computed as described in Section IV-B. Once we complete all samplings, all objects will have been placed in their desired locations. However, this task cannot be executed by the robot 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 MCTS algorithm to simultaneously arrange the task and address object relocation issues. The objective of our MCTS-Planner is to fulfill all object position requirements defined by $\mathscr{F}$, the spatial distribution list. 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 spatial requirements $\mathscr{F}_r$, where $\mathscr{F}_r \subseteq \mathscr{F}$.

The MCTS-Planner operates through four phases: selection, expansion, simulation, and back-propagation, as described in Section III-B, adhering to the methodology from [33] except for the simulation stage. The reward for transitioning to a new state s is quantified by the reduction in the number of spatial requirements, that is, $|\mathscr{F}| - |\mathscr{F}_r|$. This reward structure mirrors the approach in [33], 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 of placing an object o_i in a pose $p_i \sim f_i$, where f_i is the pose distribution explained in the previous section. If the targeted object o_i is not reachable, e.g. there exist some obstacles above it, a reachable obstacle is randomly selected from those above it, and a collision-free pose within the workspace is sampled for relocation (Lines 2-5). If o_i is reachable, an attempt is made to sample its pose with distribution f_i (Line 7). If the sampled pose is not collision-free, we will randomly choose an obstacle in the collision and relocate it to a collision-free pose within the workspace (Lines 9-13). Note that obstacle relocation may add previously placed objects back to the requirement list $\mathscr{F}_r$. However, this 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.

Fig. 3 presents an illustrative 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 behind the spoon? And I also want the cup to be at the right of the apple." In response, the LLM generates the spatial distribution list $\mathscr{F} = \{f_1, f_2\}$, where f_1 is for positioning the apple behind the spoon, and f_2 is for placing the cup to the right of the apple. Fig. 3(a) illustrates the initial arrangement of objects. Given the dependency of f_2 on the apple's position, K actions will be sampled for f_1, but none for f_2 in this initial setup. Fig. 3(b) depicts the outcome of an action of sampling a pose from f_1, where the apple is relocated according to f_1. The dashed-line circles represent the other $K-1$ actions originating from the root node. After sampling f_1, we are left with $\mathscr{F}_r = \{f_2\}$ as shown in Fig. 3(b), and an attempt is made to position the cup to the right of the apple. However, the goal position is in collision with the knife, so we need to relocate the knife. Fig. 3(c) demonstrates the knife's relocation, maintaining $\mathscr{F}_r = \{f_2\}$. Ultimately, Fig. 3(d) showcases the final planning result.

Algorithm 1: Simulation

Input : s: an MCTS state,
f_i: a pose distribution (place o_i in a pose $p_i \sim f_i$).
Output: (o, p): a rearrangement action (place o in pose p).

1 **if** o_i *is not reachable* **then**
2 Select a reachable object o that is blocking object o_i;
3 $p \leftarrow$uniformSampling(o, s);
4 **if** p *exists* **then return** (o, p);
5 **else return failure**;
6 **else**
7 $p \leftarrow$sampling(o_i, s, f_i);
8 **if** *collisionFree*(o_i, p, s) **then return** (o_i, p);
9 **else**
10 $o \leftarrow$ Randomly choose an obstacle in collision;
11 $p \leftarrow$uniformSampling(o, s);
12 **if** p *exists* **then return** (o, p);
13 **else return failure**;

V. EXPERIMENTS

A. Baselines

We compare our approach with the following baselines.
Structformer [6]. It is a multi-modal transformer specifically designed for language-guided rearrangement tasks.

Method	Line (4295)	Circle (3416)	Tower (1335)	Dinner (2440)
LFSP* [11], [17]	41.16%	51.75%	88.80%	27.05%
Structformer [6]	47.24%	62.64%	99.10%	28.36%
StructDiffusion [5]	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, LFSP are evaluated on 1150 (10%) randomly selected scenes of the Structformer dataset.

StructDiffusion [5]. It employs a diffusion model combined with a learning-based collision checker for pattern pose generation.

LLMs as Few-Shot Planners [11], [17]. We integrate *Code as Policies* and *Progprompt* into our evaluation, using the former to generate policy code and the latter to produce Pythonic code. Since this code is not directly usable, we modify it into action sequences (object IDs and target poses) for our TAMP planner. The LLM first processes scene details like object names, IDs, textures, poses, and region boundaries, then converts natural language commands into ordered lists of object IDs and goal poses. For the Structformer dataset, we use structured goal specifications directly to avoid redundancy. During evaluations, we provide example scenes with a defined output format and ground-truth sequence, calling this approach LLMs as Few-Shot Planners (LFSP).

Pose+MCTS. The Pose+MCTS (PMCTS) approach assumes a collision-free, semantically aligned goal pose. If the target space is occupied, MCTS searches for a viable plan to place objects in their goal poses. Here, MCTS functions solely as a motion planner, following a two-step process that separates goal pose determination from task planning.

B. Structformer Dataset

We evaluate the goal pose generation ability using the Structformer dataset's test set, comprising approximately 11,500 rearrangement tasks categorized into four patterns: line, circle, tower, and dinner. A successful rearrangement plan adheres to language constraints and is collision-free, except in the "tower" task where collisions are unavoidable. The "dinner" task is treated as a composition of patterns, arranging items like plates and bowls into a "tower" with other utensils lined up beside it. Both Structformer and StructDiffusion use object selection based on shape and size, but for our evaluation, we provide ground-truth object selection, as our setup does not rely on these criteria. Since the tasks already specify objects based on language instructions, we did not use the LLM parser in LGMCTS for this dataset.

As shown in TABLE I, LGMCTS outperformed all baselines in the four task categories, achieving success rates of 95.99% for "line," 95.25% for "circle," and 100% for both "tower" and "dinner." LSFP underperformed due to LLMs' inability to produce goal patterns with high geometric fidelity. While StructDiffusion improved over Structformer, it did not match LGMCTS's effectiveness.

The dataset strictly evaluates the accuracy of collision-free geometric patterns, ignoring plan executability—an inherent limitation. This is addressed in our proposed ELGR-Benchmark (Section V-C), which introduces the PMCTS method to verify plan executability. As PMCTS uses collision-free ground-truth poses, it was excluded from further comparisons on this dataset.

Method	SR_p	SR_{ep}
LFSP [11], [17]	**100%**	45.2%
Structformer [6]	n.a.	n.a.
StructDiffusion [5]	n.a.	n.a.
PMCTS	82.9%	74.1%
LGMCTS (Ours)	90.9%	**79.2%**

TABLE II: SR_{ep}, the executable plan success rate, measures both the success of planning and the execution of plans, ensuring final object positions meet language-based constraints. SR_p, a subset of SR_{ep}, only tracks planning success. Both PMCTS and LGMCTS are capped at 10,000 planning steps; if this limit is exceeded, typically due to dense object placement, the motion planning is deemed a failure.

C. ELGR-Benchmark

Existing datasets for semantic object rearrangement, like Structformer, are limited by featuring only one pattern per scene and lacking crowded scenarios. They also do not address feasibility challenges, such as infeasible starting configurations (e.g., one object placed under another). To address these gaps, we introduce ELGR-Bench (**E**xecutable **L**anguage-**G**uided **R**earrangement **Bench**mark), which includes scenarios with infeasible starting configurations and tasks that require unstacking before rearrangement.

A key feature of ELGR-Bench is the "multi-pattern task," which requires satisfying multiple pattern goals (e.g., "line," "circle," "rectangle," "spatial") during rearrangement. For each scene, we randomly combine two patterns to create the multi-pattern task. Success is based on the executability of the generated plan and its adherence to semantic requirements. ELGR-Bench extends the VIMA-Benchmark [34].

In our benchmark, we compared LGMCTS with two baselines: LFSP and PMCTS, excluding Structformer and StructDiffusion due to their inability to handle composite geometric patterns. LFSP uses an LLM to plan goal poses and action sequences simultaneously, while PMCTS follows a two-step method, using a given goal pose and MCTS for action planning. LGMCTS uniquely integrates goal generation with action planning for more executable outcomes. As shown in TABLE II, LFSP achieves a 100% planning success rate, but over 50% of these plans are inexecutable, as indicated by SR_{ep} scores. LGMCTS achieves a 90% success rate, with about 80% of plans being executable. This highlights LLM limitations in direct TAMP solving and demonstrates LGMCTS's advantage over PMCTS, even with accurate goal poses.

D. Real Robot Experiments

We qualitatively evaluated our system using a UR5e robot with a D455 depth camera, employing the Recognize-Anything-Model (RAM) [31], [32] and an HSV-based color detector for object and color detection. Five language instructions involving different objects and configurations were tested, as shown in Fig. 4. For example, Fig. 4(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. 4. 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. Its main limitation is the extended execution time for complex scenes, highlighting the need for improved tree search efficiency. Future research should focus on adapting LGMCTS to more complex rearrangement scenarios.

REFERENCES

[1] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[2] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[3] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, pp. 72–83.

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

[5] 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.

[6] 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.

[7] 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.

[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] 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.

[10] 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.

[11] 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.

[12] 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

[13] 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.

[14] 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.

[15] H. Chang, K. Boyalakuntla, S. Lu, S. Cai, E. Jing, S. Keskar, S. Geng, A. Abbas, L. Zhou, K. Bekris, and A. Boularias, "Context-aware entity grounding with open-vocabulary 3d scene graphs," 2023. [Online]. Available: https://arxiv.org/abs/2309.15940

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[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] 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.

[32] 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.

[33] 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.

[34] 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.