VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models

Wenlong Huang[1], **Chen Wang**[1], **Ruohan Zhang**[1], **Yunzhu Li**[1,2], **Jiajun Wu**[1], **Li Fei-Fei**[1]
[1]Stanford University [2]University of Illinois Urbana-Champaign

Abstract: Large language models (LLMs) are shown to possess a wealth of actionable knowledge that can be extracted for robot manipulation in the form of reasoning and planning. Despite the progress, most still rely on pre-defined motion primitives to carry out the physical interactions with the environment, which remains a major bottleneck. In this work, we aim to synthesize robot trajectories, i.e., a dense sequence of 6-DoF end-effector waypoints, for a large variety of manipulation tasks given an *open-set of instructions* and an *open-set of objects*. We achieve this by first observing that LLMs excel at inferring affordances and constraints given a free-form language instruction. More importantly, by leveraging their code-writing capabilities, they can interact with a vision-language model (VLM) to compose 3D value maps to ground the knowledge into the observation space of the agent. The composed value maps are then used in a model-based planning framework to *zero-shot* synthesize closed-loop robot trajectories with robustness to dynamic perturbations. We further demonstrate how the proposed framework can benefit from online experiences by efficiently learning a dynamics model for scenes that involve contact-rich interactions. We present a large-scale study of the proposed method in both simulated and real-robot environments, showcasing the ability to perform a large variety of everyday manipulation tasks specified in free-form natural language. Videos and code at voxposer.github.io.

Keywords: Manipulation, Large Language Models, Model-based Planning

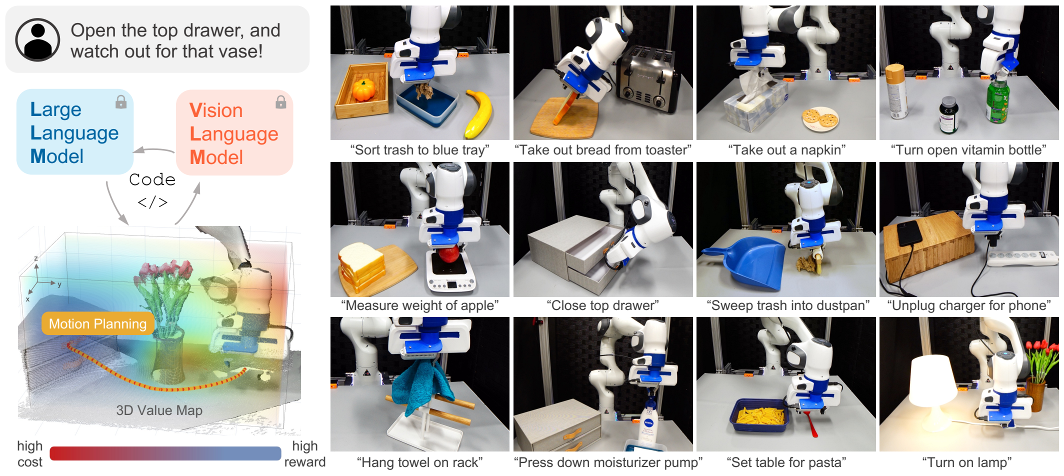

Open the top drawer, and watch out for that vase!
Large Language Model
Vision Language Model
Code </>
Motion Planning
3D Value Map
high cost
high reward
"Sort trash to blue tray"
"Take out bread from toaster"
"Take out a napkin"
"Turn open vitamin bottle"
"Measure weight of apple"
"Close top drawer"
"Sweep trash into dustpan"
"Unplug charger for phone"
"Hang towel on rack"
"Press down moisturizer pump"
"Set table for pasta"
"Turn on lamp"

Figure 1: **VoxPoser** extracts language-conditioned **affordances** and **constraints** from LLMs and grounds them to the perceptual space using VLMs, using a code interface and without additional training to either component. The composed map is referred to as a 3D value map, which enables **zero-shot** synthesis of trajectories for large varieties of everyday manipulation tasks with an **open-set of instructions** and an **open-set of objects**.

Correspondence to Wenlong Huang <wenlongh@stanford.edu>.

1 Introduction

Language is a compressed medium through which humans distill and communicate their knowledge and experience of the world. Large language models (LLMs) have emerged as a promising approach to capture this abstraction, learning to represent the world through projection into language space [1–4]. While these models are believed to internalize generalizable knowledge as text, it remains a question about how to use it to enable embodied agents to physically *act* in the real world.

We look at the problem of grounding abstract language instructions (e.g., "set up the table") in robot actions [5]. Prior works have leveraged lexical analysis to parse the instructions [6–8], while more recently language models have been used to decompose the instructions into a textual sequence of steps [9–11]. However, to enable physical interactions with the environment, existing approaches typically rely on a repertoire of pre-defined motion primitives (i.e., skills) that may be invoked by an LLM or a planner, and this reliance on individual skill acquisition is often considered a major bottleneck of the system due to the lack of large-scale robotic data. The question then arises: how can we leverage the wealth of internalized knowledge of LLMs at the even fine-grained action level for robots, without requiring laborious data collection or manual designs for each individual primitive?

In addressing this challenge, we first note that it is impractical for LLMs to directly output control actions in text, which are typically driven by high-frequency control signals in high-dimensional space. However, we find that LLMs excel at inferring language-conditioned *affordances* and *constraints*, and by leveraging their code-writing capabilities, they can compose dense 3D voxel maps that ground them in the visual space by orchestrating perception calls (e.g., via CLIP [12] or open-vocabulary detectors [13–15]) and array operations (e.g., via NumPy [16]). For example, given an instruction "open the top drawer and watch out for the vase", LLMs can be prompted to infer: 1) the top drawer handle should be grasped, 2) the handle needs to be translated outwards, and 3) the robot should stay away from the vase. By generating Python code to invoke perception APIs, LLMs can obtain spatial-geometric information of relevant objects or parts and then manipulate the 3D voxels to prescribe reward or cost at relevant locations in observation space (e.g., the handle region is assigned high values while the surrounding of the vase is assigned low values). Finally, the composed value maps can serve as objective functions for motion planners to directly synthesize robot trajectories that achieve the given instruction [1] , without requiring additional training data for each task or for the LLM. An illustration diagram and a subset of tasks we considered are shown in Fig. 1.

We term this approach **VoxPoser** , a formulation that extracts affordances and constraints from LLMs to compose 3D value maps in observation space for guiding robotic interactions. Rather than relying on robotic data that are often of limited amount or variability, the method leverages LLMs for *open-world reasoning* and VLMs for *generalizable visual grounding* in a model-based planning framework that directly enables *physical* robot actions. We demonstrate its zero-shot generalization for *open-set* instructions with *open-set* objects for various everyday manipulation tasks. We further showcase how VoxPoser can also benefit from limited online interactions to efficiently learn a dynamics model that involves contact-rich interactions.

2 Related Works

Grounding Language Instructions. Language grounding has been studied extensively both in terms of intelligent agents [19–22] and of robotics [23, 6, 24, 25, 5, 7, 26], where language can be used as a tool for compositional goal specification [5, 27–33], semantic anchor for training multi-modal representation [12, 34, 35], or as an intermediate substrate for planning and reasoning [36–38, 9, 10, 39, 40]. Prior works have looked at using classical tools such as lexical analysis, formal logic, and graphical models to interpret language instructions [27, 7, 6, 26]. More recently, end-to-end approaches, popularized by successful applications to offline domains [41–43, 1], have been applied to directly ground language instructions in robot interactions by learning from data with

[1]The approach also bears resemblance and connections to potential field methods in path planning [17] and constrained optimization methods in manipulation planning [18].

language annotations, spanning from model learning [44], imitation learning [45, 46, 30, 47–54], to reinforcement learning [55–57]. Most closely related to our work is Sharma et al. [50], where an end-to-end cost predictor is optimized via supervised learning to map language instructions to 2D costmaps, which are used to steer a motion planner to generate preferred trajectories in a collision-free manner. In contrast, we rely on pre-trained language models for their open-world knowledge and tackle the more challenging robotic manipulation in 3D.

Language Models for Robotics. Leveraging pre-trained language models for embodied applications is an active area of research, where a large body of works focus on planning and reasoning with language models [9–11, 58, 31, 39, 59–72, 36, 73, 74]. To allow language models to perceive the physical environments, textual descriptions of the scene [39, 11, 59] or perception APIs [75] can be given, vision can be used during decoding [67] or can be directly taken as input by multi-modal language models [68, 2]. In addition to perception, to truly bridge the perception-action loop, an embodied language model must also know how to *act*, which typically is achieved by a library of pre-defined primitives. Liang et al. [75] showed that LLMs exhibit behavioral commonsense that can be useful for low-level control. Despite the promising signs, hand-designed motion primitives are still required, and while LLMs are shown to be capable of composing *sequential* policy logic, it remains unclear whether composition can happen at *spatial* level. A related line of works has also explored using LLMs for reward specification in the context of reward design [76], exploration [77–80], and preference learning [81]. For robotic applications, concurrent works explored LLM-based reward generation [82–88], among which Yu et al. [82] use MuJoCo [89] as a high-fidelity physics model for model predictive control. In contrast, we focus exclusively on grounding the reward generated by LLMs in the *3D observation space* of the robot.

Learning-based Trajectory Optimization. Many works have explored leveraging learning-based approaches for trajectory optimization. While the literature is vast, they can be broadly categorized into those that learn the models [90–98] and those that learn the cost/reward or constraints [99–102, 50, 103], where data are typically collected from in-domain interactions. To enable generalization in the wild, a parallel line of works has explored learning task specification from large-scale offline data [104–106, 35, 34, 44, 107, 108, 54], particularly egocentric videos [109, 110], or leveraging pre-trained foundation models [111–113, 33, 114, 115]. The learned cost functions are then used by reinforcement learning [111, 108, 116], imitation learning [106, 105], or trajectory optimization [104, 35] to generate robot actions. In this work, we leverage LLMs for *zero-shot in-the-wild* cost specification with superior generalization. Compared to prior works that leverage foundation models, we ground the cost directly in *3D observation space* with *real-time* visual feedback, which makes VoxPoser amenable to closed-loop MPC that's robust in execution.

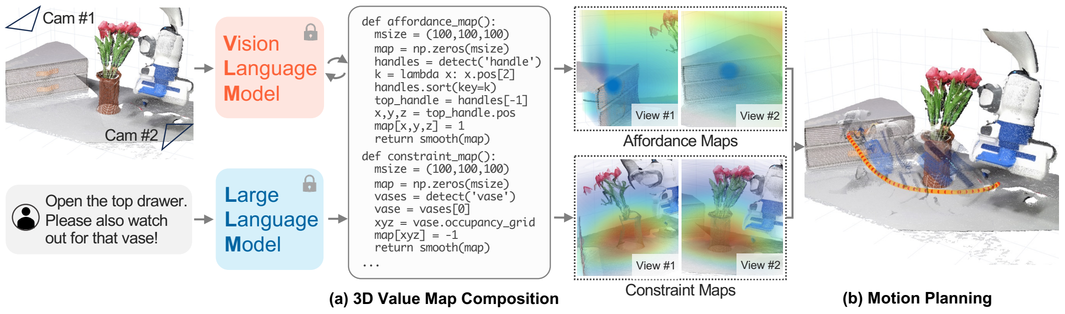
3 Method

We first provide the formulation of VoxPoser as an optimization problem (Sec. 3.1). Then we describe how VoxPoser can be used as a general zero-shot framework to map language instructions to 3D value maps (Sec. 3.2). We subsequently demonstrate how trajectories can be synthesized in closed-loop for robotic manipulation (Sec. 3.3). While zero-shot in nature, we demonstrate how VoxPoser can learn from online interactions to efficiently solve contact-rich tasks (Sec. 3.4).

3.1 Problem Formulation

Consider a manipulation problem given as a *free-form* language instruction $\mathcal{L}$ (e.g., "open the top drawer"). Generating robot trajectories according to $\mathcal{L}$ can be very challenging because $\mathcal{L}$ may be arbitrarily long-horizon or under-specified (i.e., requires contextual understanding). Instead, we focus on individual phases (sub-tasks) of the problem ℓ_i that distinctively specify a manipulation task (e.g., "grasp the drawer handle", "pull open the drawer"), where the decomposition $\mathcal{L} \rightarrow (\ell_1, \ell_2, \ldots, \ell_n)$ is given by a high-level planner (e.g., an LLM or a search-based planner) [2]. The central problem

[2]Note that the decomposition and sequencing of these sub-tasks are also done by LLMs in this work, though we do not investigate this aspect extensively as it is not the focus of our contributions.



```
def affordance_map():
  msize = (100,100,100)
  map = np.zeros(msize)
  handles = detect('handle')
  k = lambda x: x.pos[2]
  handles.sort(key=k)
  top_handle = handles[-1]
  x,y,z = top_handle.pos
  map[x,y,z] = 1
  return smooth(map)
def constraint_map():
  msize = (100,100,100)
  map = np.zeros(msize)
  vases = detect('vase')
  vase = vases[0]
  xyz = vase.occupancy_grid
  map[xyz] = -1
  return smooth(map)
...
```

Figure 2: Overview of VoxPoser. Given the RGB-D observation of the environment and a language instruction, LLMs generate code, which interacts with VLMs, to produce a sequence of 3D affordance maps and constraint maps (collectively referred to as value maps) grounded in the observation space of the robot **(a)**. The composed value maps then serve as objective functions for motion planners to synthesize trajectories for robot manipulation **(b)**. The entire process does not involve any additional training.

investigated in this work is to generate a motion trajectory $\tau_i^{\mathbf{r}}$ for robot $\mathbf{r}$ and each manipulation phase described by instruction ℓ_i. We represent $\tau_i^{\mathbf{r}}$ as a sequence of dense end-effector waypoints to be executed by an Operational Space Controller [117], where each waypoint consists of a desired 6-DoF end-effector pose, end-effector velocity, and gripper action. However, it is worth noting that other representations of trajectories, such as joint space trajectories, can also be used. Given the i-th sub-task described by ℓ_i, we formulate an optimization problem defined as follows:

$$\min_{\tau_i^{\mathbf{r}}} \{\mathcal{F}_{task}(\mathbf{T}_i, \ell_i) + \mathcal{F}_{control}(\tau_i^{\mathbf{r}})\} \quad \text{subject to} \quad \mathcal{C}(\mathbf{T}_i) \tag{1}$$

where $\mathbf{T}_i$ is the evolution of environment state, and $\tau_i^{\mathbf{r}} \subseteq \mathbf{T}_i$ is the robot trajectory. $\mathcal{F}_{task}$ scores the extent of $\mathbf{T}_i$ completes the instruction ℓ_i while $\mathcal{F}_{control}$ specifies the control costs, e.g., to encourage $\tau_i^{\mathbf{r}}$ to minimize total control effort or total time. $\mathcal{C}(\mathbf{T}_i)$ denotes the dynamics and kinematics constraints, which are enforced by the known model of the robot and a physics-based or learning-based model of the environment. By solving this optimization for each sub-task ℓ_i, we obtain a sequence of robot trajectories that collectively achieve the overall task specified by the instruction $\mathcal{L}$.

3.2 Grounding Language Instruction via VoxPoser

Calculating $\mathcal{F}_{task}$ with respect to free-form language instructions is extremely challenging, not only because of the rich space of semantics language can convey but also because of the lack of robot data labeled with $\mathbf{T}$ and ℓ. However, we provide a critical observation that a large number of tasks can be characterized by a voxel value map $\mathbf{V} \in \mathbb{R}^{w \times h \times d}$ in robot's observation space, which guides the motion of an "entity of interest" in the scene, such as the robot end-effector, an object, or an object part. For example, consider the task "open the top drawer" and its first sub-task "grasp the top drawer handle" (inferred by LLMs) in Fig. 2. The "entity of interest" is the robot end-effector, and the voxel value map should reflect the attraction toward the drawer handle. By further commanding "watch out for the vase", the map can also be updated to reflect the repulsion from the vase. We denote the "entity of interest" as $\mathbf{e}$ and its trajectory as $\tau^{\mathbf{e}}$. Using this voxel value map for a given instruction ℓ_i, $\mathcal{F}_{task}$ can be approximated by accumulating the values of $\mathbf{e}$ traversing through $\mathbf{V}_i$, formally calculated as $\mathcal{F}_{task} = -\sum_{j=1}^{|\tau_i^{\mathbf{e}}|} \mathbf{V}(p_j^{\mathbf{e}})$, where $p_j^{\mathbf{e}} \in \mathbb{N}^3$ is the discretized (x, y, z) position of $\mathbf{e}$ at step j.

Notably, we observe large language models, by being pre-trained on Internet-scale data, exhibit capabilities not only to identify the "entity of interest" but also to compose value maps that accurately reflect the task instruction by writing Python programs. Specifically, when an instruction is given as a comment in the code, LLMs can be prompted to 1) call perception APIs (which invoke vision-language models (VLM) such as an open-vocabulary detector [13–15]) to obtain spatial-geometrical information of relevant objects, 2) generate NumPy operations to manipulate 3D arrays, and 3) prescribe precise values at relevant locations. We term this approach as **VoxPoser**. Concretely, we aim to obtain a voxel value map $\mathbf{V}_i^t = \text{VoxPoser}(\mathbf{o}^t, \ell_i)$ by prompting an LLM and executing the code via a Python interpreter, where $\mathbf{o}^t$ is the RGB-D observation at time t and ℓ_i is the current

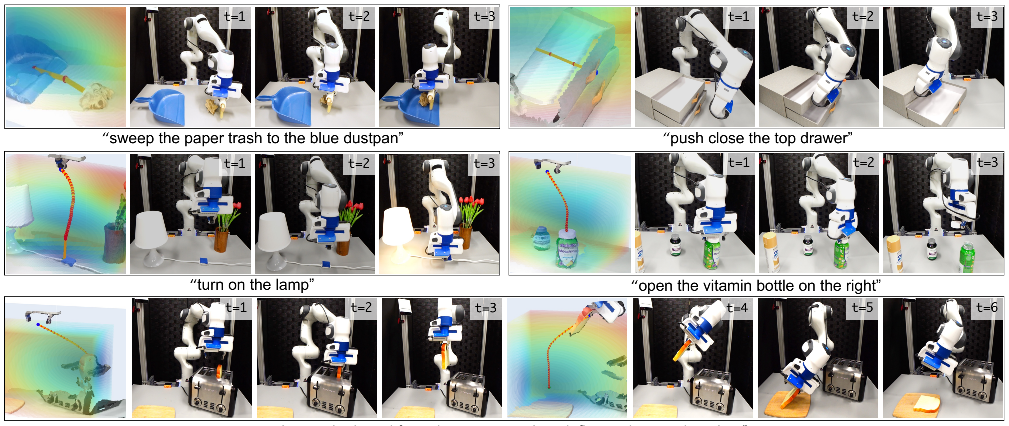

t=1
t=2
t=3
"sweep the paper trash to the blue dustpan"
t=1
t=2
t=3
"push close the top drawer"
t=1
t=2
t=3
"turn on the lamp"
t=1
t=2
t=3
"open the vitamin bottle on the right"
t=1
t=2
t=3
t=4
t=5
t=6
"take out the bread from the toaster and put it flat on the wooden plate"

Figure 3: Visualization of composed 3D value maps and rollouts in real-world environments. The top row demonstrates where "entity of interest" is an object or part, and the value maps guide them toward target positions. The bottom two rows showcase tasks where "entity of interest" is the robot end-effector. The bottom-most task involves two phases, which are also orchestrated by LLMs.

instruction. Additionally, because $\mathbf{V}$ is often sparse, we densify the voxel maps via smoothing operations, as they encourage smoother trajectories optimized by motion planners.

Additional Trajectory Parametrization. The above formulation of VoxPoser uses LLMs to compose $\mathbf{V}: \mathbb{N}^3 \to \mathbb{R}$ to map from discretized coordinates in voxel space to a real-valued "cost", which we can use to optimize a path consisting only of the positional terms. To extend to $\mathrm{SE}(3)$ poses, we can also use LLMs to compose rotation maps $\mathbf{V}_r: \mathbb{N}^3 \to \mathrm{SO}(3)$ at coordinates relevant to the task objectives (e.g., "end-effector should face the support normal of the handle"). Similarly, we further compose gripper maps $\mathbf{V}_g: \mathbb{N}^3 \to \{0, 1\}$ to control gripper open/close and velocity maps $\mathbf{V}_v: \mathbb{N}^3 \to \mathbb{R}$ to specify target velocities. Note that while these additional trajectory parametrizations are not mapped to a real-valued "cost", they can also be factored in the optimization procedure (Equation 1) to parametrize the trajectories.

3.3 Zero-Shot Trajectory Synthesis with VoxPoser

After obtaining the task cost $\mathcal{F}_{task}$, we can now approach the full problem defined in Equation 1 to plan a motion trajectory. We use simple zeroth-order optimization by randomly sampling trajectories and scoring them with the proposed objective. The optimization is implemented in a model predictive control framework that iteratively replans the trajectory at every step using the current observation to robustly execute the trajectories even under dynamic disturbances [3] , where either a learned or physics-based model can be used. However, because VoxPoser effectively provides "dense rewards" in the observation space and we are able to replan at every step, we surprisingly find that the overall system can already achieve a large variety of manipulation tasks considered in this work even with simple heuristics-based models. Since some value maps are defined over "entity of interest", which may not necessarily be the robot, we also use the dynamics model to find the needed robot trajectory to minimize the task cost (i.e., what interactions between the robot and the environment achieve the desired object motions).

3.4 Efficient Dynamics Learning with Online Experiences

While Sec. 3.3 presents a zero-shot framework for synthesizing trajectories for robot manipulation, VoxPoser can also benefit from online experiences by efficiently learning a dynamics model. Consider the standard setup where a robot interleaves between 1) collecting environment transition data $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$, where $\mathbf{o}_t$ is the environment observation at time t and $\mathbf{a}_t = \mathrm{MPC}(\mathbf{o}_t)$, and 2) training a dynamics model $\mathbf{g}_\theta$ parametrized by θ by minimizing the L2 loss between predicted next

[3] Although involving an LLM in the loop, closed-loop execution is possible because the generated code remains the same throughout task ℓ_i, which allows us to cache its output for the current task.

observation $\hat{\mathbf{o}}_{t+1}$ and $\mathbf{o}_{t+1}$. A critical component that determines the learning efficiency is the action sampling distribution $P(\mathbf{a}_t|\mathbf{o}_t)$ in MPC, which typically is a random distribution over the full action space $\mathbf{A}$. This is often inefficient when the goal is to solve a particular task, such as opening a door, because most actions do not interact with the relevant objects in the scene (i.e., the door handle) nor do they necessarily interact with the objects in a meaningful way (i.e., pressing down the door handle). Since VoxPoser synthesizes robot trajectories with LLMs, which have a wealth of commonsense knowledge, the zero-shot synthesized trajectory $\tau_0^{\mathbf{r}}$ can serve as a useful prior to bias the action sampling distribution $P(\mathbf{a}_t|\mathbf{o}_t, \tau_0^{\mathbf{r}})$, which can significantly speed up the learning process. In practice, this can be implemented by only sampling actions in the vicinity of $\tau_0^{\mathbf{r}}$ by adding small noise ε to encourage local exploration instead of exploring in the full action space $\mathbf{A}$.

4 Experiments and Analysis

We first discuss our implementation details. Then we validate VoxPoser for real-world everyday manipulation (Sec. 4.1). We also study its generalization in simulation (Sec. 4.2). We further demonstrate how VoxPoser enables efficient learning of more challenging tasks (Sec. 4.3). Finally, we analyze its source of errors and discuss how improvement can be made (Sec. 4.4).

LLMs and Prompting. We follow prompting structure by Liang et al. [75], which recursively calls LLMs using their own generated code, where each language model program (LMP) is responsible for a unique functionality (e.g., processing perception calls). We use GPT-4 [2] from OpenAI API. For each LMP, we include 5-20 example queries and corresponding responses as part of the prompt. An example can be found in Fig. 2 (simplified for clarity). Full prompts are in Appendix.

VLMs and Perception. Given an object/part query from LLMs, we first invoke open-vocab detector OWL-ViT [15] to obtain a bounding box, then feed it into Segment Anything [118] to obtain a mask, and finally track the mask using video tracker XMEM [119]. The tracked mask is used with RGB-D observation to reconstruct the object/part point cloud.

Value Map Composition. We define the following types of value maps: affordance, avoidance, end-effector velocity, end-effector rotation, and gripper action. Each type uses a different LMP, which takes in an instruction and outputs a voxel map of shape $(100, 100, 100, k)$, where k differs for each value map (e.g., $k = 1$ for affordance and avoidance as it specifies cost, and $k = 4$ for rotation as it specifies SO(3)). We apply Euclidean distance transform to affordance maps and Gaussian filters for avoidance maps. On top of value map LMPs, we define two high-level LMPs to orchestrate their behaviors: `planner` takes user instruction $\mathcal{L}$ as input (e.g., "open drawer") and outputs a sequence of sub-tasks $\ell_{1:N}$, and `composer` takes in sub-task ℓ_i and invokes relevant value map LMPs with detailed language parameterization.

Motion Planner. We consider only affordance and avoidance maps in the planner optimization, which finds a sequence of collision-free end-effector positions $p_{1:N} \in \mathbb{R}^3$ using greedy search. Then we enforce other parametrization at each p by the remaining value maps (e.g., rotation map, velocity map). The cost map used by the motion planner is computed as the negative of the weighted sum of normalized affordance and avoidance maps with weights 2 and 1. After a 6-DoF trajectory is synthesized, the first waypoint is executed, and then a new trajectory is re-planned at 5 Hz.

Dynamics Model. We use the known robot dynamics model in all tasks, where it is used in motion planning for the end-effector to follow the waypoints. For the majority of our considered tasks where the "entity of interest" is the robot, no environment dynamics model is used (i.e., scene is assumed to be static), but we replan at every step to account for the latest observation. For tasks in which the "entity of interest" is an object, we study only a planar pushing model parametrized by contact point, push direction, and push distance. We use a heuristic-based dynamics model that translates an input point cloud along the push direction by the push distance. We use MPC with random shooting to optimize for the action parameters. Then a pre-defined pushing primitive is executed based on the action parameters. However, we note that a primitive is not necessary when action parameters are defined over the end-effector or joint space of the robot, which would likely yield

Task	LLM + Prim. [75]		VoxPoser	
	Static	Dist.	Static	Dist.
Move & Avoid	0/10	0/10	9/10	8/10
Set Up Table	7/10	0/10	9/10	7/10
Close Drawer	0/10	0/10	10/10	7/10
Open Bottle	5/10	0/10	7/10	5/10
Sweep Trash	0/10	0/10	9/10	8/10
Total	24.0%	0.0%	88.0%	70.0%

Table 1: Success rate in real-world domain. VoxPoser performs everyday manipulation tasks with high success and is more robust to disturbances than the baseline using action primitives.

		U-Net	Language Models	
Train/Test	Category	MP [50]	Prim. [75]	MP (Ours)
SI SA	Object Int.	21.0%	41.0%	64.0%
SI SA	Composition	53.8%	43.8%	77.5%
SI UA	Object Int.	3.0%	46.0%	60.0%
SI UA	Composition	3.8%	25.0%	58.8%
UI UA	Object Int.	0.0%	17.5%	65.0%
UI UA	Composition	0.0%	25.0%	76.7%

Table 2: Success rate in simulated domain. "SI" and "UI" are seen and unseen instructions. "SA" and "UA" are seen and unseen attributes. VoxPoser outperforms both baselines across 13 tasks from two categories on both seen and unseen tasks and maintains similar success rates.

smoother trajectories but takes more time for optimization. We also explore the use of a learning-based dynamics model in Section 4.3, which enables VoxPoser to benefit from online experiences.

4.1 VoxPoser for Everyday Manipulation Tasks

We study whether VoxPoser can zero-shot synthesize robot trajectories to perform everyday manipulation tasks in the real world. Details of the environment setup can be found in Appendix A.4. While the proposed method can generalize to an open-set of instructions and an open-set of objects as shown in Fig. 1, we pick 5 representative tasks to provide quantitative evaluations in Table 1. Qualitative results including environment rollouts and value map visualizations are shown in Fig. 3. We find that VoxPoser can effectively synthesize robot trajectories for everyday manipulation tasks with a high average success rate. Due to fast replanning capabilities, it is also robust to external disturbances, such as moving targets/obstacles and pulling the drawer open after it has been closed by the robot. We further compare to a variant of Code as Policies [75] that uses LLMs to parameterize a pre-defined list of simple primitives (e.g., `move_to_pose`, `open_gripper`). We find that compared to chaining sequential policy logic, the ability to *compose spatially* while considering other constraints under a joint optimization scheme is a more flexible formulation, unlocking the possibility for more manipulation tasks and leading to more robust execution.

4.2 Generalization to Unseen Instructions and Attributes

To provide rigorous quantitative evaluations on generalization, we set up a simulated block-world environment that mirrors our real-world robot setup [120, 121] but features 13 highly-randomizable tasks with 2766 unique instructions. Eash task comes with a templated instruction (e.g., "push [obj] to [pos]") that contains randomizable attributes chosen from a pre-defined list. Details are in Appendix A.5. Seen instructions/attributes may appear in the prompt (or in the training data for supervised baselines). The tasks are grouped into 2 categories, where "Object Interactions" are tasks that require interactions with objects, and "Spatial Composition" are tasks involving spatial constraints (e.g., moving slower near a particular object). For baselines, we ablate the two components of VoxPoser, LLM and motion planner, by comparing to a variant of [75] that combines an LLM with primitives and to a variant of [50] that learns a U-Net [122] to synthesize costmaps for motion planning. Table 2 shows the success rates averaged across 20 episodes per task. We find VoxPoser exhibits superior generalization in all scenarios. Compared to learned cost specification, LLMs generalize better by explicitly reasoning about affordances and constraints. On the other hand, grounding LLM knowledge in robot perception through *value map composition* rather than directly specifying primitive parameters offers more flexibility and better generalization.

4.3 Efficient Dynamics Learning with Online Experiences

As discussed in Sec. 3.4, we investigate how VoxPoser can optionally benefit from online experiences for tasks that involve more intricacies of contact, such as opening doors, fridges, and windows, in a simulated environment. Specifically, we first synthesize k zero-shot trajectories using VoxPoser,

	Zero-Shot	No Prior		w/ Prior	
Task	Success	Success	Time(s)	Success	Time(s)
Door	6.7%$\pm$4.4%	58.3$\pm$4.4%	TLE	88.3%$\pm$1.67%	142.3$\pm$22.4
Window	3.3%$\pm$3.3%	36.7%$\pm$1.7%	TLE	80.0%$\pm$2.9%	137.0$\pm$7.5
Fridge	18.3%$\pm$3.3%	70.0%$\pm$2.9%	TLE	91.7%$\pm$4.4%	71.0$\pm$4.4

Table 3: VoxPoser enables efficient dynamics learning by using zero-shot synthesized trajectories as prior. TLE (time limit exceeded) means exceeding 12 hours. Results are reported over 3 runs different seeds.

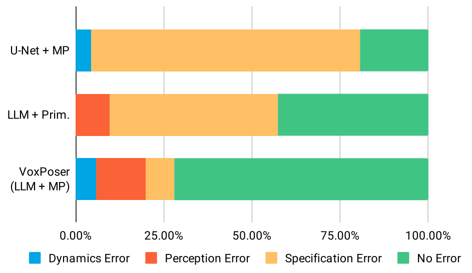


Figure 4: Error breakdown of components. VoxPoser significantly reduces specification error.

each represented as a sequence of end-effector waypoints, that act as priors for exploration (e.g., "handle needs to be pressed down first in order to open a door"). Then an MLP dynamics model is learned through an iterative procedure where the agent alternates between data collection and model learning. During data collection, we add $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ to each waypoint in $\tau_0^{\mathbf{r}}$ to encourage local exploration. As shown in Tab. 3, we find zero-shot synthesized trajectories are typically meaningful but insufficient. However, we can learn an effective dynamics model with less than 3 minutes of online interactions by using these trajectories as exploration prior, leading to high eventual success rates. In comparison, exploring without prior all exceed the maximum 12-hour limit.

4.4 Error Breakdown

In this section, we analyze the errors resulting from each component of VoxPoser and how the overall system can be further improved. We conduct experiments in simulation where we have access to ground-truth perception and dynamics model (i.e., the simulator). . "Dynamics error" refers to errors made by the dynamics model[4]. "Perception error" refers to errors made by the perception module[5]. "Specification error" refers to errors made by the module specifying cost or parameters for the low-level motion planner or primitives. Examples for each method include 1) noisy prediction by the U-Net, 2) incorrect parameters specified by the LLM, and 3) incorrect value maps specified by the LLM. As shown in Fig. 4, VoxPoser achieves lowest "specification error" due to its generalization and flexibility. We also find that having access to a more robust perception pipeline and a physically-realistic dynamics model can contribute to better overall performance. This observation aligns with our real-world experiment, where most errors are from perception. For example, we find that the detector is sensitive to initial poses of objects and is less robust when detecting object parts.

5 Conclusion, Limitations, & Future Works

In this work, we present **VoxPoser**, a general framework for extracting affordances and constraints, grounded in 3D perceptual space, from LLMs and VLMs for everyday manipulation tasks in the real world, offering significant generalization advantages for open-set instructions and objects. Despite compelling results, VoxPoser has several limitations. First, it relies on external perception modules, which is limiting in tasks that require holistic visual reasoning or understanding of fine-grained object geometries. Second, while applicable to efficient dynamics learning, a general-purpose dynamics model is still required to achieve contact-rich tasks with the same level of generalization. Third, our motion planner considers only end-effector trajectories while whole-arm planning is also feasible and likely a better design choice [124–126]. Finally, manual prompt engineering is required for LLMs. We also see several exciting venues for future work. For instance, recent success of multi-modal LLMs [68, 2, 127] can be directly translated into VoxPoser for direct visual grounding. Methods developed for alignment [128, 129] and prompting [130–133] may be used to alleviate prompt engineering effort. Finally, more advanced trajectory optimization methods can be developed that best interface with value maps synthesized by VoxPoser.

[4]**LLM + Primitives** [75] does not use model-based planning, thus not having a dynamics module.

[5]**U-Net + MP** [50] maps RGB-D to costmaps using U-Net [122, 123], thus not having perception module. Errors by which are attributed to "specification error".

Acknowledgments

We would like to thank Andy Zeng, Igor Mordatch, and the members of the Stanford Vision and Learning Lab for the fruitful discussions. This work was in part supported by AFOSR YIP FA9550-23-1-0127, ONR MURI N00014-22-1-2740, ONR MURI N00014-21-1-2801, ONR N00014-23-1-2355, the Stanford Institute for Human-Centered AI (HAI), JPMC, and Analog Devices. Wenlong Huang is partially supported by Stanford School of Engineering Fellowship. Ruohan Zhang is partially supported by Wu Tsai Human Performance Alliance Fellowship.

References

[1] 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*, 33:1877–1901, 2020.

[2] OpenAI. Gpt-4 technical report. *arXiv*, 2023.

[3] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[4] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[5] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 2020.

[6] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1507–1514, 2011.

[7] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266. IEEE, 2010.

[8] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

[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.

[10] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[11] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[12] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[13] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[14] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[15] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, et al. Simple open-vocabulary object detection with vision transformers. *arXiv preprint arXiv:2205.06230*, 2022.

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi:10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[17] Y. K. Hwang, N. Ahuja, et al. A potential field approach to path planning. *IEEE transactions on robotics and automation*, 8(1):23–32, 1992.

[18] M. Toussaint, J. Harris, J.-S. Ha, D. Driess, and W. Hönig. Sequence-of-constraints mpc: Reactive timing-optimal control of sequential manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13753–13760. IEEE, 2022.

[19] J. Andreas, D. Klein, and S. Levine. Learning with latent language. *arXiv preprint arXiv:1711.00482*, 2017.

[20] R. Zellers, A. Holtzman, M. Peters, R. Mottaghi, A. Kembhavi, A. Farhadi, and Y. Choi. Piglet: Language grounding through neuro-symbolic interaction in a 3d world. *arXiv preprint arXiv:2106.00188*, 2021.

[21] R. Zellers, X. Lu, J. Hessel, Y. Yu, J. S. Park, J. Cao, A. Farhadi, and Y. Choi. Merlot: Multimodal neural script knowledge models. *Advances in Neural Information Processing Systems*, 2021.

[22] V. Shwartz, P. West, R. L. Bras, C. Bhagavatula, and Y. Choi. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*, 2020.

[23] T. Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

[24] V. Blukis, R. A. Knepper, and Y. Artzi. Few-shot object grounding and mapping for natural language robot instruction following. *arXiv preprint arXiv:2011.07384*, 2020.

[25] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. Asking for help using inverse semantics. *Robotics: Science and Systems Foundation*, 2014.

[26] T. Kollar, S. Tellex, D. Roy, and N. Roy. Grounding verbs of motion in natural language commands to robots. In *Experimental robotics*, pages 31–47. Springer, 2014.

[27] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[28] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, J. Hart, P. Stone, and R. Mooney. Jointly improving parsing and perception for natural language commands through human-robot dialog. *Journal of Artificial Intelligence Research*, 67:327–374, 2020.

[29] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2021.

[30] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[31] D. Shah, B. Osinski, B. Ichter, and S. Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint arXiv:2207.04429*, 2022.

[32] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh. " no, to the right"– online language corrections for robotic manipulation via shared autonomy. *arXiv preprint arXiv:2301.02555*, 2023.

[33] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.

[34] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[35] Y. J. Ma, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman. Liv: Language-image representations and rewards for robotic control. *arXiv e-prints*, 2023.

[36] P. A. Jansen. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. *arXiv preprint arXiv:2009.14259*, 2020.

[37] V. Micheli and F. Fleuret. Language models are few-shot butlers. *arXiv preprint arXiv:2104.07972*, 2021.

[38] P. Sharma, A. Torralba, and J. Andreas. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

[39] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022.

[40] B. Z. Li, W. Chen, P. Sharma, and J. Andreas. Lampp: Language models as probabilistic priors for perception and action. *arXiv e-prints*, pages arXiv–2302, 2023.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[43] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[44] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022.

[45] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[46] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[47] S. Li, X. Puig, Y. Du, C. Wang, E. Akyurek, A. Torralba, J. Andreas, and I. Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

[48] O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation learning. *arXiv preprint arXiv:2204.06252*, 2022.

[49] O. Mees, J. Borja-Diaz, and W. Burgard. Grounding language with visual affordances over unstructured data. *arXiv preprint arXiv:2210.01911*, 2022.

[50] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox. Correcting robot plans with natural language feedback. *arXiv preprint arXiv:2204.05186*, 2022.

[51] 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)*, pages 6322–6329. IEEE, 2022.

[52] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021. URL https://arxiv.org/abs/2005.07648.

[53] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*, 2022.

[54] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

[55] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, 2019.

[56] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. *ArXiv*, abs/1611.01796, 2017.

[57] Y. Jiang, S. S. Gu, K. P. Murphy, and C. Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[58] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler. Open-vocabulary queryable scene representations for real world planning. *arXiv preprint arXiv:2209.09874*, 2022.

[59] 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. *arXiv preprint arXiv:2209.11302*, 2022.

[60] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022.

[61] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[62] 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. *arXiv preprint arXiv:2212.04088*, 2022.

[63] 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.

[64] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor. Chatgpt for robotics: Design principles and model abilities. *2023*, 2023.

[65] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans. *arXiv preprint arXiv:2303.12153*, 2023.

[66] 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.

[67] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, et al. Grounded decoding: Guiding text generation with grounded models for robot control. *arXiv preprint arXiv:2303.00855*, 2023.

[68] 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.

[69] H. Yuan, C. Zhang, H. Wang, F. Xie, P. Cai, H. Dong, and Z. Lu. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023.

[70] 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.

[71] Y. Lu, P. Lu, Z. Chen, W. Zhu, X. E. Wang, and W. Y. Wang. Multimodal procedural planning via dual text-image prompting. *arXiv preprint arXiv:2305.01795*, 2023.

[72] D. Patel, H. Eghbalzadeh, N. Kamra, M. L. Iuzzolino, U. Jain, and R. Desai. Pretrained language models as visual planners for human assistance. *arXiv preprint arXiv:2304.09179*, 2023.

[73] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[74] J. Yang, W. Tan, C. Jin, B. Liu, J. Fu, R. Song, and L. Wang. Pave the way to grasp anything: Transferring foundation models for universal pick-place robots. *arXiv preprint arXiv:2306.05716*, 2023.

[75] 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. *arXiv preprint arXiv:2209.07753*, 2022.

[76] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

[77] A. Tam, N. Rabinowitz, A. Lampinen, N. A. Roy, S. Chan, D. Strouse, J. Wang, A. Banino, and F. Hill. Semantic exploration from language abstractions and pretrained representations. *Advances in Neural Information Processing Systems*, 35:25377–25389, 2022.

[78] J. Mu, V. Zhong, R. Raileanu, M. Jiang, N. Goodman, T. Rocktäschel, and E. Grefenstette. Improving intrinsic exploration with language abstractions. *arXiv preprint arXiv:2202.08938*, 2022.

[79] C. Colas, T. Karch, N. Lair, J.-M. Dussoux, C. Moulin-Frier, P. Dominey, and P.-Y. Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems*, 33:3761–3774, 2020.

[80] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*, 2023.

[81] H. Hu and D. Sadigh. Language instructed reinforcement learning for human-ai coordination. *arXiv preprint arXiv:2304.07297*, 2023.

[82] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[83] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. *arXiv preprint arXiv:2307.14535*, 2023.

[84] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[85] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023.

[86] Y. Zeng and Y. Xu. Learning reward for physical skills using large language model. *arXiv preprint arXiv:2310.14092*, 2023.

[87] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

[88] J. Perez, D. Proux, C. Roux, and M. Niemaz. Larg, language-based automatic reward and goal generation. *arXiv preprint arXiv:2306.10985*, 2023.

[89] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[90] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10. Rome, Italy, 2015.

[91] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[92] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[93] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[94] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

[95] A. Byravan and D. Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017.

[96] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[97] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[98] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[99] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[100] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[101] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9563–9569. IEEE, 2020.

[102] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018.

[103] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1647–1654. IEEE, 2022.

[104] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022.

[105] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13778–13790, 2023.

[106] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[107] H. Bharadhwaj, A. Gupta, S. Tulsiani, and V. Kumar. Zero-shot robot manipulation from passive human videos. *arXiv preprint arXiv:2302.02011*, 2023.

[108] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[109] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[110] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[111] Y. Cui, S. Niekum, A. Gupta, V. Kumar, and A. Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation? In *Learning for Dynamics and Control Conference*, pages 893–905. PMLR, 2022.

[112] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv:2302.11550*, 2023.

[113] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar. Cacti: A framework for scalable multi-task multi-scene visual imitation learning. *arXiv preprint arXiv:2212.05711*, 2022.

[114] T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson. Robotic skill acquisition via instruction augmentation with vision-language models. *arXiv preprint arXiv:2211.11736*, 2022.

[115] C. Wang, D. Xu, and L. Fei-Fei. Generalizable task planning through representation pretraining. *IEEE Robotics and Automation Letters*, 7(3):8299–8306, 2022.

[116] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[117] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[118] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[119] H. K. Cheng and A. G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 640–658. Springer, 2022.

[120] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[121] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6813–6823, 2021.

[122] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[123] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[124] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[125] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[126] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake. Motion planning around obstacles with convex optimization. *arXiv preprint arXiv:2205.04422*, 2022.

[127] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[128] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[129] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[130] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[131] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[132] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[133] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[134] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[135] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[136] T. Gupta and A. Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.

[137] D. Surís, S. Menon, and C. Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

[138] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. *6th Annual Conference on Robot Learning*, 2022.

A Appendix

A.1 Code Release

We provide an open-sourced implementation of VoxPoser at github.com/huangwl18/VoxPoser based on RLBench [134], as its diversity of tasks and scenes best resembles our real-world setup.

A.2 Emergent Behavioral Capabilities

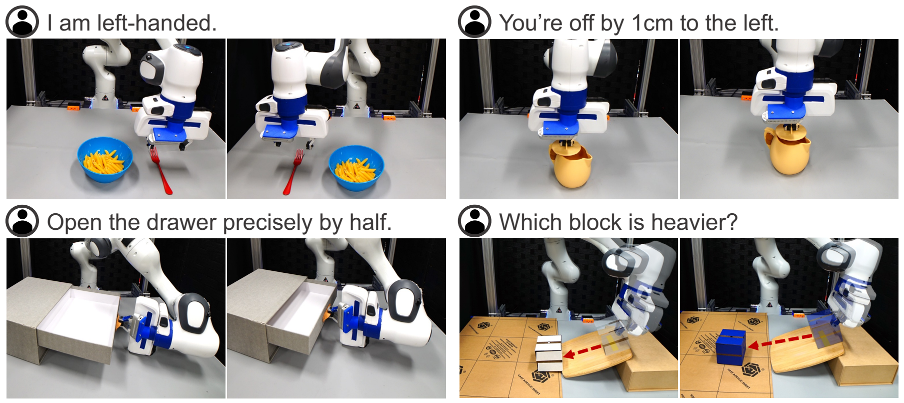

I am left-handed.
You're off by 1cm to the left.
Open the drawer precisely by half.
Which block is heavier?

Figure 5: Emergent behavioral capabilities by VoxPoser inherited from the language model, including behavioral commonsense reasoning (**top left**), fine-grained language correction (**top right**), multi-step visual program (**bottom left**), and estimating physical properties of objects (**bottom right**).

Emergent capabilities refer to unpredictable phenomenons that are only present in large models [135]. As VoxPoser uses pre-trained LLMs as backbone, we observe similar embodied emergent capabilities driven by the rich world knowledge of LLMs. In particular, we focus our study on the *behavioral capabilities* that are unique to VoxPoser. We observe the following capabilities:

- **Behavioral Commonsense Reasoning**: During a task where robot is setting the table, the user can specify behavioral preferences such as "I am left-handed", which requires the robot to comprehend its meaning in the context of the task. VoxPoser decides that it should move the fork from the right side of the bowl to the left side.
- **Fine-grained Language Correction**: For tasks that require high precision such as "covering the teapot with the lid", the user can give precise instructions to the robot such as "you're off by 1cm". VoxPoser similarly adjusts its action based on the feedback.
- **Multi-step Visual Program** [136, 137]: Given a task "open the drawer precisely by half" where there is insufficient information because object models are not available, VoxPoser can come up with multi-step manipulation strategies based on visual feedback that first opens the drawer fully while recording handle displacement, then close it back to the midpoint to satisfy the requirement.
- **Estimating Physical Properties**: Given two blocks of unknown mass, the robot is tasked to conduct physics experiments using an existing ramp to determine which block is heavier. VoxPoser decides to push both blocks off the ramp and choose the block traveling the farthest as the heavier block. Interestingly, this mirrors a common human oversight: in an ideal, frictionless world, both blocks would traverse the same distance under the influence of gravity. This serves as a lighthearted example that language models can exhibit limitations similar to human reasoning.

A.3 APIs for VoxPoser

Central to VoxPoser is an LLM generating Python code that is executed by a Python interpreter. Besides exposing NumPy [16] and the Transforms3d library to the LLM, we provide the following environment APIs that LLMs can choose to invoke:

`detect(obj_name)`: Takes in an object name and returns a list of dictionaries, where each dictionary corresponds to one instance of the matching object, containing center position, occupancy grid, and mean normal vector.

`execute(movable,affordance_map,avoidance_map,rotation_map,velocity_map,gripper_map)`: Takes in an "entity of interest" as "movable" (a dictionary returned by `detect`) and (optionally) a list of value maps and invokes the motion planner to execute the trajectory. Note that in MPC settings, "movable" and the input value maps are functions that can be re-evaluated to reflect the latest environment observation.

`cm2index(cm,direction)`: Takes in a desired offset distance in centimeters along direction and returns 3-dim vector reflecting displacement in voxel coordinates.

`index2cm(index,direction)`: Inverse of `cm2index`. Takes in an integer "index" and a "direction" vector and returns the distance in centimeters in world coordinates displaced by the "integer" in voxel coordinates.

`pointat2quat(vector)`: Takes in a desired pointing direction for the end-effector and returns a satisfying target quaternion.

`set_voxel_by_radius(voxel_map,voxel_xyz,radius_cm,value)`: Assigns "value" to voxels within "radious_cm" from "voxel_xyz" in "voxel_map".

`get_empty_affordance_map()`: Returns a default affordance map initialized with 0, where a high value attracts the entity.

`get_empty_avoidance_map()`: Returns a default avoidance map initialized with 0, where a high value repulses the entity.

`get_empty_rotation_map()`: Returns a default rotation map initialized with current end-effector quaternion.

`get_empty_gripper_map()`: Returns a default gripper map initialized with current gripper action, where 1 indicates "closed" and 0 indicates "open".

`get_empty_velocity_map()`: Returns a default affordance map initialized with 1, where the number represents scale factor (e.g., 0.5 for half of the default velocity).

`reset_to_default_pose()`: Reset to robot rest pose.

A.4 Real-World Environment Setup

We use a Franka Emika Panda robot with a tabletop setup. We use Operational Space Controller with impedance from Deoxys [138]. We mount two RGB-D cameras (Azure Kinect) at two opposite ends of the table: bottom right and top left from the top down view. At the start of each rollout, both cameras start recording and return the real-time RGB-D observations at 20 Hz.

For each task, we evaluate each method on two settings: without and with disturbances. For tasks with disturbances, we apply three kinds of disturbances to the environment, which we pre-select a sequence of them at the start of the evaluation: 1) random forces applied to the robot, 2) random displacement of task-relevant and distractor objects, and 3) reverting task progress (e.g., pull drawer open while it's being closed by the robot). We only apply the third disturbances to tasks where "entity of interest" is an object or object part.

We compare to a variant of Code as Policies [75] as a baseline that uses an LLM with action primitives. The primitives include: `move_to_pos`, `rotate_by_quat`, `set_vel`, `open_gripper`, `close_gripper`. We do not provide primitives such as pick-and-place as they would be tailored for a particular suite of tasks that we do not constrain to in our study (similar to the control APIs for VoxPoser specified in Sec. A.3).

A.4.1 Tasks

Move & Avoid: "Move to the top of [obj1] while staying away from [obj2]", where [obj1] and [obj2] are randomized everyday objects selected from the list: apple, banana, yellow bowl, headphones, mug, wood block.

Set Up Table: "Please set up the table by placing utensils for my pasta".

Close Drawer: "Close the [deixis] drawer", where [deixis] can be "top" or "bottom".

Open Bottle: "Turn open the vitamin bottle".

Sweep Trash: "Please sweep the paper trash into the blue dustpan".

A.5 Simulated Environment Setup

We implement a tabletop manipulation environment with a Franka Emika Panda robot in SAPIEN [120]. The controller takes as input a desired end-effector 6-DoF pose, calculates a sequence of interpolated waypoints using inverse kinematics, and finally follows the waypoints using a PD controller. We use a set of 10 colored blocks and 10 colored lines in addition to an articulated cabinet with 3 drawers. They are initialized differently depending on the specific task. The lines are used as visual landmarks and are not interactable. For perception, a total of 4 RGB-D cameras are mounted at each end of the table pointing at the center of the workspace.

A.5.1 Tasks

We create a custom suite of 13 tasks shown in Table 4. Each task comes with a templated instruction (shown in Table 4) where there may be one or multiple attributes randomized from the pre-defined list below. At reset time, a number of objects are selected (depending on the specific task) and are randomized across the workspace while making sure that task is not completed at reset and that task completion is feasible. A complete list of attributes can be found below, divided into "seen" and "unseen" categories:

Seen Attributes:

- **`[pos]`**: ["back left corner of the table", "front right corner of the table", "right side of the table", "back side of the table"]
- **`[obj]`**: ["blue block", "green block", "yellow block", "pink block", "brown block"]
- **`[preposition]`**: ["left of", "front side of", "top of"]
- **`[deixis]`**: ["topmost", "second to the bottom"]
- **`[dist]`**: [3, 5, 7, 9, 11]
- **`[region]`**: ["right side of the table", "back side of the table"]
- **`[velocity]`**: ["faster speed", "a quarter of the speed"]
- **`[line]`**: ["blue line", "green line", "yellow line", "pink line", "brown line"]

Unseen Attributes:

- **`[pos]`**: ["back right corner of the table", "front left corner of the table", "left side of the table", "front side of the table"]
- **`[obj]`**: ["red block", "orange block", "purple block", "cyan block", "gray block"]
- **`[preposition]`**: ["right of", "back side of"]
- **`[deixis]`**: ["bottommost", "second to the top"]
- **`[dist]`**: [4, 6, 8, 10]
- **`[region]`**: ["left side of the table", "front side of the table"]
- **`[velocity]`**: ["slower speed", "3x speed"]
- **`[line]`**: ["red line", "orange line", "purple line", "cyan line", "gray line"]

A.5.2 Full Results on Simulated Environments

	U-Net + MP		**LLM + Prim.**		**VoxPoser**	
Tasks	SA	UA	SA	UA	SA	UA
move to the [preposition] the [obj]	95.0%	0.0%	85.0%	60.0%	90.0%	55.0%
move to the [pos] while staying on the [preposition] the [obj]	100.0%	10.0%	80.0%	30.0%	95.0%	50.0%
move to the [pos] while moving at [velocity] when within [dist]cm from the obj	80.0%	0.0%	10.0%	0.0%	100.0%	95.0%
close the [deixis] drawer by pushing	0.0%	0.0%	60.0%	60.0%	80.0%	80.0%
push the [obj] along the [line]	0.0%	0.0%	0.0%	0.0%	65.0%	30.0%
grasp the [obj] from the table at [velocity]	35.0%	0.0%	75.0%	70.0%	65.0%	40.0%
drop the [obj] to the [pos]	70.0%	10.0%	60.0%	100.0%	60.0%	100.0%
push the [obj] while letting it stay on [region]	0.0%	5.0%	10.0%	0.0%	50.0%	50.0%
move to the [region]	5.0%	0.0%	100.0%	95.0%	100.0%	100.0%
move to the [pos] while staying at least [dist]cm from the [obj]	0.0%	0.0%	15.0%	20.0%	85.0%	90.0%
move to the [pos] while moving at [velocity] in the [region]	0.0%	0.0%	90.0%	45.0%	85.0%	85.0%
push the [obj] to the [pos] while staying away from [obstacle]	0.0%	0.0%	0.0%	10.0%	45.0%	55.0%
push the [obj] to the [pos]	0.0%	0.0%	20.0%	25.0%	80.0%	75.0%

Table 4: Full experimental results in simulation on seen tasks and unseen tasks. “SA” indicates seen attributes and “UA” indicates unseen attributes. Each entry represents success rate averaged across 20 episodes.

A.6 Prompts

Prompts used in Sec. 4.1 and Sec. 4.2 can be found below.

`planner`: Takes in a user instruction $\mathcal{L}$ and generates a sequence of sub-tasks ℓ_i which is fed into "composer" (Note that planner is not used in simulation as the evaluated tasks consist of a single manipulation phase).

real-world: voxposer.github.io/prompts/real_planner_prompt.txt.

`composer`: Takes in sub-task instruction ℓ_i and invokes necessary value map LMPs to compose affordance maps and constraint maps.

simulation: voxposer.github.io/prompts/sim_composer_prompt.txt.

real-world: voxposer.github.io/prompts/real_composer_prompt.txt.

`parse_query_obj`: Takes in a text query of object/part name and returns a list of dictionaries, where each dictionary corresponds to one instance of the matching object containing center position, occupancy grid, and mean normal vector.

simulation: voxposer.github.io/prompts/sim_parse_query_obj_prompt.txt.

real-world: voxposer.github.io/prompts/real_parse_query_obj_prompt.txt.

`get_affordance_map`: Takes in natural language parametrization from `composer` and returns a NumPy array for task affordance map.

simulation: voxposer.github.io/prompts/sim_get_affordance_map_prompt.txt.

real-world: voxposer.github.io/prompts/real_get_affordance_map_prompt.txt.

`get_avoidance_map`: Takes in natural language parametrization from `composer` and returns a NumPy array for task avoidance map.

simulation: voxposer.github.io/prompts/sim_get_avoidance_map_prompt.txt.

real-world: voxposer.github.io/prompts/real_get_avoidance_map_prompt.txt.

`get_rotation_map`: Takes in natural language parametrization from `composer` and returns a NumPy array for end-effector rotation map.

simulation: voxposer.github.io/prompts/sim_get_rotation_map_prompt.txt.

real-world: voxposer.github.io/prompts/real_get_rotation_map_prompt.txt.

`get_gripper_map`: Takes in natural language parametrization from `composer` and returns a NumPy array for gripper action map.

simulation: voxposer.github.io/prompts/sim_get_gripper_map_prompt.txt.

real-world: voxposer.github.io/prompts/real_get_gripper_map_prompt.txt.

`get_velocity_map`: Takes in natural language parametrization from `composer` and returns a NumPy array for end-effector velocity map.

simulation: voxposer.github.io/prompts/sim_get_velocity_map_prompt.txt.

real-world: voxposer.github.io/prompts/real_get_velocity_map_prompt.txt.