Blox-Net: Generative Design-for-Robot-Assembly Using VLM Supervision, Physics Simulation, and a Robot with Reset

Andrew Goldberg[1], Kavish Kondap[1], Tianshuang Qiu[1], Zehan Ma[1], Letian Fu[1]
Justin Kerr[1], Huang Huang[1], Kaiyuan Chen[1], Kuan Fang[2], Ken Goldberg[1]

https://bloxnet.org/

***Abstract*— Generative AI systems have shown impressive capabilities in creating text, code, and images. Inspired by the importance of research in industrial Design for Assembly, we introduce a novel problem: Generative Design-for-Robot-Assembly (GDfRA). The task is to generate an assembly based on a natural language prompt (e.g., "giraffe") and an image of available physical components, such as 3D-printed blocks. The output is an assembly, a spatial arrangement of these components, accompanied by instructions for a robot to build it. The output geometry must 1) resemble the requested object and 2) be reliably assembled by a 6 DoF robot arm with a suction gripper. We then present Blox-Net, a GDfRA system that combines generative vision language models with well-established methods in computer vision, simulation, perturbation analysis, motion planning, and physical robot experimentation to solve a class of GDfRA problems without human supervision. Blox-Net achieved a Top-1 accuracy of 63.5% in the semantic accuracy of its designed assemblies. Six designs, after Blox-Net's automated pertubation redesign, were reliably assembled by a robot, achieving near-perfect success across 10 consecutive assembly iterations with human intervention only during reset prior to assembly. The entire pipeline from the textual word to reliable physical assembly is performed without human intervention. Project Page: https://bloxnet.org/**

I. Introduction

Design-for-Assembly (DfA) has a long history dating back to the start of the Industrial Revolution, where guns, pocket watches, and clocks were designed with interchangeable parts to facilitate mass production on human assembly lines [1]. With the advent of industrial automation in the 20th century, DfA was expanded to take into account the error tolerances of mechanical assembly systems driven by mechanical cams and belts, and later for robotic assembly systems, the latter known as Design-for-Robot-Assembly (DfRA) [2]. DfRA is the process of designing a product and robot assembly system together to facilitate reliable assembly, for example designing an injection molded part along with a custom workcell for manipulating it. DfRA systems were enhanced by the emergence of Computer-Aided Design (CAD) and Computer-Aided-Manufacturing (CAM) software that streamline human visualization and evaluation of components and assemblies using Finite Element Methods (FEM) and perturbation analysis [3–6]. Such systems help human designers visualize and arrange mechanical components with realistic tolerances, checking for potential jamming and wedging conditions [7].

[1]The AUTOLab at UC Berkeley, [2]Cornell University.
Emails: {apgoldberg, kavish_kondap, ethantqiu, zehanma, max.fu.letian, justin_kerr, huangr, kych, goldberg}@berkeley.edu, {kuanfang}@cornell.edu
AUTOLab Website: https://autolab.berkeley.edu/

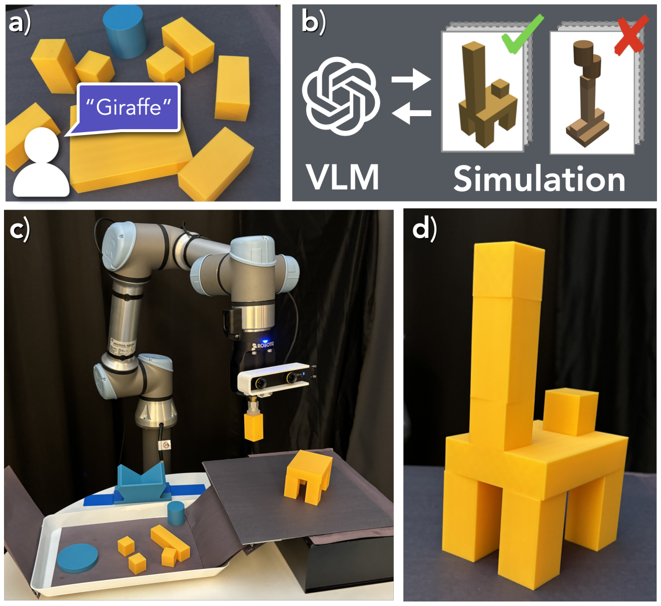

Fig. 1: *Can a vision-language model generate designs suitable for robot assembly?* **Blox-Net** is a GDfRA system that produces 3D designs constructible by robots subject to physical material constraints. **(a)** Starting with a phrase (e.g., "giraffe") and a set of blocks, **(b)** Blox-Net iteratively prompts GPT-4o to generate designs, using simulation to verify stability. **(c)** A physical robot then assembles the design to test stability and constructibility, **(d)** resulting in the successful assembly of the design.

All existing DfRA systems require human designers in the loop [3, 4, 7]. One factor that is difficult for DfRA systems to accurately model is the reliability of robot assembly, which depends on the inherent uncertainty in robot perception, control, and physics (eg, friction) [8–13]. This can to some degree be modeled with simulation, but it is well-known that 3D simulation systems struggle to accurately model minute 3D deformations, friction, and collisions that occur during robot grasping and effects such as deformations of robot gripper and suction cups which can produce substantial errors leading to assembly failures [14–18]. Therefore, physical assembly trials are ideal for evaluation.

Recent advances in Generative AI systems have demonstrated remarkable abilities to create novel texts, code, and images [19–21]. Researchers are actively exploring "text-to-video" [22–24] and "text-to-3D" [25–27] systems, where the

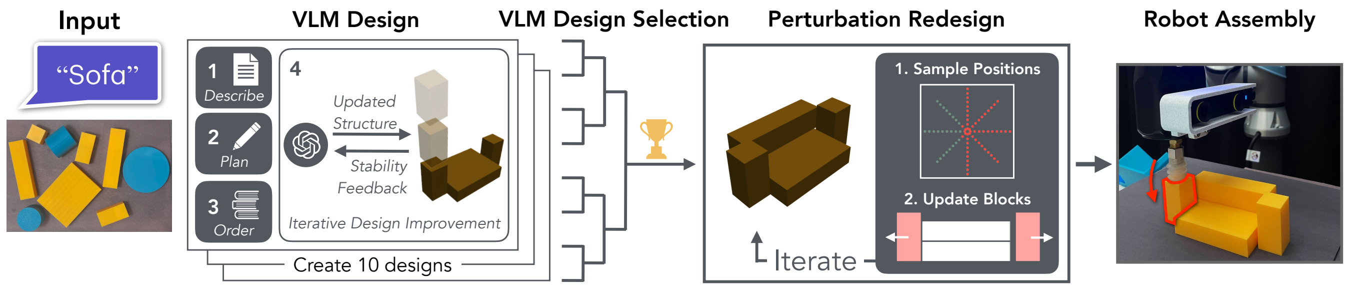

Fig. 2: **Overview of Blox-Net.** We present a multi-stage framework for producing physically constructible models based on a user-specified prompt. The Blox-Net pipeline begins with a natural language input and JSON detailing the available blocks. These parameters are passed into a series of VLM prompts, beginning with a high-level overview (*Describe*), followed by requesting specific blocks to use in construction (*Plan*) and a sequence to place them in (*Order*). Finally, the VLM generates the initial design and enters a feedback loop, continuously receiving visual and stability feedback from the simulator. After generating 10 candidate designs, a separate VLM selects the best structure through head-to-head image comparisons. The perturbation redesign phase then adjusts the selected structure to enhance its physical constructability before it is assembled by a real robot.

latter generates 3D mesh structures from textual descriptions. There are ongoing research efforts applying Gen AI for eCAD design of chips [28]). This suggests that Generative AI may have potential for DfRA, and that if coupled with a physical robot, it may be possible in certain cases to fully automate the design cycle.

This paper makes the following contributions:

1) Formulation of a novel problem, Generative Design-for-Robot-Assembly (GDfRA).
2) Blox-Net, a system that combines novel prompting of GPT-4o with a physics simulation, and robot motion planning to automatically address a class of GDfRA problems given available building blocks.
3) Results from experiments suggesting that Blox-Net can produce assemblies that closely resemble the requested object while being stable under gravity throughout the construction process and feasible to be reliably assembled by a robot arm. Starting from singulated objects, Blox-Net achieves 99.2% accuracy in autonomous assembly.

II. Related Work

A. Design for Robot Assembly

The concept of Design for Assembly (DfA) was advanced by Geoffrey Boothroyd and Peter Dewhurst in the early 1980s [29]. Hitachi developed its Assemblability Evaluation Method (AEM) in 1986 [30]. These works laid the foundation for systematic approaches [31] that enhance efficient assembly processes. As robotics advanced, Design for Robot Assembly (DfRA) emerged as an extension of DfA principles, addressing the unique capabilities and limitations of robot systems in industrial assembly tasks [32, 33].

Design for Robot Assembly (DfRA) [32, 34–36] has evolved significantly with the advent of Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) software, which expedite design and evaluation of components and assemblies using Finite Element Methods and perturbation analysis [3–7]. While these tools facilitate visualization and analysis of tolerances, stresses, and forces, all existing DfRA systems require extensive human input [3, 4, 7]. A persistent challenge in DfRA is accurately modeling assembly reliability, given the inherent uncertainties in perception, control, and physics [8–13]. Simulation can partially address this, but struggles to capture 3D deformations and collisions crucial to robot grasping, necessitating iterative real-world testing and redesign [14–18, 37]. ASAP [37] generates physically feasible robotic assembly procedures for complex objects including furniture, tools, and geared mechanisms with many parts. Like Blox-Net, ASAP leverages physics simulation to create executable assembly sequences. Blox-Net goes beyond sequential planning, by also generating the design itself from a text prompt.

Recent advancements also leverage large language models (LLMs) [20, 38] for various aspects of design, including task planning, robot code generation [39, 40], engineering documentation understanding [41], generating planar layouts or CAD models [28, 42–44], and planning for long-horizon assembly tasks [45]. However, these methods primarily focus on determining assembly sequences for given product designs. Emerging startups are using LLMs for design and manufacturing [46, 47]. Blox-Net incorporates both the design and execution aspects of robot assembly, aiming to create physically feasible designs for robotic assembly with minimal human supervision.

B. Text-to-Shape Generation

Semantic generation of realistic 3D shapes and structures is a long-standing problem in computer vision and computer graphics [48]. Deep generative models have enabled a wide range of approaches that learn to capture the distribution of realistic 3D shapes, in the format of voxel maps [49], meshes [50], point clouds [51], sign distance functions [52], CAD models [53], and implicit representations [54]. A growing number of approaches have also been proposed to generate objects and environments to create digital twins or curricula for robotic control and scale up learning [55–62]. With recent advances in aligned text-image representations and large, generative vision-language models [63], researchers seek to generate semantically meaningful shapes specified by natural language instructions [25, 64–66]. Unlike these methods, Blox-Net prompts an LLM (ChatGPT 4o [67]) to generate 3D shapes using a constrained set of building blocks

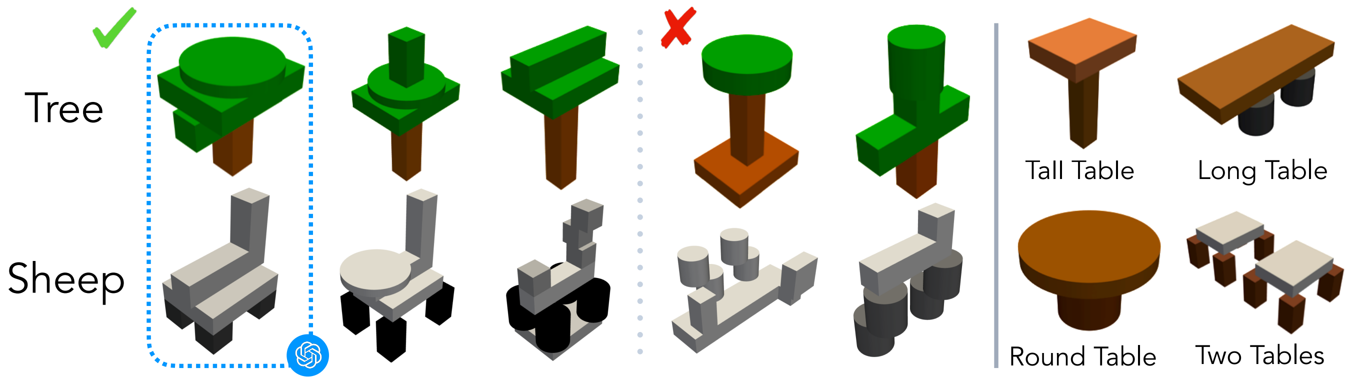

Fig. 3: **Diverse Design Generations:** *Left:* Blox-Net generates a diverse set of candidate designs and uses the VLM (GPT 4o) to select the most suitable one. *Right:* Blox-Net accurately generates a variety of structural designs, adhering to specific input constraints. A set of 10 designs can be generated in 81 seconds, and the selection of the best design takes an additional 60 seconds.

and iteratively generates assembly plans.

C. *Robotic Control with Foundation Models*

Recent advancements in large pre-trained models, such as large language models (LLMs) and vision-language models (VLMs) [20, 63, 68–73], have shown potential to learn generalist robot policies by leveraging vast internet-scale data. These models enable autonomous robotic systems through training on robotics datasets [74–80] or by allowing LLMs to directly generate actions or plans in text or code [79, 81–90]. Rather than directly predicting robot actions, Blox-Net prompts the VLM to generate a construction plan by determining the poses of blocks to form semantically meaningful and physically feasible structures, which are then assembled using motion planning and force feedback control.

III. GDfRA Problem

We formally define the problem of Generative Design for Robotic Assembly (GDfRA). We consider the design of a 3D structure that can be assembled with an industrial robot arm (see Figure 1). The input is a word or phrase (e.g., *"bridge"*) and an image of available components for assembly. The objective for the GDfRA system is to design a structure which is (1) "recognizable" meaning the structure visually resembles the provided text input and (2) "constructible" meaning the structure can be reliably assembled by a robot.

IV. Method

We present Blox-Net, a system for GDfRA that assumes (1) the components are a set of rectilinear and cylindrical blocks, and (2) they are lying in stable poses within the robot's reachable workspace. Blox-Net includes three phases.

A. *Phase I: VLM Design and Selection*

Given the a natural language description of the desired shape (e.g. "giraffe") and a set of blocks with known sizes and shapes, Blox-Net uses a VLM to generate candidate structure designs. Unlike text-to-3D generation methods that produce unconstrained meshes [25, 65], Blox-Net generates 3D structures subject to the physical constraints imposed by the available blocks. It prompts the VLM to generate an assembly plan that specifies the 3D locations and orientations for placing each block using the available components (as shown in Fig. 3). To facilitate high-quality generation, similar to DALL-E 3 [91], Blox-Net first elaborates the prompt. For example, to construct a "giraffe", the VLM is prompted to give a concise, qualitative textual description that conveys the key features of a giraffe by highlighting the overall structure and proportions.

After shape design elaboration, the VLM is prompted for an assembly plan. The prompt includes the target object ("giraffe"), the VLM's elaboration response, and the set of available blocks. The set of available blocks is encoded as JSON, which provides a structured, flexible format familiar to VLM models. Based on these inputs, the VLM is prompted to justify each block's role in the 3D structure.

Blox-Net then prompts the VLM to produce an assembly plan, specifying the rotation, position and color of objects. Instead of using common rotation parameterizations like Euler angles, Blox-Net allows the VLM to specify block orientations by rearranging their width, height, and depth dimensions, applying 90 degree rotations. This, in addition to a yaw angle specified in degrees, provides a simple way to define orientation. Blox-Net also prompts the VLM to output the (x, y) coordinates for each block placement. The blocks are then dropped from a tall height, following the order specified by the VLM. The simulation stops each block's movement upon contact with the ground or another block. This approach ensures that blocks do not intersect and simplifies the action space by eliminating the need to specify the z-coordinate directly.

To enhance stability and correct misplaced blocks, Blox-Net performs iterative prompting, grounded by feedback from PyBullet [92], a real-time physics simulation library with accurate collision detection and rigid-body dynamics. Each block's placement is simulated by dropping it from above. After each placement, the system verifies the structure's stability by checking if a block changed position while simulating gravity. If instability is detected, the block that moved, the direction of movement, and two orthographic views with the unstable block highlighted are included in a prompt sent back to the VLM for correction. This process

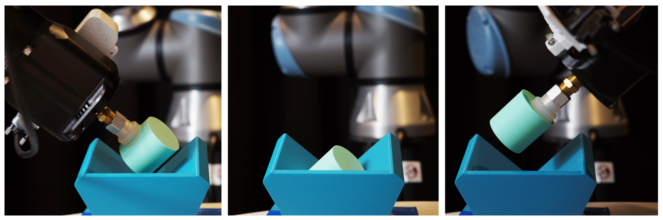

Fig. 4: **Block Reorientation:** To reorient a cylindrical block lying on its curved surface or an incorrectly oriented cuboid, the robot first places the block into a custom 3D printed angle bracket. Then, the block is regrasped on a different face, achieving a 90 degree rotation.

repeats until all blocks are stable or a maximum of two iterations is reached.

This full prompting pipeline is run in parallel with 10 VLM instances, each generating a design candidate. For each design candadite, the VLM is queried with a rendered image from the simulation, and provides a rating from 1 to 5 based on how well the structure resembles the intended design. The top-rated stable design candadites are then paired in a head-to-head comparison, where two images are shown to the VLM, and the design that most closely reflects the original prompt's semantic meaning is selected. This process is repeated in a knockout format (Fig. 3) until a final design is chosen.

B. *Phase II: Perturbation-Based Redesign*

In GDfRA, accounting for imprecise state estimation and robot control is important to ensure robust assembly. The VLM does not have knowledge about such uncertainty, which can result in collisions and misplaced blocks during assembly, so we add a perturbation-based redesign process.

The redesign process iterates through each of the blocks and determines if adjustments are needed. A block will be perturbed if it violates at least one of the following three criterion: (1) the distance to another block is less than a specified collision threshold and the two blocks overlap in the gravity-aligned axis (2) the block is already in collision with another block; or (3) the block is unstable at some nearby sampled point within a predefined radius.

For each block, Blox-Net perturbs its nominal pose by sampling points evenly along regularly spaced, concentric circles centered at the block's initial position and checks for stability and collision at each sampled point. The block position is updated to the average of positions that are stable and free from collision. This process is applied to all blocks in the structure until no further adjustments are needed or each block has been visited a predefined maximum number of times.

C. *Phase III: Robot Assembly and Evaluation*

To evaluate constructability, Blox-Net automates physical assembly, reset, and evaluation using a UR5e robot arm with Robotiq suction gripper. Blox-Net uses SAM [93] to segment a top-down RGBD image of the available blocks and obtain image masks. SAM segmentations include regions that do not correspond to blocks. To filter out these extraneous segmentations, we discard masks that are outside the tray, below a certain minimum area, or not circular or rectangular.

Blox-Net refines each mask to segment the top of each block by fitting a RANSAC [94] plane to the pointcloud and retaining only inliers. Block orientation is determined by fitting the tightest oriented bounding box to the refined mask. Block position is the mean of the points in the filtered point cloud, with the x and y dimensions measured from the point cloud and the z dimension derived from its height relative to the tray base. Blocks may require rotations about their x or y axis to align with the pose used in the plan. This rotation is facilitated by placing the block in a 90-degree angle bracket and regrasping the block from a different side (Fig. 4).

To assemble the final structure, Blox-Net uses custom robot motion planning to generate UR5e arm commands such that each block is grasped at its centroid, rotated to the planned orientation, and placed at the location specified in the design. Force feedback control is used for both grasping and placing blocks: during each grasp, the robot lowers onto the block with the suction gripper activated until a small (8N) force is detected; similarly, during placement, it descends and stops the suction flow to release the block once a small force is sensed.

To automate robot assembly evaluation, Blox-Net captures an image of the completed assembly after each trial. Then, Blox-Net presses down on a tilt plate, dumping the blocks back into the tray. This resets blocks for subsequent trials.

V. Experiments

To evaluate how well the generated structures satisfy the GDfRA objective, we assess both the *semantic recognizability* of the designs (in Section V-B), which refers to how well the designs semantically match the input shape description, and their *constructability* (in Section V-C), which refers to how reliably they are constructed by a real robot. Additionally, we evaluate the effectiveness of the perturbation redesign (in Section V-D). To identify natural language 3D shape descriptions for evaluation, we prompt GPT-4o to generate a list of 200 object names spanning categories such as furniture, alphabet letters, architecture, and animals. We run Blox-Net's design generation (Section IV-A) on all 200 objects using a fixed set of block shapes and dimensions. We evaluate semantic recognizability on all 200 designs and evaluate real-world assembly on a representative subset of 11 designs, to illustrate the capabilities and limitations of Blox-Net using a physical robot. Ranging from simple (as few as three blocks) to complex (up to 10 blocks), these structures show how Blox-Net creatively uses the limited set of provided blocks. The selected designs make use of a variety of block types, and feature blocks arranged in different orientations, providing a rigorous test of Blox-net's physical setup. We selected six designs to evaluate the reliability of the full, end-to-end pipeline (including automated block reorientation and reset) and five designs to demonstrate the effectiveness of perturbation redesign.

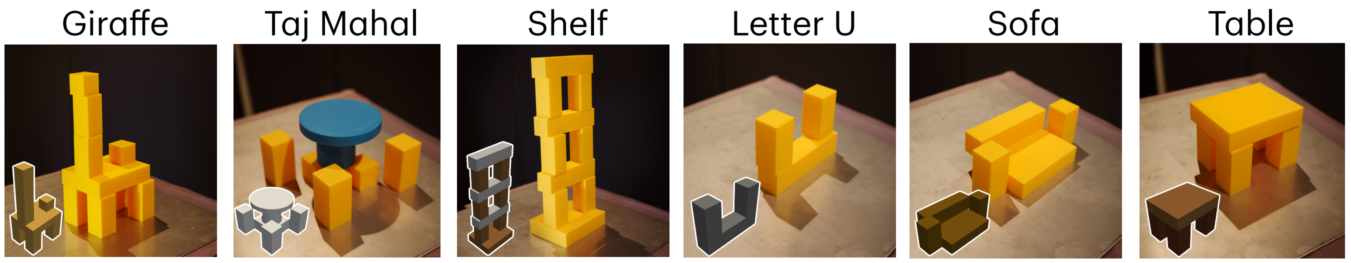

Fig. 5: **Task Execution:** We present Blox-Net VLM generated designs assembled by a robot paired with simulation renderings

A. Implementation Details

Blox-Net is implemented with the following components: GPT-4o, PyBullet, UR5e robot arm, Robotiq suction gripper, Zed Mini Stereo Camera, and 3D printed cuboidal and cylindrical blocks. We use PyBullet as a simulator and define simulation parameters as: uniform object density of $1000\,\mathrm{kg/m^3}$, lateral friction coefficient of 0.5, spinning friction coefficient of 0.2, gravity of $-9.81\,\mathrm{m/s^2}$. A block is classified as unstable if, after 2 seconds, the block's position deviated by more than $1\mathrm{cm}$ or rotated by more than $.1$ radians from its starting position.

During perturbation redesign, Blox-Net samples 8 points from 10 concentric circles with radii from $1\mathrm{mm}$ to $15\mathrm{mm}$ and each block is perturbed a maximum of 10 times. Blox-Net filters SAM masks from the top-down image by shape by fitting a minimum area bounding rectangle and minimum bounding circle to each mask. Masks are discarded if their area is less than 80% of the areas of both bounding shapes.

B. Semantic Recognizability

To measure how well the generated structure resembles the requested language description, we use GPT-4o as an evaluator to assess the semantic distinctiveness and accuracy of each design, following methodologies similar to those used in VLM answer scoring [95–97]. In this experiment, we use a set of N object labels, where N includes the correct label alongside $N-1$ randomly selected distractor labels from the pool of 200 objects. We provide GPT-4o with a rendered image of the generated assembly and the N labels in random order, and task the VLM with ranking these labels based on how well each one matches the image. In Table I, we report the percentage of correct Top-1 predictions, and for imperfect guesses, we analyze the average ranking of the correct label within GPT-4o's ordered list, (where a ranking of 1 is best). Additionally, we report the average ranking relative to N, with results presented for Top-1 accuracy and average ranking for N=5, 10, 15, 20. Fig. 6 shows failure modes for the design generation stage. We observe that failures can often be grouped into three categories: the use of blocks not provided in the block set, misoriented blocks, and incorrect spatial reasoning (where structures have significant overhangs or unsupported blocks).

C. Constructability

In Table II, we report constructability on a real robot over 10 trials on 6 designs selected to highlight diversity. For each trial, we record the % of blocks correctly positioned at the time of placement, and the % of trials where the structure is fully successfully assembled. To assess the system's autonomy, we track human interventions while running trials. Impressively, Blox-Net is capable of performing fully autonomous, end-to-end assembly including identifying blocks in the tray and placing them in the correct positions for assembly. Interventions are necessary when blocks in the tray occlude one another, are not adequately separated, or fall out of the tray. Additionally, if, after reorientation, the robot fails to regrasp a block, the block will be manually reoriented. Human interventions are *only* required during the reset phase between trials and never during the assembly process.

N	Top-1 Accuracy	Avg. Ranking	Relative Ranking
5	63.5%	1.70	34.0%
10	48.5%	2.92	29.1%
15	46.0%	3.68	24.5%
20	41.5%	5.05	25.3%

TABLE I: **VLM-Based Design Recognizability**: Top-1 accuracy, average ranking, and relative ranking based on GPT-4o responses averaged across all 200 objects. Relative ranking is reported as the average ranking divided by N where N is the number of labels.

D. Perturbation Redesign Ablation

In Table III, we evaluate the effect of phase II, perturbation redesign, (Section IV-B) on construction success. We conduct experiments on 5 objects, each assembled 10 times with and without perturbation redesign. Each trial begins with all blocks singulated and in the correct orientation to isolate the effect of perturbation design. Assembly is performed fully autonomously. During evaluation, we observe that incorrectly placed blocks are occasionally nudged into or out of their correct position during later placements. Thus, we report the percentage of blocks correctly placed at the time of their placement, the average percentage of blocks in the correct location at the end of each trial, and the percentage of trials where the structure is fully completed. We observe a significant performance decrease across all objects in all metrics when constructing without perturbation redesign, demonstrating its impact on design success.

E. Quantitative Results

Semantic Shape Recognizability: We present experimental results in Table I. Results from the evaluation of Blox-Net's designs using GPT-4o as an evaluator suggest that the generated designs closely resemble the desired shape descriptions as evaluated by GPT-4o. Notably, with $N=5$ random shape labels, Blox-Net achieves a Top-1 accuracy of 63.5%. We observe that the most common failure modes for this metric

Assembly Design (# of Blocks)	% of Blocks Adjusted During Reset Phase	% Correct Blocks Placed	% of Assemblies Completed
Filament Roll (3)	14%	100%	100%
Giraffe (9)	28%	100%	100%
Lighthouse (7)	18%	100%	100%
Letter-U (3)	7%	100%	100%
Shelf (10)	34%	100%	100%
Table (5)	11%	98%	90%

TABLE II: **Robot Assembly Reliability during 10 trials**: The table presents the robot assembly results for six designs, each assembled by the robot over 10 trials with automated reset and reorientation.

Object	% Correct Blocks Placed		% Correct in End State		% Full Completion	
	✗	✓	✗	✓	✗	✓
Ceiling Fan (7)	66.7%	**100%**	91.0%	**100%**	40%	**100%**
Sandbox (5)	50.0%	**96%**	50.0%	**96%**	20%	**80%**
Shark (6)	75.0%	**100%**	76.7%	**100%**	40%	**100%**
Sofa (4)	72.5%	**100%**	72.5%	**100%**	30%	**100%**
Taj Mahal (10)	71.0%	**100%**	72.0%	**100%**	10%	**100%**
Average	67.1%	**99.2%**	72.4%	**99.2%**	28.0%	**96.0%**

TABLE III: **Perturbation-based Redesign Ablation**: We run each design for 10 iterations. ✗ indicates experiments **without** perturbation redesign and ✓ indicates experiments **with** perturbation redesign. **% Correct Blocks Placed**: the percent of blocks that were placed correctly (determined by a group) to the whole structure. **% Correct In End State**: the percent of blocks that remain in their correct pose at the end (blocks may be knocked down by later placements). **% Full Completion**: the percent of overall success (0 or 1 for the whole structure per run over all 10 runs).

arise from overly simple, complex, or abstract geometries. For instance, Blox-Net's generation of a "barrier" can easily be confused with a brick, tablet, board, or panel, since all are planar, rectangular structures. Conversely, for highly complex structures, such as "airport", Blox-Net is unable to recreate the object's intricate geometry using a limited blockset, leading to poor evaluations. Finally, for abstract concepts, such as "universe", Blox-Net struggles to convert the prompt into a semantically meaningful structure.

Constructability: Results are in Table II. All designs are reliably assembled by the robot without human intervention during assembly. Five of six designs achieve a perfect assembly completion rate in 10 out of 10 trials. Human interventions, which occur only during the reset phase, are needed on average for 22.5% of blocks to assist with singulation and reorientation.

Perturbation Redesign Ablation: Results are summarized in Table III. The Blox-Net simulation-based perturbation redesign greatly improves all three metrics across all 5 designs. Fig. 7 shows examples of failed construction when ablating perturbation redesign. When blocks are directly adjacent to one another, we observe that minor inaccuracies in grasping or block rotation cause incorrect placement and an incorrect final structure. Since blocks are placed sequentially on top of one another, a single error in placement can lead to multiple blocks ending up int he incorrect final pose, such as in the Taj Mahal example shown. Remarkably, perturbation redesign improves the full completion success rate by an average of 4x. By spreading blocks further apart, this technique significantly lowers the likelihood of such failure modes, while preserving the semantic recognizability of the structure.

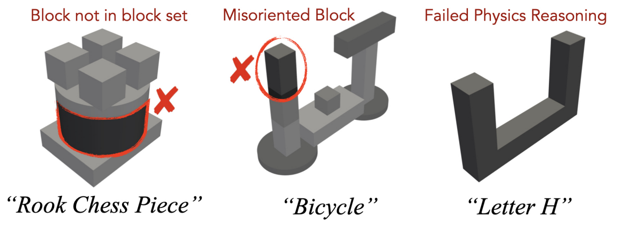

Fig. 6: **VLM Generation Failures** Blox-Net's design generation occasionally produces designs that: include unavailable blocks (Rook Chess Piece), incorrectly orient blocks (Bicycle's Handlebar), or fail to account for gravity (Letter H).

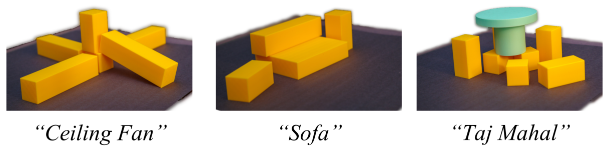

Fig. 7: **Perturbation Redesign Ablation Failures** Omitting perturbation redesign from the Blox-Net leads to a significant increase in physical construction failures. Small inaccuracies in block placement result in collisions, fallen blocks, and structural collapses.

VI. Limitations and Conclusion

Blox-Net shows promising results for this instance of GDfRA. However, limitations exist that restrict its performance. Blox-Net is limited to non-deformable cuboid and cylindrical blocks, limiting its ability to represent geometric shapes. Some of its assembly designs are not clearly recognizable, due in part to these block limitations. The current Blox-Net system uses only a suction-based gripper and sometimes requires human intervention during resetting, reducing assembly efficacy. In future work, we will study the performance of different VLMs on larger blocksets, including chain-of-thought models, such as GPT-o1 and Deepseek R1 [98], and we will apply GDfRA to industrial tasks and deformable object assembly.

This paper introduces Blox-Net, a novel system addressing a version of the Generative Design-for-Robot-Assembly problem using a three-phase approach: creating the initial designs by prompting a vision language model, conducting simulation-based analysis for constructability, and utilizing a physical robot for assembly evaluation. Experiment results suggest that Blox-Net can bridge the gap between abstract design concepts and robot-executable assemblies. Remarkably, five Blox-Net assembly designs, each using 3 to 10 blocks and scoring high in recognizability, were successfully assembled 10 consecutive times by the robot without human intervention.

Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab.

REFERENCES

[1] E. Foner, *Give Me Liberty! An American History: Seagull Fourth Edition*. WW Norton & Company, 2013, vol. 1.

[2] G. Boothroyd, *Assembly automation and product design*. crc press, 2005.

[3] R. W. Kennard and L. A. Stone, "Computer aided design of experiments," *Technometrics*, vol. 11, no. 1, pp. 137–148, 1969.

[4] T.-C. Chang, R. A. Wysk, and H.-P. Wang, *Computer-aided manufacturing*. Prentice-Hall, Inc., 1991.

[5] K. Millheim, S. Jordan, and C. Ritter, "Bottom-hole assembly analysis using the finite-element method," *Journal of Petroleum Technology*, vol. 30, no. 02, pp. 265–274, 1978.

[6] S. Lee and H. H. Asada, "A perturbation/correlation method for force guided robot assembly," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 764–773, 1999.

[7] Z. Bi and X. Wang, *Computer aided design and manufacturing*. John Wiley & Sons, 2020.

[8] S. Y. Nof, *Handbook of industrial robotics*. John Wiley & Sons, 1999.

[9] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, no. 2, pp. 201–225, 1993.

[10] A. A. Apolinarska *et al.*, "Robotic assembly of timber joints using reinforcement learning," *Automation in Construction*, vol. 125, p. 103 569, 2021.

[11] Y. Tian *et al.*, "Assemble them all: Physics-based planning for generalizable assembly by disassembly," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–11, 2022.

[12] J. Luo and H. Li, "A learning approach to robot-agnostic force-guided high precision assembly," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 2151–2157.

[13] L. Fu, H. Huang, L. Berscheid, H. Li, K. Goldberg, and S. Chitta, "Safe self-supervised learning in real of visuo-tactile feedback policies for industrial insertion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 10 380–10 386.

[14] J. Xu, M. Danielczuk, E. Steinbach, and K. Goldberg, "6dfc: Efficiently planning soft non-planar area contact grasps using 6d friction cones," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 7891–7897.

[15] C. M. Kim, M. Danielczuk, I. Huang, and K. Goldberg, "Ipc-graspsim: Reducing the sim2real gap for parallel-jaw grasping with the incremental potential contact model," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6180–6187.

[16] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[17] H. Huang *et al.*, "Mechanical search on shelves with efficient stacking and destacking of objects," in *The International Symposium of Robotics Research*, Springer, 2022, pp. 205–221.

[18] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *2018 IEEE International Conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 5620–5627.

[19] A. Ramesh *et al.*, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[20] T. Brown *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[21] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[22] J. Ho *et al.*, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[23] T. Brooks *et al.*, "Generating long videos of dynamic scenes," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 769–31 781, 2022.

[24] L. Castrejon, N. Ballas, and A. Courville, "Improved conditional vrnns for video prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7608–7617.

[25] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," in *International Conference on Learning Representations (ICLR)*, 2023.

[26] C.-H. Lin *et al.*, "Magic3d: High-resolution text-to-3d content creation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[27] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich, "Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 619–12 629.

[28] M. Liu *et al.*, "Chipnemo: Domain-adapted llms for chip design," *arXiv preprint arXiv:2311.00176*, 2023.

[29] G. Boothroyd and P. Dewhurst, *Design for Assembly - A Designer's Handbook*. Amherst, MA: University of Massachusetts, 1983.

[30] S. Miyakawa, "The hitachi assemblability evaluation mrthod (aem)," in *Proceedings of 1st Int. Conf. on Product Design for Assembly, 1986*, 1986.

[31] General Electric Co., *Manufacturing Producibility Handbook*. Schenectady, NY: Manufacturing Services, General Electric Co., 1960.

[32] S. Y. Nof, W. E. Wilhelm, and H.-J. Warnecke, "Design for assembly," in *Industrial Assembly*. Boston, MA: Springer US, 1997, pp. 84–134.

[33] G. Boothroyd, "Design for assembly—the key to design for manufacture," *The International Journal of Advanced Manufacturing Technology*, vol. 2, pp. 3–11, 1987.

[34] T. Ohashi, M. Iwata, S. Arimoto, and S. Miyakawa, "Extended assemblability evaluation method (aem)(extended quantitative assembly producibility evaluation for assembled parts and products)," *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, vol. 45, no. 2, pp. 567–574, 2002.

[35] G. Boothroyd, P. Dewhurst, and W. A. Knight, *Product design for manufacture and assembly*. CRC press, 2010.

[36] H. K. Rampersad, *Integrated and Simultaneous Design for Robotic Assembly: Product Development, Planning...* John Wiley & Sons, Inc., 1994.

[37] Y. Tian *et al.*, "Asap: Automated sequence planning for complex robotic assembly with physical feasibility," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[38] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[39] A. Macaluso, N. Cote, and S. Chitta, "Toward automated programming for robotic assembly using chatgpt," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[40] H. You, Y. Ye, T. Zhou, Q. Zhu, and J. Du, "Robot-enabled construction assembly with automated sequence planning based on chatgpt: Robogpt," *Buildings*, vol. 13, no. 7, p. 1772, 2023.

[41] A. C. Doris *et al.*, "Designqa: A multimodal benchmark for evaluating large language models' understanding of engineering documentation," *Journal of Computing and Information Science in Engineering*, vol. 25, no. 2, p. 021 009, 2025.

[42] A. Gaier, J. Stoddart, L. Villaggi, and S. Sudhakaran, "Generative design through quality-diversity data synthesis and language models," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 823–831.

[43] A. Badagabettu, S. S. Yarlagadda, and A. B. Farimani, "Query2cad: Generating cad models using natural language queries," *arXiv preprint arXiv:2406.00144*, 2024.

[44] S. Wu *et al.*, "Cad-llm: Large language model for cad generation," in *Proceedings of the neural information processing systems conference. neurIPS*, 2023.

[45] Y. Feng, J. Han, Z. Yang, X. Yue, S. Levine, and J. Luo, *Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation*, 2025. arXiv: 2502.16707 [cs.RO].

[46] J. Lessin, *Larry page has a new ai startup — the information*, 2025.

[47] *Adam: AI Powered CAD*, Accessed: 2025-03-11, 2025.

[48] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, Aug. 2009.

[49] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *29th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[50] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018.

[51] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,

Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2017, pp. 2463–2471.

[52] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[53] H. Izadinia, Q. Shan, and S. M. Seitz, "Im2cad," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, 2017.

[54] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[55] A. Kar *et al.*, "Meta-sim: Learning to generate synthetic datasets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[56] K. Fang, Y. Zhu, S. Savarese, and L. Fei-Fei, "Adaptive procedural task generation for hard-exploration problems," *International Conference on Representation Learning (ICLR)*, 2021.

[57] K. Fang, Y. Zhu, S. Savarese, and L. Fei-Fei, "Learning generalizable skills via automated generation of diverse tasks," *Robotics: Science and Systems (RSS)*, 2021.

[58] K. Fang, T. Migimatsu, A. Mandlekar, L. Fei-Fei, and J. Bohg, "Active task randomization: Learning robust skills via unsupervised generation of diverse and feasible tasks," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2022.

[59] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[60] C.-C. Hsu, Z. Jiang, and Y. Zhu, "Ditto in the house: Building articulation models of indoor scenes through interactive perception," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[61] L. Wang *et al.*, "Gensim: Generating robotic simulation tasks via large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[62] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Proceedings of The 7th Conference on Robot Learning*, J. Tan, M. Toussaint, and K. Darvish, Eds., ser. Proceedings of Machine Learning Research, vol. 229, PMLR, Jun. 2023, pp. 3766–3777.

[63] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[64] K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, and S. Savarese, "Text2shape: Generating shapes from natural language by learning joint embeddings," in *Computer Vision – ACCV 2018*, C. V. Jawahar, H. Li, G. Mori, and K. Schindler, Eds., Springer International Publishing, 2019.

[65] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, "Zero-shot text-guided object generation with dream fields," *CVPR*, 2022.

[66] A. Haque, M. Tancik, A. A. Efros, A. Holynski, and A. Kanazawa, "Instruct-nerf2nerf: Editing 3d scenes with instructions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 740–19 750.

[67] OpenAI, *Gpt-4o system card*, https://cdn.openai.com/gpt-4o-system-card.pdf, Accessed: 2024-09-14, 2024.

[68] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019.

[69] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[70] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[71] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[72] J. Achiam *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[73] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *International Conference on Machine Learning*, 2023.

[74] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese, "Scene memory transformer for embodied agents in long-horizon tasks," *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[75] A. Brohan *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[76] A. Brohan *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[77] Y. Jiang *et al.*, "Vima: General robot manipulation with multimodal prompts," in *Fortieth International Conference on Machine Learning*, 2023.

[78] Octo Model Team *et al.*, *Octo: An open-source generalist robot policy*, https://octo-models.github.io, 2023.

[79] K. Fang, F. Liu, P. Abbeel, and S. Levine, "MOKA: Open-World Robotic Manipulation through Mark-Based Visual Prompting," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, Jul. 2024.

[80] L. Fu *et al.*, "In-context imitation learning via next-token prediction," in *International Conference on Robotics and Automation (ICRA)*, 2025.

[81] M. Ahn *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[82] W. Huang *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

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

[84] B. Chen *et al.*, "Open-vocabulary queryable scene representations for real world planning," in *IEEE International Conference on Robotics and Automation*, IEEE, 2023, pp. 11 509–11 522.

[85] J. Liang *et al.*, "Code as policies: Language model programs for embodied control," in *IEEE International Conference on Robotics and Automation*, IEEE, 2023, pp. 9493–9500.

[86] I. Singh *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *IEEE International Conference on Robotics and Automation*, IEEE, 2023, pp. 11 523–11 530.

[87] G. Wang *et al.*, "Voyager: An open-ended embodied agent with large language models," in *Neural Information Processing Systems (NIPS)*, 2024.

[88] S. Mirchandani *et al.*, "Large language models as general pattern machines," in *Conference on Robot Learning (CoRL)*, 2023.

[89] G. Tang, S. Rajkumar, Y. Zhou, H. R. Walke, S. Levine, and K. Fang, *Kalie: Fine-tuning vision-language models for open-world manipulation without robot data*, 2024. arXiv: 2409.14066 [cs.RO].

[90] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, Nov. 2023.

[91] J. Betker *et al.*, "Improving image generation with better captions," *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, vol. 2, no. 3, p. 8, 2023.

[92] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, http://pybullet.org, 2016–2021.

[93] A. Kirillov *et al.*, "Segment anything," 2023.

[94] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[95] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *NeurIPS*, 2023.

[96] W.-L. Chiang *et al.*, *Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality*, Mar. 2023.

[97] L. Fu *et al.*, "A touch, vision, and language dataset for multimodal alignment," in *Forty-first International Conference on Machine Learning*, 2024.

[98] DeepSeek-AI *et al.*, *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*, 2025. arXiv: 2501.12948 [cs.CL].