Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

"Set It Up!": Functional Object Arrangement with Compositional Generative Models

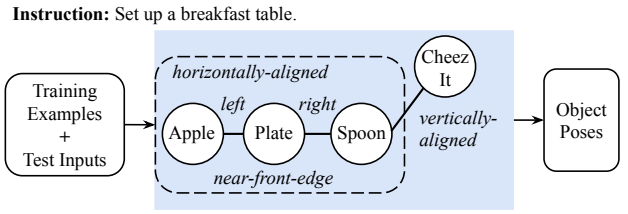
Yiqing Xu*, Jiayuan Mao†, Yilun Du†, Tomas Lozáno-Pérez†, Leslie Pack Kaebling†, and David Hsu*
*School of Computing, Smart System Institute, National University of Singapore
† CSAIL, Massachusetts Institute of Technology

***Abstract*—This paper studies the challenge of developing robots capable of understanding under-specified instructions for creating functional object arrangements, such as "set up a dining table for two"; previous arrangement approaches have focused on much more explicit instructions, such as "put object A on the table." We introduce a framework, *SetItUp*, for learning to interpret under-specified instructions. SetItUp takes a small number of training examples and a human-crafted program sketch to uncover arrangement rules for specific scene types. By leveraging an intermediate graph-like representation of *abstract spatial relationships* among objects, SetItUp decomposes the arrangement problem into two subproblems: i) learning the arrangement patterns from limited data and ii) grounding these abstract relationships into object poses. SetItUp leverages large language models (LLMs) to propose the abstract spatial relationships among objects in novel scenes as the constraints to be satisfied; then, it composes a library of diffusion models associated with these abstract relationships to find object poses that satisfy the constraints. We validate our framework on a dataset comprising study desks, dining tables, and coffee tables, with the results showing superior performance in generating physically plausible, functional, and aesthetically pleasing object arrangements compared to existing models.** [1]

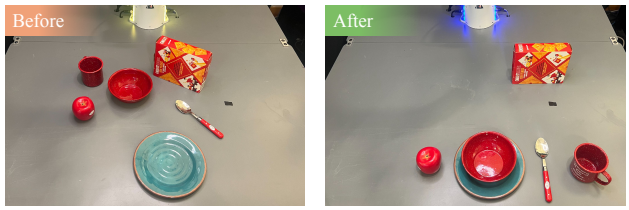
I. Introduction

Developing robots capable of understanding human goals and making plans to achieve them is a crucial step forward in embodied intelligence. However, this endeavor is complicated by the inherent ambiguity and under-specification of a broad range of human goals. For example, consider one of the most common yet time-consuming daily tasks [American Time Use Survey; 26]: setting and cleaning up tables. Human instructions are inherently under-specified; they can be as vague as "Could you set up a dining table?" Understanding such under-specified instructions is fundamentally challenging, requiring robots to understand and ground physical feasibility, object functionality, commonsense aesthetics, and user preferences.

While a large body of work has tackled the problem of generating object arrangements at a table-top or even at a house scale, most of it focused on grounding spatial relationships and making robot plans under explicit and unambiguous instructions, *e.g.*, "put the plate next to the fork" [3, 5, 8, 19, 20, 23, 24, 34, 36]. By contrast, in this paper, we consider the task of generating object arrangements based on under-specified descriptions, such as "*tidy up the study table*," "*set a Chinese dinner table for two*," and "*make space on a coffee table for a chess game*." Given such ambiguous instructions, our goal is to generate configurations of objects that are physically feasible, functional, aesthetic, and aligned with user preferences. We call this task *functional object arrangement* (FORM), and it presents three challenges.



(a) We first generate abstract spatial relationships given input instructions and objects, and then ground these relationships into object poses.



(b) Initial and final object arrangement.

Fig. 1. (a) At test time, given human instruction and a set of objects (possibly unseen during training), our framework SetItUp first generates a set of multi-ary spatial relationships among subsets of objects. These spatial relationships are based on a library of abstract spatial relationships and are visualized in the multi-ary graphical representation. (b) Then, we employ a compositional diffusion model to generate concrete object poses that a robot can execute based on a general motion planner.

First, unlike many purely spatial relationships that have large-scale annotations [15] or can be synthetically generated according to human-written rules, global scene annotations for *functional* object arrangements are usually scarce. This data scarcity issue becomes more severe when considering the learning of subjective preferences of a particular user. Second, not only is there very little data available, but there is also a need to make arrangement plans for a wide variety of objects, many of which might be unseen during training. The criteria for successful *functional* object arrangement are not one-size-fits-all; they must apply to different objects and settings, leading to numerous acceptable arrangements, posing a significant challenge in generalization. Finally, pinpointing a universal measure or rules for the commonsense object arrangement is challenging due to its multifaceted nature.

To tackle the under-specified and multifaceted nature of functional object arrangements, our first contribution is a novel

[1] Project page: https://setitup-rss.github.io/

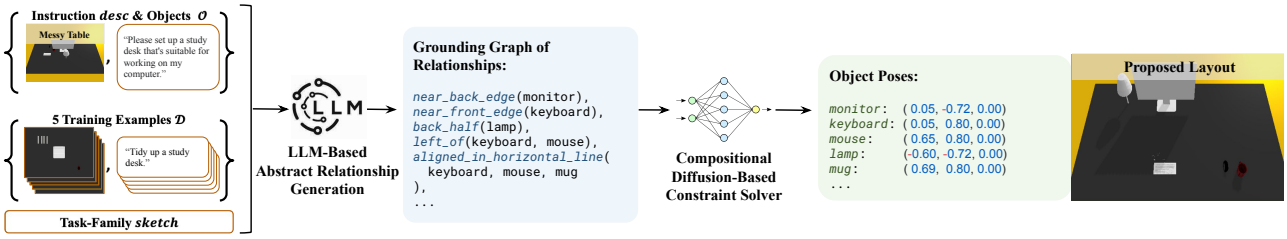


Fig. 2. Overall architecture of SetItUp. Given a novel instruction *desc* and a set of objects $\mathcal{O}$, we first query an LLM to induce an abstract spatial relationship description of the target object arrangements. The input to the LLM also includes a handful of training examples $\mathcal{D}$ and a human-defined task-family *sketch*. Next, we ground these abstract relationships into object poses by composing a library of diffusion models to generate object poses that simultaneously comply with all proposed spatial relationships.

task formulation and the corresponding evaluation metrics. For a given scene type, *e.g.*, study desks or dining tables, we formalize the task as generating the poses of objects given their category names and shapes, based on a small set of examples. We used 5 examples per scene type in our experiments. These training examples include paired instructions and reference object arrangements, which can be easily collected by users. Furthermore, to enable strong and controllable generalization to unseen instructions and objects, for each scene type, we provide a short program *sketch* written in Python-like domain-specific language as a hierarchical generative model specification. The sketch provides a list of function names, signatures, and descriptions, but does not include any implementation. Essentially, it decomposes the arrangement task into simpler subproblems as a global "guideline" for machine learning algorithms. Finally, our benchmark also comes with a collection of rule-based metrics and human experiment rubrics for holistically evaluating different solutions in terms of their physical feasibility, functionality, and other aesthetic aspects.

Our second contribution is a novel hierarchically generative approach, as illustrated in Figure 1a. Its key idea is to use a library of abstract spatial relationships, e.g., *left-of, horizontally-aligned*, as its intermediate prediction task. This breaks down the task of predicting object poses into two steps: generating a set of object relationships that should be satisfied in the final arrangement, and generating concrete object poses that comply with these relationships. In our system SetItUp, we leverage large language models (LLMs) for the first prediction task and use a library of compositional generative models to ground the relationships into poses.

This problem decomposition brought by our abstract relationship representation significantly improves the model along all evaluation metrics, especially its generalization to scenarios involving unseen objects and novel instructional contexts. On a new dataset we collected, which comprises three scene types: study desks, dining tables, and coffee tables, we compare our framework with methods based solely on neural generative models (inspired by Liu et al. [18]) or large language models (inspired by Wu et al. [30]). Both qualitative examples and quantitative human studies demonstrate that our model surpasses baselines in all aspects of generating physically plausible, functional, and aesthetically appealing object arrangement plans.

II. Problem Formulation

We frame functional object arrangement (FORM), using a novel few-shot learning paradigm. Specifically, each object arrangement *task family* is a tuple $\langle \mathcal{D}, desc, \mathcal{O}, sketch \rangle$, where $\mathcal{D}$ represents a small dataset of examples corresponding to a specific scene type and user preferences. Each training example $d \in \mathcal{D}$ is a tuple $\langle desc_d, \mathcal{O}_d, \mathcal{P}_d \rangle$. Here, $desc_d$ denotes a natural language instruction for a specific task in that family (*e.g.*, "set up a dining table for two") that implies a preferred object arrangement. $\mathcal{O}_d$ and $\mathcal{P}_d$ encode a desired scene configuration that fulfills the instruction $desc_d$. Specifically, $\mathcal{O}_d = \{o_0, o_1, \ldots, o_{N-1}\}$ denotes a collection of input objects categorized by two static properties: types (n_i) and shapes (g_i, represented as 2D bounding boxes in this paper). The desired "output" arrangement $\mathcal{P}_d = \{p_0, p_1, \ldots, p_{N-1}\}$ is represented by a planar pose for each object, expressed isn coordinates and orientation (x, y, θ)[2]. Since we only focus on tabletop arrangement tasks in this paper, all poses are represented in a canonical table frame. We need very few (on the order of five) examples, making it easy for them to be collected by users.

At deployment time, we consider a new task instruction *desc* and a novel set of objects $\mathcal{O} = \{o_0, o_1, \ldots, o_{N-1}\}$, which may vary in sizes and categories from the training examples; the system must determine a set of poses $\mathcal{P}$ for each object. These poses should satisfy user instructions by being physically plausible, functional (logically positioned for their intended use), aesthetic (*e.g.*, arrangements are visually appealing), and aligned with human preferences (conforming to the pattern of arrangements communicated in the training data $\mathcal{D}$ as well as meet specific additional criteria in the user's instruction *desc*).

To enable strong and controllable generalization to an unseen set of objects and instructions, we introduce a program *sketch* for each task family. This *sketch* defines a hierarchical generative model for generating object arrangements for a particular family of tasks. It uses a Python-like syntax to outline the task-solving procedures with only function names, signatures, and descriptions, but no implementations. Essentially, it

[2]This basic problem formulation can be extended to 3D shapes and poses.

decomposes the problem of generating the full arrangement into a hierarchy of smaller sub-problems. Each sub-problem has a simpler input-output specification, such as categorizing object types and generating the arrangement of a subset of the object. In this paper, we focus on leveraging human-defined sketch descriptions for three domains: *study desks, coffee tables, and dining tables*. Moving forward, the sketches could conceivably be generated from free-form language instructions from humans as in Zelikman et al. [35].

III. Set It Up

To address the challenges of data-efficient learning and robust generalization in *functional* object arrangement (FORM) tasks, we introduce a novel hierarchical generative framework, namely *SetItUp*. The key idea of SetItUp is to use a library of abstract spatial relationships to construct a grounding graph, which serves as an intermediate representation for solving the arrangement task. This library is composed of basic spatial relationships such as *left-of* and *horizontally-aligned*, which can be unambiguously defined by considering the geometric features of objects. Based on this library, we can decompose the problem of learning to generate object poses given their categories, shapes, and the task instructions into two subproblems: learning to generate the set of spatial relationships that encodes the arrangement plan, and learning to generate object poses based on a set of specified spatial relationships.

Figure 2 illustrates the overall framework. Central to our framework is the use of abstract spatial relationships as the intermediate representation. This high-level abstraction over object poses naturally breaks down FORM tasks into two generative sub-tasks. The first sub-task involves a LLM-based symbolic generative model that generates the functional abstract relationships for the test objects. The second sub-task involves a diffusion-based generative model that takes these abstract relationships as input and propose the corresponding object poses. Essentially, SetItUp combines large language models (LLMs) as a repository of commonsense knowledge and as a strong few-shot learner for generating abstract descriptions of the arrangement, and compositional diffusion models as powerful generative models to ground these spatial relationships into object poses. Given the task input tuple $\langle \mathcal{D}, desc, \mathcal{O}, sketch \rangle$, we first convert all training examples in $\mathcal{D}$ into a language description based on the abstract spatial relationship library. Next, we prompt an LLM, based on the task inputs, to generate a set of abstract spatial relationships among test objects. This gives us a factor graph-like representation of the desired arrangement. Finally, we leverage a composition of the diffusion-based generative models associated with each abstract relationship to predict the output object poses associated with each object, which comply with the factor graph description generated by the LLM.

In the following sections, we will first present the library of abstract spatial relationships (Section III-A), where each relationship is associated with a simple geometric-rule-based classifier and a neural network-based generative model. Following that, we will introduce a compositional diffusion model inference algorithm capable of generating object poses based on the factor graph specification of object relationships. Finally, in Section III-C, we specify our strategy for prompting an LLM based on the training examples, test inputs, and task-family sketches to propose the abstract spatial relationships.

TABLE I
THE SET OF ABSTRACT SPATIAL RELATIONSHIPS AMONG OBJECTS. THESE RELATIONSHIPS ARE FORMALLY DEFINED BY RULES BASED ON 2D OBJECT SHAPES AND POSES IN THE CAMERA FRAME.

Unary Relationships	
central_column	central_row
central_table	left_half
right_half	front_half
back_half	near_left_edge
near_right_edge	near_front_edge
near_back_edge	
Binary Relationships	
horizontally_aligned	vertically_aligned
horizontal_symmetry_on_table	vertical_symmetry_on_table
left_of	right_of
centered	on_top_of
Ternary Relationships	
horizontal_symmetry_about_axis_obj	vertical_symmetry_about_axis_obj
Variable-Arity Relationships	
aligned_in_horizontal_line	aligned_in_vertical_line
regular_grid	

A. The Spatial Relationship Library

Our spatial relationship library, $\mathcal{R}$, encompasses 24 basic relationships listed in Table IV. Each abstract relationship takes a (possibly variable-sized) set of objects as arguments — and describes a desired spatial relationship among them. These abstract relationships provide one level of abstraction over 2D poses and, therefore, can serve as a natural input-output format for LLMs. On the other hand, they are sufficiently detailed to be directly interpreted as geometric concepts. To extend the system to task families requiring novel relationships, it would be straightforward to augment this set. Furthermore, these spatial constraints are defined unambiguously based on simple geometric transformations, which enables us to effectively classify and generate random object arrangements that satisfy a particular relationship. Finally and most importantly, this finite set of relationships can be *composed* to describe an expansive set of possible scene-level arrangements with indefinite number of objects and relationships.

Formally, each abstract spatial relationship R is associated with two models: a classifier model h_R and a generative model f_R. Let k_R be the arity of the relationship. The classifier h_R is a function, denoted as $h_R(g_1, \ldots, g_{k_R}, p_1, \ldots, p_{k_R})$, that takes the static properties of k_R objects (shapes represented by 2D bounding-boxes $\{g_i\}$ in our case) and their poses $\{p_i\}$ as input, and outputs a Boolean value indicating whether the input objects satisfy the relationship R. Similarly, the generative model f_R is a function that takes the static properties of k_R objects $\{g_i\}$ and produces samples of

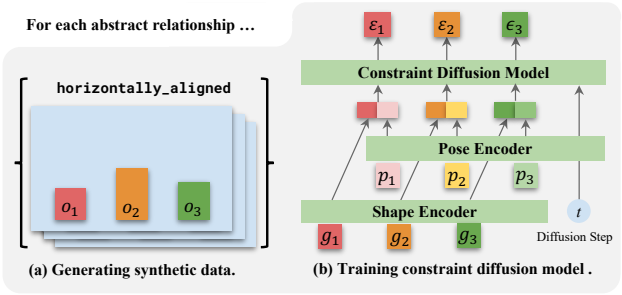


Fig. 3. Training a single constraint diffusion model involves a two-stage process. First, for every abstract relationship listed in Table IV, we generate a synthetic dataset based on predefined rules. Then, we train a relation-specific diffusion model that can draw samples of object poses that satisfy the relationship.

$\{p_i\}$ from the distribution[3] $q_R\left(p_1, \ldots, p_{k_R} \mid g_1, \ldots, g_{k_R}\right) \propto \mathbb{1}\left[h_R\left(g_1, \ldots, g_{k_R}, p_1, \ldots, p_{k_R}\right)\right]$, where $\mathbb{1}[\cdot]$ is the indicator function. In general, R (and therefore the associated h_R and f_R) can be a set-based function and, therefore, may have a variable arity, in which case $k_R = \star$.

Grounding graph. Given the library of abstract spatial relationships $\mathcal{R}$, we can encode a desired spatial arrangement of an object set $\mathcal{O}$ as a graph of ground spatial relations, $\mathcal{G} = \{r_i(o^i_1, \ldots, o^i_{k_i})\}_i$ where each $r_i \in \mathcal{R}$ is a relationship (such as *horizontally-aligned*), k_i is the arity of r_i, and $o^i_1, \ldots, o^i_{k_i}$ are elements of $\mathcal{O}$. The objective is to produce a set of poses $\mathcal{P} = \{p_i\}$, such that, for all elements $(R, (o_1, \ldots, o_{k_R})) \in \mathcal{G}$, $h_R(g_1, \ldots g_{k_R}, p_1, \ldots, p_{k_R})$ is true, where g_i is the corresponding shape of o_i.

We can interpret $(\mathcal{P}, \mathcal{G})$ as a graph of constraints. Based on the probabilistic distribution specified with all relationships in $\mathcal{R}$ (*i.e.*, uniform distributions over allowable assignments), then $(\mathcal{P}, \mathcal{G})$ can also be interpreted as a factor graph, specifying a joint distribution over values of $\mathcal{P}$. This will be our inference-time objective.

Classifying spatial relationships Since all relationships used in our examples are unambiguously defined based on simple geometric transformations (*e.g.*, by comparing the 2D coordinate of objects in a canonical table frame), we use a small set of rules to construct the classifier function h_R. We include the details of the rules in the appendix C. They could instead be learned in conjunction with the generative models for relationships that are defined through examples only.

Generating spatial relationships. One straightforward approach to the overall problem would be to hand-specify f_R for each relationship type and use standard non-linear optimization methods to search for $\mathcal{P}$. There are two substantial difficulties with this approach. First, we may want to extend to relationship types for which we do not know an analytical form for f, and so would want to acquire it via learning. Second, the optimization problem for sampling assignments to P given a ground graph representation $(\mathcal{P}, \mathcal{G})$ is highly non-convex and hence very difficult, and the standard method would typically require a great deal of tuning (e.g., of the steepness of the objective near the constraint boundary).

For these reasons, we adopt a strategy that is inspired by Yang et al. [33], which is 1) to pre-train an individual diffusion-based generative model for each relationship *type* $R \in \mathcal{R}$ and 2) to combine the resulting "denoising" gradients to generate samples of $\mathcal{P}$ from the high-scoring region. One significant deviation from the method of Yang et al. [33] is that we train diffusion models for each relationship type completely independently and combine them only at inference time, using a novel inference mechanism that is both easy to implement and theoretically sound.

Our model architecture and the training paradigm are illustrated in Figure 3. Specifically, for each relationship type $R \in \mathcal{R}$ of arity k, we require a training dataset of *positive examples* $\mathcal{D}_R = \{(g_1, \ldots, g_k, p_1, \ldots, p_k)\}$ specifying satisfactory poses $\{p_i\}$ for the given object shapes $\{g_i\}$. Note that for set-based relations, the examples in the training set may have differing arity. For all relations in our library, we generate these datasets synthetically, as described in the appendix C.

We construct a denoising diffusion model [9] for each relationship R where the distribution $q_R(p_1, \ldots, p_k \mid g_1, \ldots, g_k)$ maximizes the likelihood $\{(g_1, \ldots, g_k, p_1, \ldots, p_k)\}$. We denote this distribution as $q_R(\boldsymbol{p} \mid \boldsymbol{g})$ for brevity, where $\boldsymbol{p}$ and $\boldsymbol{g}$ are vector representations of the poses and the shapes, respectively. We learn a denoising function $\epsilon_R(\boldsymbol{p}, \boldsymbol{g}, t)$ which learns to denoise poses across a set of timesteps of t in $\{1, \ldots, T\}$:

$$\mathcal{L}_{MSE} = \mathbb{E}_{(\boldsymbol{p},\boldsymbol{g})\sim\mathcal{D}_R, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}), t\sim U(0,T)} \left[\left\|\epsilon - \epsilon_R(\sqrt{\bar{\alpha}_t}\boldsymbol{p} + \sqrt{1-\bar{\alpha}_t}\epsilon, \boldsymbol{g}, t)\right\|^2\right],$$

where t is a uniformly sampled diffusion step, ϵ is a sample of Gaussian noise, and $\bar{\alpha}_t$ is the diffusion denoising schedule. In the case that R has fixed arity, the network ϵ_θ is a multi-layer perceptron (MLP); however, when it is set-based, we use a transformer to handle arbitrary input set sizes. Details of these networks are provided in the appendix D.

The denoising functions $\{\epsilon_R(\boldsymbol{p}, \boldsymbol{g}, t)\}_{t=0:T}$ represent the score of a sequence of T individual distributions, $\{q^t_R(\boldsymbol{p} \mid \boldsymbol{g})\}_{t=0:T}$, transitioning from $q^0_R(\boldsymbol{p} \mid \boldsymbol{g}) = q_R(\boldsymbol{p} \mid \boldsymbol{g})$ to $q^T_R(\boldsymbol{p} \mid \boldsymbol{g}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Therefore, to draw samples with the diffusion process, we initialize a sample $\boldsymbol{p}_T$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (*i.e.* a sample from $q^T_R(\cdot)$). We then use a reverse diffusion transition kernel to construct a simple $\boldsymbol{p}_{t-1}$ from distribution $q^{t-1}_R(\cdot)$, given a sample p_t from $q^t_R(\cdot)$. This reverse diffusion kernel corresponds to:

$$\boldsymbol{p}_{t-1} = B_t(\boldsymbol{p}_t - C_t\epsilon_R(\boldsymbol{p}_t, \boldsymbol{g}, t) + D_t\xi), \quad \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (1)$$

where B_t, C_t, and D_t are all constant terms and $\epsilon_R(\boldsymbol{p}_t, \boldsymbol{g}, t)$ is our learned denoising function. The final generated sample, $\boldsymbol{p}_0$ corresponds to a sample from $q^0_R(\cdot) = q_R(\cdot)$ and is our final generated sample.

[3] In practice, we always constrain the value range of p_i to be within $[0, 1]^*$; therefore, this distribution is properly defined.

B. Pose Generation via Compositional Diffusion Models

In a single diffusion model, we can generate new samples from the learned distribution by sampling an initial pose $\boldsymbol{p}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then repeatedly applying the learned transition kernel, sequentially sampling objects $q_R^{t-1}(\cdot)$ until we reach $\boldsymbol{p}_0$, which is a sample from the desired distribution.

However, in the composed diffusion model setting, our target distribution is defined by an entire factor graph, and we wish to sample from a sequence of product distributions $\{\prod_{R\in G} q_R^t(\boldsymbol{p} \mid \boldsymbol{g})\}_{t=0:T}$ starting from an initial sample $\boldsymbol{p}_T$ drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. For brevity, we refer to $\prod_{R\in G} q_R^t(\boldsymbol{p} \mid \boldsymbol{g})$ as $q_{prod}^t(\boldsymbol{p} \mid \boldsymbol{g})$[4]. While it is tempting to use the reverse diffusion kernel in Equation 1 to transition to each distribution $q_{prod}^t(\boldsymbol{p} \mid \boldsymbol{g})$, we do not have access to the score function for this distribution [6].

We can instead transition between distributions by using annealed MCMC , where we essentially use the composite score function $\sum_R \epsilon_R(\boldsymbol{p}_t, \boldsymbol{g}, t)$ across factors as a gradient for various MCMC transition kernels to transition from $q_{prod}^t(\boldsymbol{p} \mid \boldsymbol{g})$ to $q_{prod}^{t-1}(\boldsymbol{p} \mid \boldsymbol{g})$. Du et al. [6] suggested a set of MCMC kernels to use in this process, which requires substantial extra work to find hyperparameters to enable accurate sampling. We observe (and proved in Appendix A) that a very simple variant of the ULA sampler can be implemented by using the same parameters as the reverse kernel in Equation 1 on the composite score function

$$\boldsymbol{p}_t' = \boldsymbol{p}_t - C_t \sum_R \epsilon_R(\boldsymbol{p}_t, \boldsymbol{g}, t) + D_t \xi, \quad \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where C_t and D_t correspond to the same constant terms previously defined. Note, however, that this is not the reverse sampling kernel needed to transition directly from $q_{prod}^t(\boldsymbol{p} \mid \boldsymbol{g})$ to $q_{prod}^{t-1}(\boldsymbol{p} \mid \boldsymbol{g})$ as this requires a different score function [6], but rather a MCMC sampling step for $q_{prod}^t(\boldsymbol{p} \mid \boldsymbol{g})$.

Our variant of the ULA sampler allows for a simple implementation for sampling from composed diffusion models. We can directly compute a composite score function and apply the reverse diffusion step with this score function at a fixed noise level as running one step of MCMC at a distribution $q_{prod}^t(\cdot)$ Therefore, we can start with a random sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and then repeatedly run M reverse diffusion step at each noise level to generate a final sample from $q_{prod}^0(\cdot)$.

C. Abstract Relationship Generation via Program Induction

Now that we have a module for going from a ground graph description into object poses, we further generate graph descriptions $\mathcal{G}$ using a large language model in this section. As shown in Figure 4, our approach involves a two-stage generative process using the LLM. At the training stage, we synthesize a *task family rule-based program* that captures patterns for object arrangements. This program comprises

[4]Here we used a simplified notation. Each relation $g \in G$ is actually a tuple of $\big(R, (o_1, \ldots, o_{k_R})\big)$. It selects a particular relation R and s subset of objects to which to apply it. Thus, the corresponding distribution q^g should be defined as $q_R(\boldsymbol{p}^g \mid \boldsymbol{g}^g) = q_R(p_1, \ldots, p_{k_R} \mid g_1, \ldots, g_{k_R})$. For brevity, we used $\prod_{R\in G} q^R(\boldsymbol{p} \mid \boldsymbol{g})$ to denote the composite distribution.

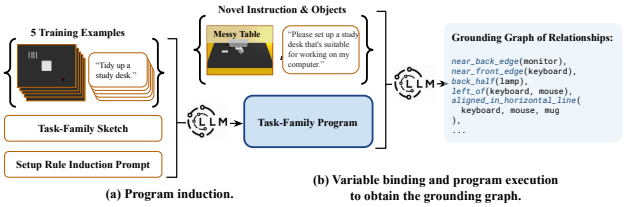


Fig. 4. Abstract relationship generation through rule induction involves two phases. Initially, in the program induction phase, we employ an LLM to create a "setup" rule-based program from a few training examples and a high-level task-family sketch defined by humans. This program contains rules and patterns for various subproblems, but it has unbound variables (i.e., the actual objects and instructions are not specified yet). In the second phase, with a new instruction and a list of test objects, the LLM binds these variables to the induced program to create an executable Python program. This executable Python program is then used to generate the final set of abstract spatial relationships as a ground graph.

Fig. 5. Example process of using an LLM to instantiate a program sketch. Sub-figure (a) presents an example of the initial program sketch. We provide this program sketch, along with five training instances, to the LLM. The LLM then creates a rule-based program, summarizing the common patterns in the form of code comments and/or templates, but with unbound variables, as illustrated in (b). Finally, given new objects and instructions, the LLM binds these variables to the induced program and generates an executable Python program. This program is then used to generate the object grounding graph. An example of an executable Python program with variable bindings during inference time is depicted in (c).

detailed comments and code templates with unbound variables, as the specific objects and instructions are not yet defined. The synthesis process uses a small set of training examples, $\mathcal{D}$, to instantiate a provided *sketch*. During the inference stage, given a new set of objects $\mathcal{O}$ and a task instruction *desc*, we bind the actual variables to the induced program, producing an *executable Python program*. This script is then executed to generate the ground relationship graph.

Recall that the training examples only record object poses, these numerical values cannot be effectively used by the LLM to infer abstract relationship patterns. Therefore, we need to "translate" the pose information in training examples $\mathcal{D}$ into

a set of active abstract relationships. To do this, for each data point in the training set $(desc_d, \mathcal{O}_d, \mathcal{P}_d) \in \mathcal{D}$, we compute the set of primitive relationships that hold by applying the classifier h_R for each abstract relationship type $R \in \mathcal{R}$, to each subset of objects in $\mathcal{O}_d$. We use a string encoding of all active relationships to describe each example scene.

Program Sketch. Our relationship generation procedure adopts a hierarchical program synthesis framework. It is based on a human-defined, task-family-specific *sketch*, with an example shown in Figure 5. Each sketch includes several functions with meaningful names, signatures, and descriptions, organized in a sequence that outlines the steps to solve the complex task. The sketch is crucial for generating abstract relationships. It acts as a guide that helps the LLM generate object relationships step-by-step. With this strong guidance, the LLM can create task-family-specific rule-based program from just five examples. Instead of directly regressing the final relationship proposals on these five training examples, we use the examples mainly to instantiate the subroutines in the sketch and to write detailed comments. Such high-level sketches decompose the generation task into four types of subroutines: 1) extracting the task relevant information from the instruction (*e.g.*, number of diners), 2) categorizing objects into groups (*e.g.*, finding all input devices to a computer), 3) generating arrangements for the objects within each group(*e.g.*, all input devices), and 4) generating arrangements for objects among groups. These subroutines usually require only a single "step" of reasoning, such as directly extracting a number from the instruction or generating object arrangements for a smaller subset of the objects. In this work, we represent the sketch in Python-like syntax, but in future work, it would be important, like in Zelikman et al. [35], to accept sketches in natural language and convert them into Python or a simpler language such as their Parsel. In practice, we follow the four subroutines and manually decompose the prediction task into 7-10 functions; we provide the sketches for the three task families we study in this work in the appendix B. To add a new task family, such as setting up for a Scrabble game, it is only necessary to provide a sketch and a few example set-ups. Furthermore, it is crucial to note that this high-level sketch does not contain concrete implementations for any subroutines; we will prompt the LLM, given the training examples, to solve each subproblem.

Program Induction. Given the sketch and training examples, we query the LLM to generate a rule-based program. This program comprising rules and patterns for each subproblem, but with unbound variables (i.e., the actual objects and instructions are not yet specified). Specifically, the program encapsulates the rules associated with each function in the form of code comments and templates. For example, it may include instructions for identifying the key object in a study desk setup task. During the rule induction stage, we input the textual descriptions of the training examples, which are detailed in terms of their spatial relations, to the LLM. We then prompt the LLM to summarize the patterns in these examples into either comments or code templates with unbound variables. After the rule induction stage, we obtain a program with comments that summarizes the patterns derived from the training examples. Examples of a sketch and a resulting LLM-generated program are illustrated in Figure 5a and b.

Variable Binding and Program Execution. At performance time, given a new set of objects $\mathcal{O}_d$, task description *descr*, we first use the LLM to bind these variables to the induced program. This process incorporates the contextual information to generate an executable Python program. Then, we execute the program to return the set of functional abstract spatial relationships. Specifically, we prompt the LLM to read the test instructions *desc* and the object names in $\mathcal{O}$, and fill in scene-specific details for each program. An example of an executable program with variable bindings, `extract_main_devices`, is shown in Figure 5c: the LLM generates a list of key objects for the new scene and incorporates it into a return statement. To generate the final set of relationships among objects, we simply execute this program with variable bindings, which now include the scene-specific comments and implementation, as well as the task description to the LLM, to generate the list of relationships.

Empirically, we have found that LLM-generated relationships often suffer from inconsistency or incompleteness. Inconsistency arises when the LLM proposes several relationships that cannot be simultaneously satisfied. Incompleteness emerges when the LLM does not provide sufficient information to predict the placement of a target object. To address these issues, we employ an iterative self-reflection process to refine the initial programs generated by the LLM. At each refinement iteration, the input to the LLM includes the task inputs, the fully instantiated programs generated by the rule instantiation step, and its execution result, which is a set of abstract spatial relationships. We then instruct the LLMs to compose two language summaries: one describing the desired scene configuration based only on the task inputs, and the other describing the scene configuration as inferred from the program execution results. Next, we prompt LLM to identify inconsistency and incompleteness between two language summaries and fix the instantiated setup rules. We include all detailed instructions and prompts to the LLM in the appendix B.

IV. Experiments

We evaluated our method in three scene types: study desks, dining tables, and coffee tables. They involve different types of objects, different aesthetic patterns, and different types of human needs (*e.g.*, using a laptop vs. paper and pencil, formal dining vs. casual dining). They also progressively increase in scene complexity: the study desk has the fewest objects and important relationships, and the dining table features the most. We generated each scene with household items that commonly appear in the particular scene type. There are in total 15 distinct tasks created for each environment: 5 for training and 10 for evaluation. We include the training and testing examples

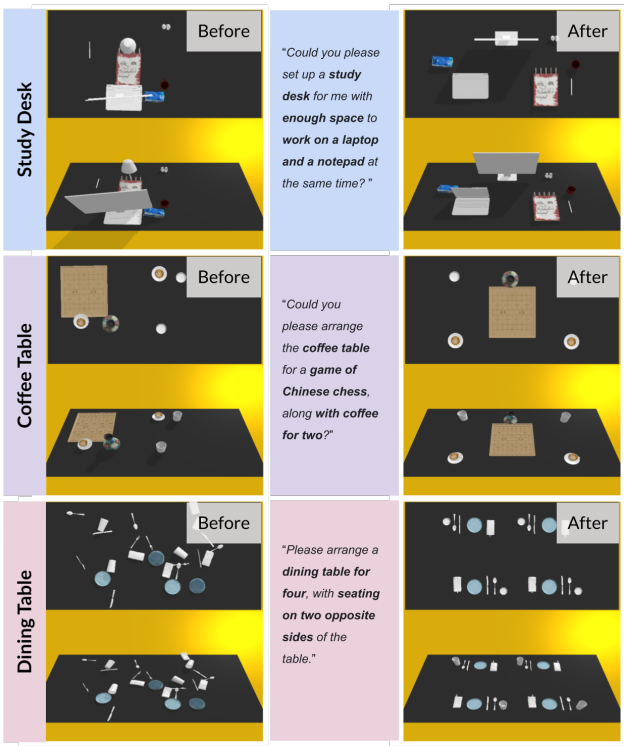


Fig. 6. Illustration of three scene types (before and after). The after-scene configuration is generated by SetItUp.

in the appendix F. Figure 6 shows examples of each scene type and the final object arrangements generated by SetItUp.

We compare SetItUp to several baselines according to the following criteria: How do different models perform across various aspects of the arrangement task? How does our neuro-symbolic design compare with monolithic neural networks or straightforward large language model predictions? Finally, how do different models generalize to novel scenarios involving unseen instructions and objects?

A. Baselines

We have implemented two baselines and an ablation variation of our model. Implementation details are included in the appendix E.

End-to-End diffusion model: we implement an end-to-end diffusion model for language-conditioned object arrangement, inspired by StructDiffusion [18]. We trained the model using a combination of the same synthetic data and 15 tidy scene examples (5 per scene type) as our model. At test time, it directly generates object poses based on the language instructions and object shapes.

Direct LLM Prediction: This baseline directly leverages a large language model (LLM) to predict test object poses, inspired by Tidybot [30]. It is conditioned on 15 given examples (5 for each scene type), as well as the object categories and shapes.

LLM-Diffusion: This model is a simplified variant of our approach, omitting the program sketch, rule induction, and iterative self-reflection mechanisms. It directly prompts an LLM to generate abstract object relationships and employs the same diffusion-model-based inference for grounding.

B. Evaluation Metrics

A *functional object arrangement* must meet several criteria: physical feasibility (i.e., being collision-free), functionality (i.e., serving the intended purpose as per human instructions), and overall convenience and aesthetic appeal (i.e., neat arrangement with alignments and symmetries). While physical feasibility and functionality can be objectively evaluated using rule-based scripts, the subjective nature of convenience and aesthetic appeal precludes such fixed evaluations. Therefore, we employ both rule-based auto-grading and human experiments to evaluate models.

Physical feasibility: We define the physical feasibility score of an arrangement as the proportion of collision-free objects in the final layout using a 2D collision detector.

Basic functionality: This measures the proportion of functional relationships satisfied in the final scene configuration. For each scene, we use a set of manually defined rules to generate the basic functional relationships, including constraints that key objects (*e.g.*, the utensil in a dining table setting) are within reach of the user (The user position(s) is predetermined), unobstructed, and arranged to fulfill their intended function (*e.g.*, left-handedness).

Aesthetics, convenience, and other preferences: We conduct a human evaluation with 20 graduate students. Each participant evaluates 30 scenes generated by either our method or a baseline, assigning a score from 1 to 5 based on a set of criteria. Table II presents the grading criteria. We report the average score for each method across every scene type. For ease of evaluation, we rendered the 2D object poses using PyBullet [2], a physical simulator. It is important to note that the 3D models of the objects were not used in any of the methods for proposing object poses.

C. Results

Table III shows the overall performance of all models across three scene types and three evaluation metrics. Overall, our model SetItUp consistently outperforms all baselines, achieving a nearly perfect collision-free rate (indicating physical feasibility) and recovering over 80% of the relationships related to basic functionalities. Additionally, it consistently scores high across all human evaluations, indicating its success in achieving functional and aesthetically pleasing arrangements. The direct LLM prediction baseline and our model variant (LLM-Diffusion) exhibit similar rankings across all three evaluation metrics. We explore the behavior of the direct LLM prediction baseline and compare it with SetItUp in detail in the remaining part of this section. The LLM-Diffusion baseline shows slightly better performance than the direct LLM prediction baseline in tasks with fewer objects and constraints (i.e., the

TABLE II
GRADING GUIDELINES FOR HUMAN EVALUATION.

Points	Grading Criteria
1	Functionally inadequate. The setup does not serve the intended purpose in the user instruction.
2	Partial functional but inconvenient. The setup can somewhat serve the intended purpose, but the arrangement makes the intended activity inconvenient since it does not conform to the user's habits or social norms.
3	Functional, but cluttered. Key items are arranged properly to serve the intended purpose, but the space is overcrowded and lacks efficient organization.
4	Fully functional, lacks aesthetic appeal. Functionality and accessibility are good, but the arrangement lacks in visual harmony and alignment.
5	Fully functional, and aesthetically pleasing. All items are well-placed, easy to access, and efficiently organized. Furthermore, the setup is visually appealing with proper alignment and symmetry.

study desk) and underperforms in tasks with many objects (i.e., the dining table). This implies that our approach of program-structured prompting and self-reflection-based refinement improves the overall system performance. A notable failure mode of the LLM-Diffusion variant is its frequent generation of conflicting relationships among objects, subsequently leading to collisions in the final scene configuration, and therefore, resulting in low human evaluation scores. The End-to-End diffusion model is the worst-performing method across all metrics and scene types, primarily due to its inability to leverage additional commonsense knowledge (e.g., from LLMs) during inference. This results in poor generalization to new tasks with only a limited number of training examples. We will further examine its generalization behavior in Section IV-D.

Figure 7 illustrates the qualitative scene arrangements generated by different methods across all three scene types. By comparing the behavior of various methods, we aim to derive insights into the effectiveness of abstract relationships and compositional generative models.

Direct LLM prediction vs. Ours: the importance of abstract relationships. A notable observation in Table III is that the direct LLM prediction baseline typically achieves a higher physical feasibility score compared to its functionality score. This pattern becomes more apparent when compared to the LLM-Diffusion variant of our model. By contrast, the LLM-Diffusion variant maintains a consistent functionality score, even though its physical feasibility score decreases with an increase in the number of objects. Examining the scene configurations in Figure 7 more closely, we observe that the direct LLM prediction baseline excels at creating scenes where objects are well-separated and aesthetically arranged but often fails to fulfill functional requirements. For instance, in the dining table task, while all plates and utensils are neatly placed, the setup does not functionally accommodate two diners. This underscores the significance of leveraging abstract relationships as intermediate interfaces between large language models and the physical scene; it helps the model to better reason about important spatial relationships for the specific task.

End-to-End diffusion model vs Ours: the importance of compositional diffusion models. Let's now compare the performance of the end-to-end diffusion model with our framework. It's important to note that both models are trained on the same dataset, including 30,000 examples of synthetically generated single-relationship arrangements and 15 scenes of human-labeled tidy arrangements. Looking at the dining table task in Figure 7, we find that the end-to-end diffusion model manages to generalize to this novel test scene to a certain extent but fails to place all objects in a physically feasible manner. This suggests that it struggles with reusing its training data on individual relationship types while generating the global scene arrangements.

A critical insight into the problem of *tidy object arrangement* is that generating synthetic data for a single relationship type is relatively inexpensive, whereas there is a significant lack of scene-level annotations of paired instructions and object arrangements. As a result, a compositional learning framework is preferred over a monolithic one, as it can learn individual relationships from synthetically generated single-relationship examples and then compose them at test time. This explicit compositional structure demonstrates a stronger performance compared to the end-to-end diffusion baseline, which naively mixes training data for single relationships and scene-level arrangements directly.

D. Generalization

We further break down our quantitative results to analyze generalization across different dimensions of the problem: generalization to novel instructions, as well as to larger scenes with more objects and relationships.

Generalization to novel instructions. Recall that our dataset comprises 5 training examples and 10 test examples for each scene type. We manually label each test example as either a "seen instruction" (i.e., it is similar to an instruction in the training examples) or a "novel instruction." We then compute the average human evaluation score for both groups of instructions. Figure 8 presents the results.

Our SetItUp achieved the highest scores in both categories, with no significant performance drop when generalizing to novel instructions. By contrast, the end-to-end diffusion model demonstrated the weakest generalization. Its monolithic model structure, heavily reliant on the coverage of training data, limits its generalization to new instructions. The direct LLM prediction baseline and our ablation variant experienced similar performance drops on novel instructions for study desks and coffee tables. We attribute the direct LLM prediction baseline's poor generalization to novel instructions to its limited capability to reason about important functional relationships within the scene, as discussed in the previous section. Our full model also surpasses the LLM-Diffusion baseline, highlighting

TABLE III
HOLISTIC EVALUATION OF DIFFERENT METHODS ACROSS TASK FAMILIES.

Model	Physical Feasibility (%)			Functionality (%)			Overall Human Judgement (1-5)		
	Study	Coffee	Dining	Study	Coffee	Dining	Study	Coffee	Dining
End-to-End Diffusion Model	$37.7_{\pm 32.1}$	$39.6_{\pm 17.3}$	$7.03_{\pm 9.59}$	$49.8_{\pm 21.5}$	$53.5_{\pm 13.1}$	$37.5_{\pm 11.1}$	$1.89_{\pm 0.834}$	$1.91_{\pm 0.650}$	$1.58_{\pm 0.433}$
Direct LLM	$63.1_{\pm 27.3}$	$58.6_{\pm 22.5}$	$63.6_{\pm 20.1}$	$59.7_{\pm 11.7}$	$53.7_{\pm 22.3}$	$56.4_{\pm 15.3}$	$3.01_{\pm 0.901}$	$2.84_{\pm 1.060}$	$2.33_{\pm 0.416}$
LLM-Diffusion (Our ablation)	$69.4_{\pm 19.2}$	$41.0_{\pm 20.1}$	$26.7_{\pm 13.8}$	$69.8_{\pm 11.6}$	$44.3_{\pm 10.4}$	$46.5_{\pm 17.8}$	$3.67_{\pm 0.934}$	$3.16_{\pm 0.657}$	$1.87_{\pm 0.652}$
SetItUp (Ours)	$\mathbf{95.0}_{\pm 10.0}$	$\mathbf{98.1}_{\pm 3.83}$	$\mathbf{95.8}_{\pm 6.72}$	$\mathbf{94.1}_{\pm 6.04}$	$\mathbf{84.4}_{\pm 13.5}$	$\mathbf{91.5}_{\pm 13.8}$	$\mathbf{4.49}_{\pm 0.343}$	$\mathbf{4.47}_{\pm 0.211}$	$\mathbf{4.79}_{\pm 0.190}$

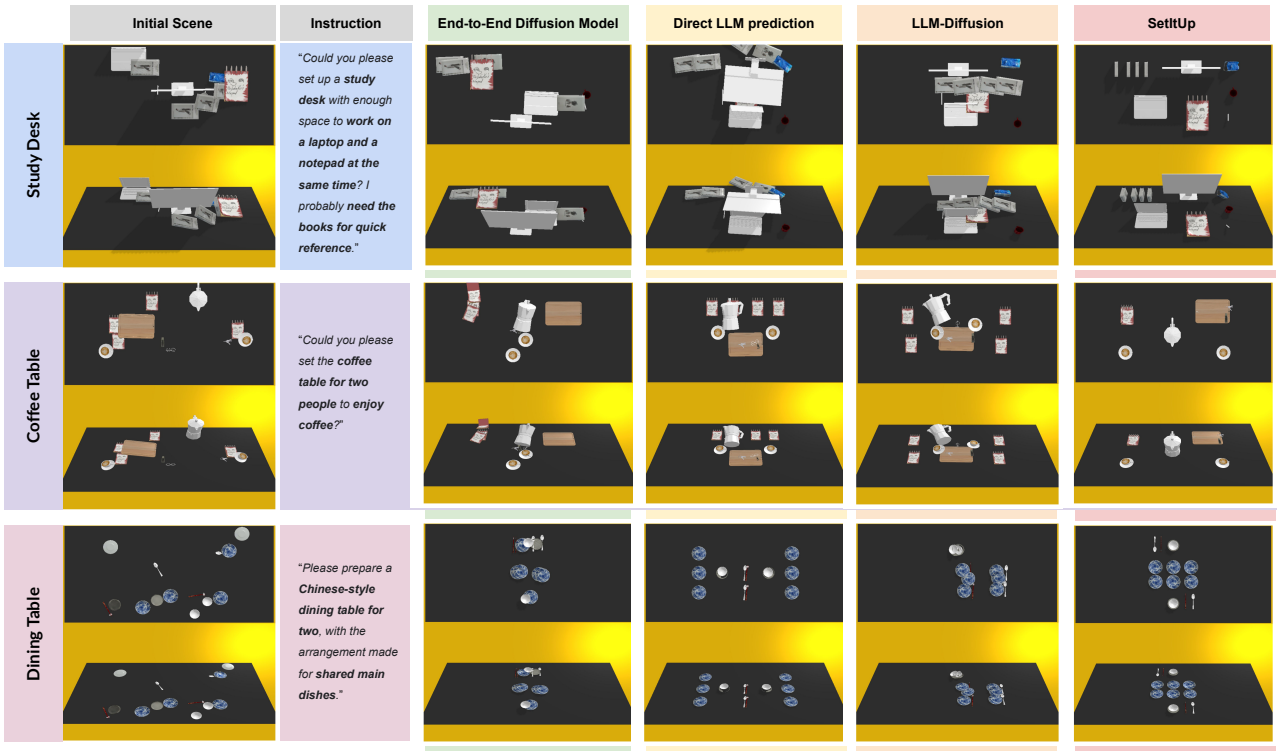


Fig. 7. Illustrations of the final arrangements generated by our method and the baselines. Our model consistently generates more physically plausible, functional, and aesthetically pleasing object arrangements.

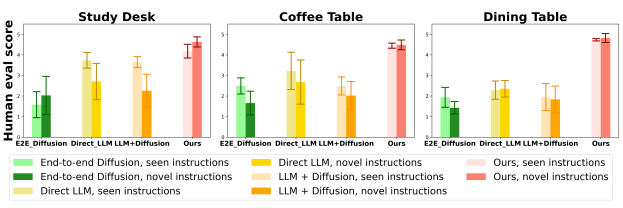


Fig. 8. Evaluation on the generalization to novel instructions. Details on the seen-unseen splits are provided in the appendix. Our model shows the least amount of performance drop when generalizing to novel instructions.

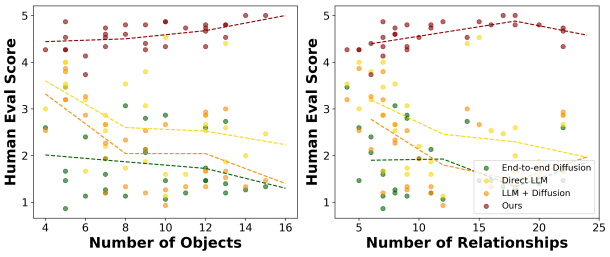


Fig. 9. Evaluation of the model performance on scenes with different numbers of objects and relationships. The dashed lines depict the mean human evaluation scores for each method. As the number of objects and active relationships increases, our model consistently produces layouts that satisfy human evaluators, while all baselines have noticeable performance declines.

the effectiveness of our rule induction steps and self-reflection-based refinements for improved consistency and completeness.

Generalization to more objects and complex scenes. Figure 9 illustrates how the human evaluation score varies with different numbers of objects and functional relationships. Our SetItUp (represented by dark red dots) achieves high scores

consistently across the spectrum of scene complexity. By contrast, we see a noticeable performance drop for all other methods as the scene becomes more complex.

We believe that the strong scalability of our model comes from two important designs of the system. First, in the abstract relationship generation stage, our rule induction step can infer abstract rules about arrangement patterns and, therefore, generalize better to larger scenes. Second, the compositional design of our pose diffusion models allows for aggressive generalization to scenes with a greater number of objects and relationships, due to the composition achieved by explicitly summing up individually trained energy functions. This is consistent with the findings from Yang et al. [33] on how the compositional diffusion model compared to monolithic models in terms of generalization.

V. Related Work

A. Object Rearrangement

The literature on robotic object rearrangement is extensive, with many studies assuming that a goal arrangement is provided, detailing the precise object positions [3, 5, 8, 19, 20, 23, 24, 34, 36]. In many cases, they focus on the detailed task and motion planning (TAMP), which comes with well-defined locations or using goal images as references. Another class of works involves interpreting language instructions to determine object placement by specified relationships [7]. ALFRED introduced a multi-step, language-guided rearrangement task [1, 22], inspiring solutions that merge high-level skills. However, these methods function at a basic level, focusing solely on placing objects in the correct receptacles without considering the necessary functional and spatial relationships among objects for effective tabletop arrangements. Our method differs from past approaches that require detailed goal specifications or concrete language instructions. Instead, we work with under-specified human instructions, inferring the necessary abstract relationships between objects within a scenario and determining their precise arrangements to fulfill them.

B. Functional Object Arrangement

Organizing objects on a tabletop to reach neatly functional arrangements is distinct from mere geometric rearrangement. It requires that objects not only appear orderly but also fulfill their intended purpose, often involving arranging like-function objects together.

Several benchmarks [25, 29, 12] address room tidying and common-sense notions of neatness at a basic level, focusing on correctly placing items in designated containers to satisfy semantic and aesthetic criteria [30, 21]. However, these do not account for the spatial relationships among the objects crucial for both functional and visual neatness on a tabletop. Previous data-driven research, such as Structformer [17] and StructDiffusion[18] focused on predicting object arrangements from vague instructions. They trained a single transformer and a diffusion model respectively to propose the placements of the objects. Other approaches trained a single model to predict the tidiness score [13, 28] or the gradient towards an orderly configuration [31, 13]. These methods often rely on extensive datasets and struggle to accommodate new object combinations or specific user preferences. Advancing towards a more generalizable object arrangement proposal, DALL-E-BOT [14] employs a pre-trained VLM to propose common-sense arrangements for open-category objects. However, its sole reliance on a single VLM for zero-shot layout generation from object types prevents it from accommodating user preferences and intended functional arrangements. Moreover, it overlooks object geometries, essential for creating physically feasible arrangements.

Our work differs by enabling zero-shot performance with novel items in various contexts through our symbolic reasoning powered by an LLM and compositional diffusion model grounding, eliminating the need for expert demonstrations.

C. Knowledge Extraction from LLM

Recent advancements have led to the development of various large language models (LLMs) capable of encoding extensive common sense knowledge and aiding in robot decision-making tasks [16]. These models have shown promise in domains such as task planning for household robotics [11, 10, 30]. Huang et al. demonstrated that iterative prompt augmentation with LLMs improves task plan generation [10]. Meanwhile, the SayCan approach integrates affordance functions into robot planning, facilitating action feasibility assessment when processing natural language service requests like “make breakfast” [11]. LLM-GROP uses LLM to suggest object placements through prompts [4]

Despite these advancements, LLMs often fails to capture the spatial understanding when organizing spaces, especially as object count and environmental complexity increase. Some research has tried to mitigate this issue by suggesting object placements using LLMs and then grounding these suggestions using vision-based models trained on orderly configurations [32]. However, this method typically underperforms with novel object arrangements due to its reliance on a single visual model. Our approach differs markedly, as we employ compositional diffusion models to anchor symbolic object relationships accurately. The inherent compositional capabilities of these models offer a principled method for optimizing relational sets, enhancing our ability to handle complex geometric arrangements effectively.

VI. Conclusion

We have proposed SetItUp, a neuro-symbolic model for compositional commonsense object arrangements. In order to achieve strong data efficiency and generalization, the design of SetItUp is based on two important principles. First, we leverage large language models as a commonsense knowledge base to generate arrangement plans in an abstract language, based on simple geometric relationships. Second, in order to find global scene arrangements satisfying all proposed relationships, we use a compositional diffusion model as a continuous constraint satisfaction problem solver. We show that by

composing individually trained diffusion models on synthetic data at test time, our system directly generalizes scenes with many objects and many relationships. This algorithm addresses the data scarcity issue of large scene arrangements and has strong extensibility. Both rule-based and human evaluations show that our model is capable of generating more physically feasible, functional, and aesthetic object placements compared to both pure LLM-based and end-to-end neural generative model baselines.

ACKNOWLEDGMENTS

This research is supported in part by the National Research Foundation (NRF), Singapore and DSO National Laboratories under the AI Singapore Program (AISG Award No: AISG2-RP-2020-016), NSF grant 2214177, AFOSR grant FA9550-22-1-0249, ONR MURI grant N00014-22-1-2740, and ARO grant W911NF-23-1-0034. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of NRF Singapore.

REFERENCES

[1] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A Persistent Spatial Semantic Representation for High-Level Natural Language Instruction Execution. In *CoRL*, 2022.

[2] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning, 2016–2019.

[3] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object Rearrangement Using Learned Implicit Collision Functions. In *ICRA*, 2021.

[4] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and Motion Planning with Large Language Models for Object Rearrangement. In *IROS*, 2023.

[5] Danny Driess, Jung-Su Ha, and Marc Toussaint. Learning to Solve Sequential Physical Reasoning Problems from a Scene Image. *IJRR*, 40(12-14):1435–1466, 2021.

[6] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC. In *ICML*, 2023.

[7] Nikolaos Gkanatsios, Ayush Jain, Zhou Xian, Yunchu Zhang, Christopher Atkeson, and Katerina Fragkiadaki. Energy-Based Models as Zero-Shot Planners for Compositional Scene Rearrangement. In *RSS*, 2023.

[8] Walter Goodwin, Sagar Vaze, Ioannis Havoutis, and Ingmar Posner. Semantically Grounded Object Matching for Robust Robotic Scene Rearrangement. In *ICRA*, 2022.

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020.

[10] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *ICML*, 2022.

[11] Brian Ichter, Fei Xia, and Karol Hausman et al. Do as I Can, Not as I Say: Grounding Language in Robotic Affordances. In *CoRL*, 2022.

[12] Yash Kant, Arun Ramachandran, Sriram Yenamandra, Igor Gilitschenski, Dhruv Batra, Andrew Szot, and Harsh Agrawal. Housekeep: Tidying Virtual Households Using Commonsense Reasoning. In *ECCV*, 2022.

[13] Ivan Kapelyukh and Edward Johns. My House, My Rules: Learning Tidying Preferences with Graph Neural Networks. In *CoRL*, 2022.

[14] Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, 2023.

[15] Ranjay Krishna, Yuke Zhu, and Oliver et al. Groth. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *IJCV*, 123:32–73, 2017.

[16] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM CSUR*, 55(9):1–35, 2023.

[17] Weiyu Liu, Chris Paxton, Tucker Hermans, and Dieter Fox. StructFormer: Learning Spatial Structure for Language-Guided Semantic Rearrangement of Novel Objects. In *ICRA*, 2022.

[18] Weiyu Liu, Yilun Du, Tucker Hermans, Sonia Chernova, and Chris Paxton. StructDiffusion: Language-Guided Creation of Physically-Valid Structures using Unseen Objects. In *RSS*, 2023.

[19] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kPAM: Keypoint Affordances for Category-Level Robotic Manipulation. In *ISRR*, 2019.

[20] Ahmed Hussain Qureshi, Arsalan Mousavian, Chris Paxton, Michael C. Yip, and Dieter Fox. NERP: Neural Rearrangement Planning for Unknown Objects. In *RSS*, 2021.

[21] Gabriel Sarch, Zhaoyuan Fang, Adam W Harley, Paul Schydlo, Michael J Tarr, Saurabh Gupta, and Katerina Fragkiadaki. TIDEE: Tidying Up Novel Rooms Using Visuo-Semantic Commonsense Priors. In *ECCV*, 2022.

[22] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *CVPR*, 2020.

[23] Anthony Simeonov, Yilun Du, Beomjoon Kim, Francois Hogan, Joshua Tenenbaum, Pulkit Agrawal, and Alberto Rodriguez. A Long Horizon Planning Framework for Manipulating Rigid Pointcloud Objects. In *CoRL*, 2021.

[24] Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. SE(3)-Equivariant Relational Rearrangement with Neural Descriptor Fields. In *CoRL*, 2023.

[25] Andrew Szot, Alexander Clegg, Eric Undersander,

Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training Home Assistants to Rearrange Their Habitat. In *NeurIPS*, 2021.

[26] U.S. Bureau of Labor Statistics. American Time Use Survey, 2023.

[27] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Comput.*, 23(7): 1661–1674, 2011.

[28] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. LEGO-Net: Learning Regular Rearrangements of Objects in Rooms. In *CVPR*, 2023.

[29] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual Room Rearrangement. In *CVPR*, 2021.

[30] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. TidyBot: Personalized Robot Assistance with Large Language Models. In *IROS*, 2023.

[31] Mingdong Wu, Fangwei Zhong, Yulong Xia, and Hao Dong. TarGF: Learning Target Gradient Field to Rearrange Objects Without Explicit Goal Specification. In *NeurIPS*, 2022.

[32] Yiqing Xu and David Hsu. How to Tidy Up a Table: Fusing Visual and Semantic Commonsense Reasoning for Robotic Tasks with Vague Objectives. *arXiv preprint arXiv:2307.11319*, 2023.

[33] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional Diffusion-Based Continuous Constraint Solvers. In *CoRL*, 2023.

[34] Wentao Yuan, Chris Paxton, Karthik Desingh, and Dieter Fox. SORNet: Spatial Object-centric Representations for Sequential Manipulation. In *CoRL*, 2022.

[35] Eric Zelikman, Qian Huang, Gabriel Poesia, Noah Goodman, and Nick Haber. Parsel: Algorithmic Reasoning with Language Models by Composing Decompositions. In *NeurIPS*, 2023.

[36] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter Networks: Rearranging the Visual World for Robotic Manipulation. In *CoRL*, 2021.