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 rich history 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, and instructions for a robot to build this assembly. The output 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 with minimal human supervision. Blox-Net achieved a Top-1 accuracy of 63.5% in the semantic accuracy of its designed assemblies. These designs, after 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. Surprisingly, this entire design process from the textual word to reliable physical assembly is performed with zero human intervention.**

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 latter half of 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 ensure feasibility, for example designing an injection molded part along with a custom workcell for manipulating it. These design systems were enhanced by the emergence of Computer-Aided Design (CAD) and Computer-Aided-Manufacturing (CAM) software that streamlined 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].

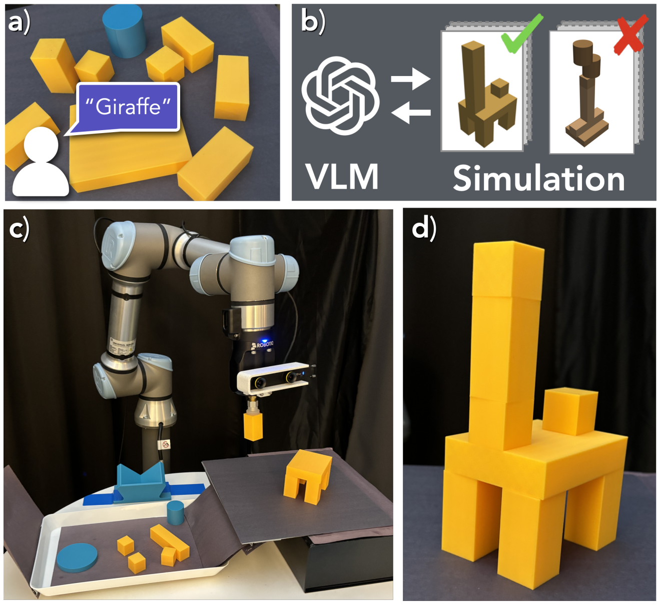


Fig. 1: *Can a vision-language model generate designs suitable for robot assembly?* **Blox-Net** is a GDfRA system that produces 3D designs constructible by robots subject to physical material constraints. **(a)** Starting with a phrase (e.g., "giraffe") and a set of blocks, **(b)** Blox-Net iteratively 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 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 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 latter generates 3D mesh structures from textual descriptions (and there are ongoing research efforts applying Gen AI for eCAD design of chips [28]). This suggests that Generative

[1]The AUTOLab at UC Berkeley, [2]Cornell University

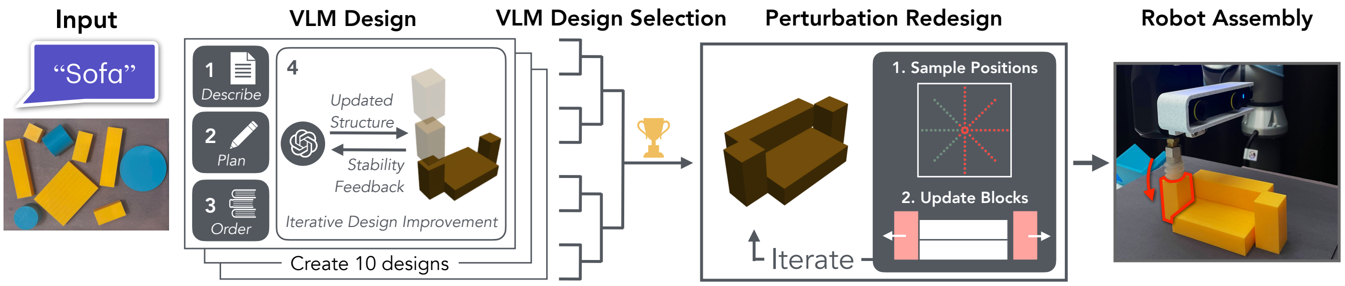


Fig. 2: **Overview of Blox-Net.** We present a multi-stage framework for producing physically constructible models based on a user-specified prompt. The Blox-Net pipeline begins with a natural language input and JSON detailing the available blocks. These parameters are passed into a series of VLM prompts, 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.

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.

In this paper, we propose Blox-Net, a fully-implemented *generative* DfRA (GDfRA) system that combines the semantic and text generation capabilities of large language models (LLM) with physical analysis from a simulator. Blox-Net includes 3 phases: 1) A vision language model (VLM) with customized iterative prompting to design a feasible 3D arrangement of the available components – an assembly – that approximates the shape of the desired object (eg "a giraffe"); 2) simulation with perturbation analysis to evaluate this assembly in terms of physical robot constructability and to revise the assembly accordingly; 3) Computer vision, motion planning, and control of a physical robot with a camera to repeatedly, through an automated reset, construct this assembly with the given components to automatically evaluate physical assembly reliability.

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 prompting with a physical robot, physics simulation, and 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% success rates.

II. Related Work

A. Design for Robot Assembly

The concept of Design for Assembly (DfA) was pioneered by Geoffrey Boothroyd and Peter Dewhurst in the early 1980s [29], with Hitachi developing its Assemblability Evaluation Method (AEM) in 1986 [30]. These seminal works laid the foundation for systematic approaches that follow product design guidelines [31] facilitate facilitate efficient assembly processes. As robotics automation in manufacturing became prevalent, Design for Robot Assembly (DfRA) emerged as an extension of DfA principles, specifically addressing the unique capabilities and limitations of robotic systems in 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]. Recent advancements leverage large language models (LLMs) [20, 38] for various aspects of design, including task planning, robot code generation [39, 40], engineering documentation understanding [41], and generating planar layouts or CAD models [28, 42–44]. However, these methods primarily focus on determining assembly sequences for fixed designs. In contrast, we tackle 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 3D shapes and structures is a long-standing problem in computer vision and computer graphics [45]. 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 [46], meshes [47], point clouds [48], sign distance functions [49], CAD models [50], and implicit representations [51]. 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 [52–59]. With the advances of aligned text-image representations and

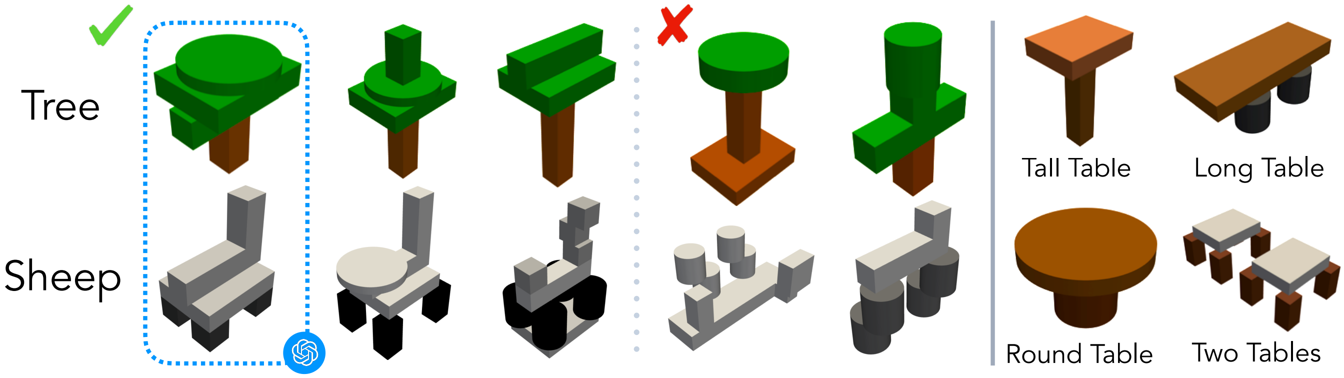


Fig. 3: **Diverse Design Generations:** *Left:* Blox-Net generates a diverse set of candidate designs and uses the VLM (GPT 4o) to select the most suitable one. *Right:* Blox-Net accurately generates a variety of structural designs, adhering to specific input constraints. A set of 10 designs can be generated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

vision-language models, an increasing number of works have aimed to generate semantically meaningful shapes specified by natural language instructions [25, 60–62]. Unlike these methods, based on the available physical building blocks, Blox-Net generates 3D shapes by prompting an LLM (ChatGPT 4o [63]) and then generates a plan for assembling the blocks to construct the desired shape.

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, 64–70], have significantly impacted robotics task planning by leveraging vast internet-scale data. These models enable end-to-end learning through training on robotics datasets [71–77] or allow LLMs to directly generate actions or plans in text or code [76, 78–87]. Rather than focusing on motion or waypoint planning, 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 assembled by a robot.

IV. Method

We present Blox-Net, a system for GDfRA that assumes (1) components are cuboids and cylinders and (2) components are lying in stable poses within a reachable planar area.

Blox-Net includes three phases. In phase I (Figure 3), Blox-Net prompts a VLM (GPT-4o [63]) to generate multiple assembly designs, from which the VLM selects the top candidate based on stability and visual fidelity. In phase II (Section IV-B), the chosen assembly design undergoes an iterative refinement process in a customized physics simulator. This simulation-based approach applies controlled perturbations to enhance the design's constructability while maintaining its core characteristics. In phase III (Section IV-C), Blox-Net utilizes a robot arm equipped with a wrist-mounted stereo camera and suction gripper to construct the optimized design using 3D printed blocks. The assembly is constructed on a tilt plate, which the robot actuates to automatically reset the blocks back into a tray.

A. Phase I: VLM Design and Selection

Given the language description and a set of blocks with known sizes and shapes, Blox-Net uses a VLM to generate candidate structure designs. Unlike existing text-to-3D generation methods that produce unconstrained meshes [25, 61], 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 [88], 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 prompt elaboration, the VLM is prompted for the assembly plan. Specifically, 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 asked to explain each block's role in the structure.

Once a high-level plan is generated, Blox-Net prompts the VLM to produce an assembly plan, specifying the rotation, position and color of objects. Instead of using common rotation parametrizations like Euler angles, Blox-Net instructs the VLM to rotate blocks by rearranging their dimensions directly, thereby providing a more simple interface for specifying orientation. Next, Blox-Net prompts the VLM to output the (x, y) coordinates for block placement. Limiting the specification to (x, y) coordinates rather than (x, y, z) simplifies the action space and avoids potential

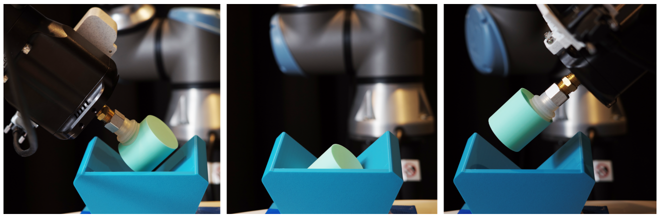


Fig. 4: **Block Reorientation:** The robot first places the block into a 90 degree angle bracket. Then, the block is regrasped on a different face, achieving a 90 degree rotation.

issues with blocks being placed inside one another. Blocks are placed by dropping them in the order.

To enhance stability and correct misplaced blocks, Blox-Net performs iterative, simulation-in-the-loop prompting. Each block's placement is simulated by dropping it in simulation from above the structure. After each placement, the system checks the block for stability. If instability is detected, details such as the specific block that moved, the direction of movement, and two orthographic views highlighting the unstable block are included in a prompt sent back to the VLM for correction. This process continues until all blocks are stable or a maximum of two iterations is reached.

This full prompting pipeline is run in parallel, generating 10 design candidates. For each design, 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 designs are then paired in a head-to-head comparison, where two images are shown to the VLM, and it selects the more recognizable design. 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 tolerances, which can result in collisions and misplaced blocks during assembly. We thus introduce 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 surface-to-surface 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 samples points evenly along regularly spaced, concentric circles centered at the block nominal location and checks for stability and collision at each 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 and evaluates the generated design on a robot. The robot first moves to a predefined pose and captures a top-down RGBD image of the blocks on a plastic tray. Blox-Net uses SAM [89] to segment an RGB image and obtain image masks. SAM segmentations include regions that do not correspond to blocks. To filter out extraneous masks, we generate a point cloud for each mask by deprojecting the masked area from the depth image obtained from a stereo camera [90]. we then 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 [91] plane to the pointcloud and retaining only inliers. The block's rotation is determined by fitting the tightest oriented bounding box to the refined mask. The block's center 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.

Upon determining the size, shape, position, and dimensions for each block, Blox-Net can obtain a new plan through the design generation and perturbation-based redesign process (Section IV-A and Section IV-B), or construct the target object based on a previously generated plan. 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). After reorientation, the robot captures a new top-down image and all block positions, rotations, shapes, and dimensions are recomputed via the aforementioned pipeline.

The assembly process begins after all blocks are properly oriented. 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 a grasp, the robot lowers onto the block until a force is detected; similarly, during placement, it descends and releases the block once a force is sensed.

To enable efficient testing and design validation, we design an *automatic* reset. After completing the full assembly, the robot arm captures an image. Then, the robot presses down on the tilt plate, dumping the blocks back into the tray. This resets the scene for subsequent trials.

V. Experiments

To evaluate how well the generated structures by Blox-Net satisfy the GDfRA objective, we assess both the *semantic recognizability* of the designs (in Section V-A), which refers to how well the designs semantically align with the prompts, and their *constructability* (in Section V-B), which refers to how reliably they can be constructed by a real robot. Additionally, we evaluate the effectiveness of the perturbation redesign (in Section V-C). To create a candidate objects list for evaluation, we prompt GPT-4o to generate a list of 200 objects spanning categories such as furniture, alphabet letters, architecture, and animals. We run Blox-Net's design generation (Section IV-A) on all objects using a fixed set of block shapes and dimensions. We evaluate semantic recognizability on all 200 designs and evaluate

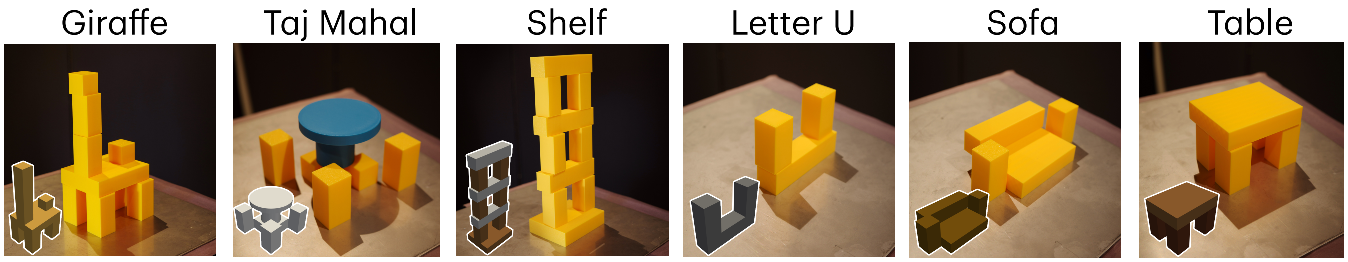


Fig. 5: **Task Execution:** We present Blox-Net VLM generated designs assembled by a robot paired with simulation renderings

N	Top-1 Accuracy	Avg. Ranking	Relative Ranking
5	63.5%	1.7	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.

constructability on a representative subset of 11 designs, which showcase the capabilities and limitations of Blox-Net, using a physical robot. Additionally, we evaluate the effectiveness of perturbation redesign on 5 designs.

A. Semantic Recognizability

To measure how well the generated structure resembles the requested language description, we design an experiment using 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 [92–94]. 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. 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.

B. Constructability

We measure 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. These experiments incorporate automated reset, block reorientation, and assembly fully end-to-end. To assess the system's autonomy, we track the average percentage of blocks per trial which require intervention during the reset phase, where an intervention is counted for each block moved. In failure cases after reset where blocks occlude each other, are not adequately separated, or fall out of the tray, blocks are repositioned and placed back in their same stable pose. In cases where the robot fails to regrasp a block during reorientation, the block is placed back in the stable pose corresponding to its final stable pose in the structure. Human interventions are *only* performed during resetting but not during the assembly process.

Object (# 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 and Reset**: The table presents the robot assembly results for six designs, each assembled by the robot over 10 trials following a reset, during which all blocks are singulated and reoriented on a plastic tray. Human intervention occurred only during the reset phase to de-stack, singulate, and reorient blocks.

C. Perturbation Redesign Ablation

We evaluate the effect of 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 their correct stable pose. This isolates the effect of perturbation redesign by eliminating influence from prior assembly states. Assembly is performed fully autonomously. 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.

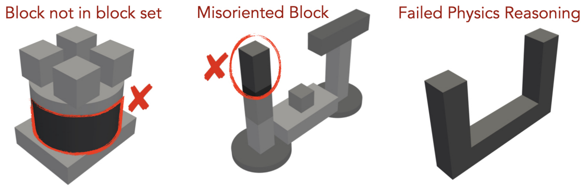
D. Implementation Details

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 a 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 measured as unstable if after 500 simulation steps at 240Hz the block's position deviates by more than 1cm or is rotated by more than .1 radians from its starting position.

During perturbation redesign, Blox-Net samples 8 points from each of 10 concentric circles with radii from 1mm to 15mm and each block is perturbed a maximum of 10 times. Blox-Net filters masks 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.

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 Redesign**: Each object is followed by the number of blocks in the design enclosed in parenthesis. We run each design for 10 iterations. ✗ indicates experiments **without** perturbation redesign and ✓ indicates experiments **with** perturbation redesign. **% Correct Blocks Placed**: the ratio of blocks that were placed correctly (determined by a group) to the whole structure. **% Correct In End State**: the ratio of blocks that remain in their correct pose at the end (blocks may be knocked down by later placements). **% Full Completion**: the ratio of overall success (0 or 1 for the whole structure per run over all 10 runs). We observe a significant performance decrease across all objects in all metrics when constructing without perturbation redesign, demonstrating its impact on design success.



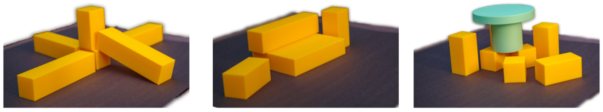
"Rook Chess Piece" *"Bicycle"* *"Letter H"*

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).

E. Quantitative Results

Semantic Recognizability: We present results in Table I. Results from the evaluation of BloxNet's designs using GPT-4o as an evaluator suggest that the generated designs closely align with the correct category semantics as recognized by GPT-4o. Notably, with $N = 5$ labels, the model achieves a Top-1 accuracy of 63.5%, demonstrating a consistent correspondence between the generated designs and the intended prompts. Importantly, even with larger label sets, the model maintains a reasonable average ranking, with the correct label placed consistently near the top. This suggests that the generated designs remain recognizable, even among a large pool of designs.

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, and all designs achieve a 98%+ placement success rate, highlighting Blox-Net's ability to assemble complex structures. Human interventions, which occur only during the reset phase, are sometimes needed to singulate or reorient blocks. Complex structures, such as the Giraffe (9 blocks) or shelf (10 blocks), have more human reset interventions due to an increasing likelihood of overlapping, non-singulated, or misoriented blocks.



"Ceiling Fan" *"Sofa"* *"Taj Mahal"*

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.

Perturbation Redesign Ablation Results are summarized in Table III. Perturbation redesign greatly improves all three metrics across all 5 designs to near-perfect. The percentage of correctly placed blocks and the percentage of correctness in the end state are similar for all objects except the ceiling fan. While incorrect block placements typically lead to errors in the final structure, later block placements sometimes correct these errors. Perturbation redesign improves the full completion success rate by an average of 4x. Overall, perturbation redesign significantly enhances the robustness of the assembly process by accommodating slight imprecisions, leading to more reliable and accurate final structures across a variety of designs.

VI. Limitations and Conclusion

While Blox-Net shows promising results in constrained 3D structure generation, it is limited to non-deformable cuboid and cylinder blocks, restricting geometric diversity and reducing Blox-Net's ability to represent complex shapes. Many assembly designs are still not clearly recognizable, likely due in part to these block limitations. The system uses only a suction-based gripper, without accounting for gripper width or slanted surfaces, and sometimes requires human intervention during resetting, reducing assembly efficacy.

This paper introduces Blox-Net, a novel system addressing 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 any human intervention.

Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab. The authors were supported in part by donations from Toyota Research Institute, Bosch, Google, Siemens, and Autodesk and by equipment grants from PhotoNeo, Nvidia, and Intuitive Surgical. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 2146752. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We thank Timothe Kasriel, Chung Min Kim, and Kush Hari for their helpful discussions and feedback.

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," *arXiv*, 2022.

[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*, 2023. arXiv: 2309.16909 [cs.RO].

[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," *arXiv preprint arXiv:2405.08216*, 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, D. Grandi, R. Tomich, M. F. Alam, H. Cheong, and F. Ahmed, "Designqa: A multimodal benchmark for evaluating large language models' understanding of engineering documentation," *arXiv preprint arXiv:2404.07917*, 2024.

[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] 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.

[46] 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*, 2017. arXiv: 1610.07584 [cs.CV].

[47] 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.

[48] 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.

[49] 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.

[50] H. Izadinia, Q. Shan, and S. M. Seitz, *Im2cad*, 2017. arXiv: 1608.05137 [cs.CV].

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] L. Wang *et al.*, *Gensim: Generating robotic simulation tasks via large language models*, 2024. arXiv: 2310.01361 [cs.LG].

[59] 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.

[60] 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," *arXiv preprint arXiv:1803.08495*, 2018.

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

[62] 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.

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

[64] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

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

[66] 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.

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

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

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

[70] 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.

[71] 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.

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

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

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

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

[76] 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.

[77] L. Fu *et al.*, "In-context imitation learning via next-token prediction," *arXiv preprint arXiv:2408.15980*, 2024.

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

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] G. Wang *et al.*, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

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

[86] 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].

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

[88] 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.

[89] A. Kirillov *et al.*, "Segment anything," *arXiv:2304.02643*, 2023.

[90] L. Lipson, Z. Teed, and J. Deng, "Raft-stereo: Multilevel recurrent field transforms for stereo matching," in *2021 International Conference on 3D Vision (3DV)*, IEEE, 2021, pp. 218–227.

[91] 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.

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

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

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