L3GO: LANGUAGE AGENTS WITH CHAIN-OF-3D-THOUGHTS FOR GENERATING UNCONVENTIONAL OBJECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based image generation models such as DALL-E 3 and Stable Diffusion-XL demonstrate remarkable capabilities in generating images with realistic and unique compositions. Yet, these models are not robust in precisely reasoning about physical and spatial configurations of objects, especially when instructed with unconventional, thereby out-of-distribution descriptions, such as *"a chair with five legs"*. In this paper, we propose an language agent with chain-of-3D-thoughts (L3GO), an inference-time approach that can reason about part-based 3D construction of unconventional objects that current data-driven diffusion models struggle with. More concretely, we use large language models as agents to compose a desired object via trial-and-error within the 3D simulation environment. To facilitate our investigation, we develop a new benchmark, Unconventionally Feasible Objects (UFO), as well as SimpleBlenv, a wrapper environment built on top of Blender[1] where language agents can build and compose atomic building blocks via API calls. Human and automatic GPT-4V evaluations show that our approach surpasses the standard GPT-4 and other language agents (e.g., ReAct and Reflexion) for 3D mesh generation on ShapeNet. Moreover, when tested on our UFO benchmark, our approach outperforms other state-of-the-art text-to-2D image and text-to-3D models based on human evaluation.

Prompt: "a chair with five legs"

DALL-E 3 L3GO

Figure 1: We compare one of the state-of-the-art text-to-image models (DALL-E 3) with our LLM-based approach (L3GO). We perform five iterations of DALL-E 3 generation with human feedback but DALL-E 3 does not strictly follow the prompt. L3GO creates a chair with the correct number of legs.

1 INTRODUCTION

AI applications that generate 2D images Betker et al.; Saharia et al. (2022); Podell et al. (2023) and 3D models Jun & Nichol (2023); Lin et al. (2023) from text instructions have opened up significant possibilities for creators. However, these tools lack precise output controls as they often produce unexpected or "*hallucinatory*" results Saharia et al. (2022) not loyal to the input prompts. Additionally, early versions of Stable Diffusion Rombach et al. (2022) had difficulty in combining

[1] https://www.blender.org/

multiple concepts in one image or would mix up different attributes. Previous efforts have improved performance on object-attribute attachment, missing objects, etc. by steering attention layers Feng et al. (2022); Chefer et al. (2023); Rassin et al. (2023), or, by training larger models with detailed captions on a vast scale (StableDiffusion-XL (SDXL) Podell et al. (2023) and DALL-E-3 Betker et al..) However, even the most performant diffusion model, DALL-E 3, still fails to generate objects that require precise 3D spatial understanding like "a chair with five legs" (Figure 1). This difficulty persists even after repeated attempts to adjust DALL-E-3's outputs with human feedback directly, e.g., "The chair you generated has seven legs. Please make a chair with exactly five legs."

(a) Lamps (b) Tables (c) Airplanes

Figure 2: GPT-4 tries to construct three types of objects from ShapeNet by writing Python scripts in Blender. It can successfully create simple items like lamps, but faces challenges with more complex objects such as tables and airplanes.

We posit that the sophisticated text-based reasoning abilities inherent in LLMs can compensate for shortcomings in the 3D spatial comprehension of text-to-2D image and text-to-3D models. We present L3GO, an inference agent capable of iteratively soliciting feedback from LLMs to integrate corrections and enhance the precision of rendering a 3D mesh used as a skeleton to generate 2D image.

We conduct our experiments within Blender —a widely acclaimed 3D modeling software. We create and release an environment called SimpleBlenv, based on Blender, to systematically evaluate text-to-3D mesh generation performance of LLM agents. State of the art LLMs such as GPT-4 Bubeck et al. (2023), despite being trained on only text has decent spatial reasoning capabilities. Figure 2 shows mixed results when GPT-4 is prompted to write a Python script that can run in Blender to create 3D meshes of basic objects solely based on the object name. On the one hand, text-only GPT-4 demonstrates surprising proficiency in creating simple 3D objects like lamps (2a) comprising of three basic shapes. However, as object complexity increases to more than four parts (ex: four legs of a table) or complex objects like airplane (2b, 2c), GPT4's success in perfectly assembling them is limited.

Our L3GO agent bridges these gaps by breaking down the constructing complex objects by employing a more structured, part-by-part approach into: (1) identifying relevant parts specifications and critiquing them, (2) identifying spatial specifications and placement, (3) running the current action to critique the spatial placement and completion. This setup iteratively seeks feedback from SimpleBlenv and the specifications and critiques are generated from LLM. Finally, we render the mesh into an image, and feed it into ControlNet Zhang et al. (2023) with Canny edge detection Canny (1986) to generate a textured and more natural looking image. We conduct human evaluations to compare the performance of LLM-based mesh creation using 13 popular object categories from ShapeNet. L3GO outperforms basic GPT-4, ReAct-B, and Relfexion-B according to both human and auto evaluation. We also show that mesh quality evaluation using GPT-4V OpenAI (2023) yields a metric with high correlation to human judgement. Finally, we introduce Unconventionally Feasible Objects, named UFO with unconventional yet feasible objects. We show that L3GO surpasses current state-of-the-art text-to-2D image and text-to-3D mesh models on UFO. Collectively, our findings indicate the promising role of integrating language agents in diffusion model pipelines, particularly for constructing objects with specific attribute requirements in the future applications of generative AI.

2 Related Work

Spatial Understanding of Language Models Numerous studies have delved into the spatial comprehension capabilities of language models. Janner et al. (2018) explored the spatial reasoning of LSTM Hochreiter & Schmidhuber (1997) in a simulated environment with agent actions and rewards, though their 2D grid world environment is notably simpler than the 3D modeling context considered in our work. Abdou et al. (2021) and Patel & Pavlick (2022) demonstrated that language models develop internal representations for directional information. Additionally, Mirzaee et al. (2021)

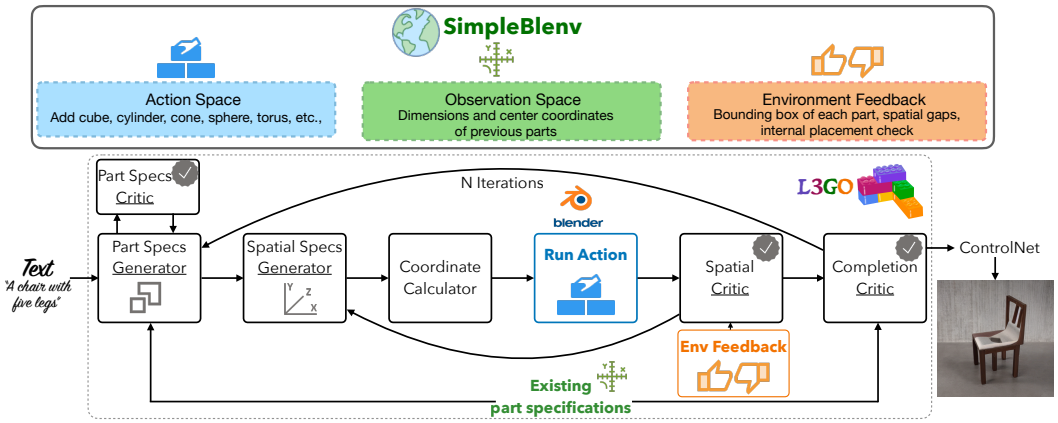


Figure 3: (Top): SimpleBlenv, a wrapper environment on top of Blender, where LLM can construct a 3D mesh by using atomic building blocks. (Bottom): Schematic diagram of L3GO.

introduced a question-answering benchmark for spatial reasoning based on language description. It is reasonable to assert that LLMs exhibit spatial reasoning capabilities and can be effectively prompted to offer feedback on spatial constructions.

Large Language Models as Agents LLM agents Ge et al. (2023); Park et al. (2023); Shen et al. (2023); Gupta & Kembhavi (2023); Wu et al. (2023); **?** represent a new category of artificial intelligence systems built upon large models. These agents are capable of acting, reasoning, and interacting with external environments. Although LLMs has limited executable skills on their own, when integrated with external APIs and knowledge sources, they can tackle a wide range of tasks Schick et al. (2023). An iterative approach has shown to be beneficial to solve natural language processing tasks, as evidenced by ReAct Yao et al. (2022b) and embodied tasks applied to games Wang et al. (2023a); **?**, web navigation Yao et al. (2022a), and robot navigation Huang et al. (2022). Our approach, which utilizes LLMs for creating 3D meshes, contributes to the expanding research in this developing field.

Text to 3D models A growing number of studies are exploring how to adapt pre-trained text-to-2D image diffusion models for text-to-3D generation Poole et al. (2022); Lin et al. (2023); Wang et al. (2023b), suggesting that similar challenges are found in text-to-3D models. Meanwhile, LLM based approaches introduce a new perspective to text-to-3D mesh creation, potentially offering ways to address issues with out-of-distribution samples.

3 L3GO FRAMEWORK

The main challenge with generating entire 3D objects in one go is that it often leads to compounding spatial inaccuracies. We propose decomposing the creation of a 3D mesh into distinct parts and placing each component step by step. We name this approach L3GO, an agent that can collect the feedback and execute the action from a chain of 3D thoughts in a simple 3D environment. This approach transforms a singular attempt at object construction into iterative feedback collection and correction processes, enabling the integration of feedback from the Blender environment. Our framework borrows ideas from previous work of LLM-based reasoning and acting Yao et al. (2022a); Wang et al. (2023a), but has been adopted for 3D mesh construction in a practical 3D modeling software.

3.1 SIMPLEBLENV

We introduce SimpleBlenv, an environment built on top of Blender, where agents can easily submit action commands and receive environmental feedback, such as bounding box information and placement errors. We plan to release the code for the environment.

Action space: In Blender, professional 3D designers have the ability to create complex 3D models, with a nearly limitless range of actions at their disposal. Although nearly every action a user can perform in Blender's UI has a corresponding Python API, we choose to focus on five basic shape primitive APIs for the sake of simplicity. These APIs are: `primitive_cube_add`,

`primitive_cylinder_add`, `primitive_cone_add`, `primitive_uv_sphere_add`, and `primitive_torus_add`. As their names imply, these are used for adding cubes, cylinders, cones, spheres, and toruses, respectively.

These API functions come with a complex set of arguments. To make it easier for LLMs to use these functions, we wrap each function with a wrapper that only requires a few key parameters, such as scale, location, and radius. Consequently, the range of actions available to our L3GO agent includes different settings for size, position, radius, and so on, for each shape-creating command. An illustration of this can be seen in the following example:

```
def create_cube(name, location, scale):
    bpy.ops.mesh.primitive_cube_add(size=1, location=location)
    cube = bpy.context.object
    cube.name = name
    cube.scale = scale
    return cube
```

For a detailed list of all the action wrapper APIs we created, refer to the Appendix. Despite using only five basic shape APIs, agents can create objects in many ways thanks to the flexibility in scaling, positioning, and adjusting the radius, among other controls.

Observations and Environment Feedback We maintain a state space representation as a list of object parts that have been created so far, including their size in terms of x, y, z-axis and location in the global coordinate. Regarding environment feedback, after the agent selects an action, we execute the action in Blender thereby creating a mesh in the virtual space. From this, we can (a) extract information such as the bounding box dimensions of the object parts, and (b) check if the part built by the agent is intersecting with any other parts or if there is an unnecessary gap between the parts (e.g. see Figure 4.) We have built a set of functions to directly gather this information from Blender. This feedback is then relayed to the L3GO agent as text messages before it takes its next action.

3.2 L3GO: LLM-BASED 3D GENERATION OF OBJECTS

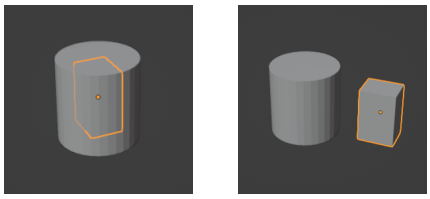
In this section, we introduce L3GO, an LLM agent specifically designed for 3D mesh creation from text. L3GO is comprised of six components, each powered by a language model that either functions as a *generator* or a *critic*. The schematic diagram in Figure 3 is shown for a visual overview.

Part Specifications Generator: L3GO first prompts the LLM to identify the most pivotal part of the object. This pivotal part makes it easier to attach subsequent components. For instance, starting with the seat of a chair is practical because it is straightforward to add legs and a backrest to it, simplifying the coordinate calculations for the other parts. After naming the part, the agent uses a size generator to determine its reasonable dimensions in terms of width, depth, and height, corresponding to the x, y, and z axes.

Part Specifications Critic: Once a part name is proposed, it undergoes a review by the Part Specifications Critic. This step is crucial to avoid ambiguity, which can confuse the agent later. For example, if "leg" is proposed while creating a chair, the agent cannot know its exact placement without a spatial descriptor like "front right leg". The Part Specifications Critic's role is to identify and correct such vague descriptions, allowing the Part Specifications Generator to revise its suggestion accordingly. The process moves forward only after the Part Specifications Critic's approval.

Spatial Specifications Generator: After establishing the part name and size, the model considers the spatial requirements of the part, given what has already been constructed. (For the first part, we simply position it at the center.) The agent begins by selecting the most appropriate base part to attach the new component to, then determine the spatial relationship between them. For instance, if constructing an airplane with the fuselage as the base and the left wing as the new part, a typical spatial requirement would be to attach the wing to the middle of the fuselage's left side.

Coordinate Calculator: Based on the spatial requirements and the base part's position, this component calculates the new part's center coordinate. Accuracy here is crucial, as even minor



(a) Fully inside (b) Spatial gap

Figure 4: Two types of error feedback we provide in SimpleBlenv: (a) The newly added cuboid (in orange) is completely inside the base cylinder. (b) There is unnecessary spatial gap between the newly added cuboid and the base cylinder.

misalignments can impact the overall correctness. To ensure precision, the agent is given access to a python execution environment: while using the LLM only to generate Python code for calculating the position of the new part. This approach is similar to the one described in Gao et al. (2023). To increase reliability, the process is repeated three times and determine the x, y, and z coordinates based on a majority vote from these repetitions (with ties broken arbitrarily)

Run action: After determining the size and spatial position, the agent asks an LLM to decide on the part's shape, choosing from cubes, cylinders, cones, spheres, and toruses. Then, the agent writes a valid Python script for Blender, specifying the size, position, and shape type. Finally, the agent runs a command to generate the part's mesh in Blender. This code is executed in Blender in a headless mode, and the environment provides important feedback, such as the bounding boxes of each generated part, which is used in the next module.

Spatial Critic: After running the Blender code, two final spatial correctness checks are conducted: a *continuity check*, which tells the agent if the newly created part is disconnected from the existing parts, and a *total overlap check* with existing parts, which tells the agent if a newly created part is entirely contained within an existing part. If either issue arises, the process returns to the spatial requirement generation stage and the agent adjusts accordingly. See examples of spatial errors in Figure 4.

Completion Critic: The final step is to determine whether the construction of a 3D mesh for an object is completed. To do this, this critic is provided with the name of the object being built and the list of its parts that have already been constructed to an LLM to make a binary decision of completion. If the critic predicts that it is incomplete, we start the next iteration with the Part Specifications Generator. If the task is completed, we proceed to generating a more natural-looking image using ControlNet.

ControlNet for 3D Meshes $\rightarrow$ 2D Images After the L3GO agent finishes the creation of a 3D mesh, we render the object into gray-scaled image. We then feed this image to ControlNet with Canny edge detection to produce a more realistic looking image.

Note that as L3GO is text-based, it does not use visual information. Therefore, all communication must be text-based, including defining spatial orientations like which direction is the front or back when asking the model to create an object. We set these spatial assumptions in the prompts in advance to guide the construction process. Unless specified otherwise, we use GPT-4 as our base LLM in L3GO in our experiments.

4 Experiments on ShapeNet

In this section, we detail the baseline methods (§4.1) to compare text-to-2D image and text-to-3D mesh generation to compare with our L3GO. We demonstrate the effectiveness of LLM-based methods in the generation of simple 3D objects, specifically focused on 13 well-known categories sourced from ShapeNet. For automatic evaluation, we use GPT-4V as evaluator for recognizing 3D meshes (§4.2) of simple objects. We also show that human assessments and GPT-4V's automated evaluations (§4.2) are well correlated.

	Human	L3GO	ReAct-B	Reflexion-B	GPT-4
GPT-4V	0.877	**0.6**	0.423	0.4	0.346
Human	0.894	**0.584**	0.385	0.403	0.445

Table 1: Mean accuracy of different LLM-based agents on ShapeNet-13, evaluated by GPT-4V (top row) and humans (bottom row); each cell is an average over 130 trials. 'Human' in the column names refers to the original ShapeNet meshes, designed by humans, which can be considered as the upper bound. We see that L3GO outperforms other GPT-4-based agents (e.g. ReAct-B, Reflexion-B, and unmodified GPT-4).

4.1 BASELINES

Given the absence of pre-existing LLM agents designed to work in Blender, we chose a range of algorithms that serve as baseline references. Originally intended for natural language processing tasks, we have adapted these baselines to function within the Blender environment, ensuring they align with our experimental framework.

ReAct-B ReAct Yao et al. (2022b) is a framework designed for implementing a language model based agent. In this framework, the agent outputs its thought process before taking any action. The observations gathered from the external environment following this action inform the subsequent steps. We implement ReAct in the SimpleBlenv setting, utilizing the environment feedback, observations, and action space outlined in Section 3. To differentiate it from the text version of ReAct, we refer to our implementation as ReAct-Blender, or ReAct-B for short.

Reflexion-B Reflexion Shinn et al. (2023) builds upon the ReAct framework by adding an extra step of reflection. In ReAct, the agent outlines its reasoning, takes an action, and then receives feedback from the environment. Reflexion goes a step further – after receiving environment feedback, the agent engages in reflection on the action taken and its results, to inform its next move. In our setup, at the end of every iteration, we consider the current object part, previously built parts, and the current environment feedback. We then ask the agent to reflect on the size and placement of the object part it has just built. After the agent shares its thoughts, we prompt it to decide whether to redo the current part or move on. If the agent chooses to redo, its reflective insights are used in the next step.

GPT-4 For a less structured approach, we use GPT-4 to generate the entire Python code needed to create an object in one single attempt. The prompt we used for GPT-4 was adapted from a Github repository gd3kr (2023), which, to our knowledge, was the first to present the open-source project that uses GPT-4 to control Blender. For the full prompt, refer to the Appendix.

Dataset: ShapeNet-13 To assess the basic mesh generation ability, we use 13 categories of ShapeNet: ['airplane', 'bench', 'cabinet', 'car', 'chair', 'display', 'lamp', 'loudspeaker', 'rifle', 'sofa', 'table', 'telephone', 'watercraft'], as introduced by Choy et al. (2016).

4.2 AUTOMATIC EVALUATION VIA GPT-4V

To streamline our evaluation process, we propose using GPT-4V to assess the performance of mesh construction. For each object category, we generate 10 meshes from GPT-4, ReAct-B, Reflexion-B, and L3GO. After the agent finishes mesh construction, we render the object from 10 different views by rotating the camera around the object at the same height. This results in 10 images per mesh. we then feed 10 images to GPT-4V all at once, and use the following prompt: *''What object do you see in these images? Answer with a single object name. Your answer must be one of the following options: [airplane, bench, cabinet, car, chair, display, lamp, loudspeaker, rifle, sofa, table, telephone, watercraft]"*.

Table 1 presents the average accuracy across different object categories. It is evident that structured methods, including ReAct-B, Reflexion-B, and L3GO, surpass GPT-4 in performance. Notably, among these structured approaches, L3GO proves to be the most effective. Delving into the results for each object, as detailed in Figure 5, it becomes clear that L3GO particularly excels in constructing complex objects like airplanes and rifles.

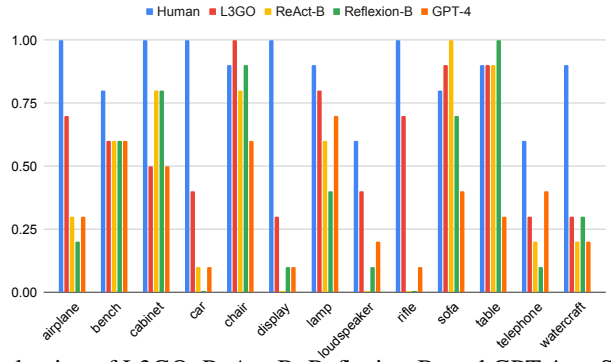


Figure 5: GPT-4V evaluation of L3GO, ReAct-B, Reflexion-B, and GPT-4 on ShapeNet-13. 'Human' refers to original ShapeNet meshes that were designed by humans. For complex objects such as airplanes and rifles, L3GO performs better than others.

	GPT-4V	InstructBLIP	BLIP-2	Human
Human	**0.512**$_{(0.028)}$	0.344$_{(0.016)}$	0.341$_{(0.012)}$	0.569$_{(0.020)}$

Table 2: The Cohen's Kappa correlation between evaluations based on models and human judgement. We report the average and standard deviation calculated from three independent human evaluators.

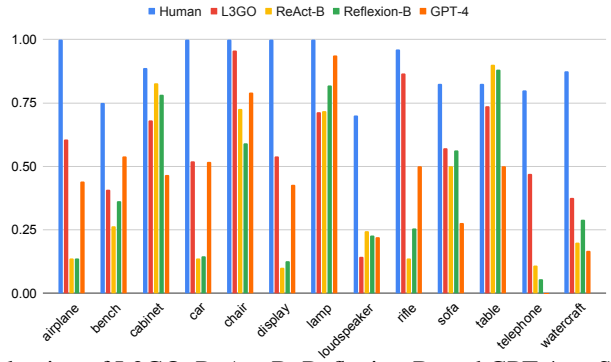


Figure 6: Human evaluation of L3GO, ReAct-B, Reflexion-B, and GPT-4 on ShapeNet-13. 'Human' refers to the original human-designed ShapeNet meshes. We observed a pattern similar to that in GPT-4V's evaluation.

Correlation with human evaluations We recruit human participants to assess whether the evaluation of GPT-4V aligns with human judgment. For each mesh, we use 10 images from 10 different angles (the same images as above for GPT-4V evaluation), and ask a human participant to classify these images into one the 13 object categories. We collect 130 human responses by showing meshes generated by GPT-4, and L3GO, as well as the original ShapeNet meshes, totaling in 390 human responses. The category-by-category results are shown in Figure 6. We can see an overall pattern, where the original ShapeNet has high accuracy, L3GO outperforms GPT-4, except for a few cases like "lamp", "bench", and "loudspeaker".

We also gather 390 responses from GPT-4V using the same set of images. To benchmark GPT-4V against other top vision-language models, we also obtain 390 responses each from BLIP-2 and InstructBLIP. Regarding human-to-human correlation evaluation, four participants were asked to classify all 390 images. However, we exclude the data from the participant whose responses most often differed from the other three. We then calculate the Cohen's Kappa score to measure the correlation between these models and three human evaluators, averaging these correlations, as shown in Table 4. Our findings indicate that GPT-4V's responses align most closely with human judgments, though it is important to note that even among humans, agreement was not perfect.

5 EXPERIMENTS ON UFO: CONSTRUCTING UNCONVENTIONALLY FEASIBLE OBJECTS

Our previous experiments show that L3GO can accurately construct a simple object from ShapeNet around 60% of the time. However, modern diffusion based models can nearly always generate an image of a given ShapeNet object. This is in part because there are many possible valid instantiations of, e.g., "car" or "bench". So: is there any potential practical advantage to using a method like L3GO?

To illustrate the potential advantages of LLMs in spatial construction, we introduce a benchmark that requires more precise spatial understanding. Inspired by DrawBench Saharia et al. (2022) and PartiPrompts Yu et al. (2022) which are collections of prompts that systematically evaluate text-to-image models, we introduce UFO: a set of 50 difficult prompts that 1) require precise spatial understanding to construct; and 2) are unusual, in the sense that they are less likely to occur during text-only pre-training, e.g., "a chair with one armrest".

The prompts in UFO span 9 object categories, and each prompt is a combination of a common object with varied characteristics such as "a chair with five legs", "a mug with two handles" and so on. The full prompt list is shown in the Appendix. We focus on everyday 3D objects to help us isolate the model's performance in accurately interpreting prompts from its inherent ability to create unconventional objects. By using simple-to-assemble items such as sofas, chairs, lamps, and tables, we can better discern whether any shortcomings are due to the model's prompt following from its object creation skills.

Baselines We compare our LLM-based approach with latest text-to-2D and text-to-3D methods such as DALL-E 3 Betker et al., Stable Diffusion XL (SDXL) Podell et al. (2023), and Shap-E Jun & Nichol (2023). DALL-E 3 uses descriptive synthetic captions to improve prompt following of DALL-E 2 Ramesh et al. (2022), the previous version of OpenAI's text-to-image diffusion model. Stable Diffusion XL is an open-sourced text-to-image diffusion model. Shap-E Jun & Nichol (2023) is a text-to-3D model that generates the parameters of implicit function for 3D models which then can be rendered. Since DALL-E 3 automatically re-writes the prompt for safety reasons and adds more detail, (and it is not possible to disable this feature at the moment) we add "I NEED to test how the tool works with extremely simple prompts. DO NOT add any detail, just use it AS-IS:" to our prompt as recommended by OpenAI [2].

Experiment procedures We again utilize the judgements of human participants to evaluate the output of our models on UFO. For each given prompt, we generate 10 random objects from one model and another 10 from a different model. A participant is then asked to judge which set of images better matches the provided text caption. If they believe that neither set accurately represents the caption, they can choose "no preference." For each experiment, we recruit 10 human evaluators. Additionally, we include 4 attention check questions on top of the 50 total questions. Any evaluator who does not correctly answer all the attention check questions is excluded from our analysis. For models that create 3D objects, we render 10 images from various angles by rotating the camera at a constant height. These 10 images are then compiled into a rotating GIF.

Results The results are shown in Figure 7. L3GO outperforms the other LLM agents (e.g. ReAct, Reflexion) and the state-of-the-art text-to-image models (DALL-E-3 and SDXL) and text-to-3D model (Shap-E) in terms of human preference. Example generated images are shown in Figure 8. DALL-E-3, SDXL and Shap-E produce images that do not perfectly follow the specific prompt instructions. While SDXL is able to generate a desk with three legs, an additional chair that is not asked in the prompt is generated. DALL-E-3 seems to completely ignore the specific requirements of prompts. In contrast, while their designs are not perfect, language model-based agents are capable of constructing chairs with the right number of legs. These results suggest that structured reasoning may serve as a viable strategy to mitigate the challenges posed by insufficient training data.

[2] https://platform.openai.com/docs/guides/images

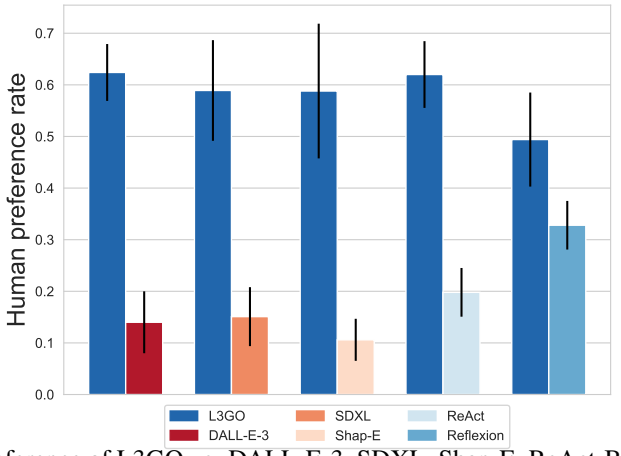


Figure 7: Human preference of L3GO vs. DALL-E-3, SDXL, Shap-E, ReAct-B, and Reflexion-B on UFO.

Figure 8: Example generated images based on UFO. The LLM-based approaches (ReAct-B, Reflexion-B, and L3GO) successfully create the desired objects, while some of the most advanced text-to-image and text-to-3D models (DALL-E 3, Stable Diffusion XL, and Shap-E) still struggle to follow the prompt perfectly.

6 Conclusion

We introduced L3GO, a language agent designed to generate 3D objects from text instructions through an API we developed for Blender, a 3D modeling software. Our evaluation using 13 largest object categories from ShapeNet shows that L3GO's superior capabilities in comparison to other models such as GPT-4, ReAct, and Relfexion. Additionally, we devised UFO, a set of challenges aimed at testing the ability of generative AI models in creating common objects with unconventional characteristics. The performance of L3GO marks a significant advancement in the application range of language models. For instance, diffusion models could be further improved with unconventional data generated by structured prompting. Moreover, analyzing how language models process spatial information with internal model representations may yield valuable insights into understanding and improving their 3D modeling abilities.

References

Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. Can Language Models Encode Perceptual Structure Without Grounding? A Case Study in Color. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pp. 109–132, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.conll-1.9.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving Image Generation with Better Captions. *https://openai.com/dall-e-3*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4, April 2023.

John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986. ISSN 1939-3539. doi: 10.1109/TPAMI.1986.4767851.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-Excite: Attention-Based Semantic Guidance for Text-to-Image Diffusion Models. *ACM Transactions on Graphics*, 42(4):148:1–148:10, July 2023. ISSN 0730-0301. doi: 10.1145/3592116.

Chris Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 9912, pp. 644, October 2016. ISBN 978-3-319-46483-1. doi: 10.1007/978-3-319-46484-8_38.

Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-Free Structured Diffusion Guidance for Compositional Text-to-Image Synthesis. In *The Eleventh International Conference on Learning Representations*, September 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10764–10799. PMLR, July 2023.

gd3kr. BlenderGPT. https://github.com/gd3kr/BlenderGPT, 2023.

Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. OpenAGI: When LLM Meets Domain Experts. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2023.

Tanmay Gupta and Aniruddha Kembhavi. Visual Programming: Compositional Visual Reasoning Without Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9, 1997.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Conference on Robot Learning*, 2022. doi: 10.48550/ARXIV.2207.05608.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. Representation Learning for Grounded Spatial Reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61, 2018. doi: 10.1162/tacl_a_00004.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of Experts, January 2024.

Heewoo Jun and Alex Nichol. Shap-E: Generating Conditional 3D Implicit Functions, May 2023.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-Resolution Text-to-3D Content Creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 300–309, 2023.

Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. SPARTQA: A Textual Question Answering Benchmark for Spatial Reasoning. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4582–4598, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.364.

OpenAI. GPT-4V(ision) technical work and authors, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, pp. 1–22, New York, NY, USA, October 2023. Association for Computing Machinery. ISBN 9798400701320. doi: 10.1145/3586183.3606763.

Roma Patel and Ellie Pavlick. Mapping Language Models to Grounded Conceptual Spaces. In *International Conference on Learning Representations*, January 2022.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis, July 2023.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations*, September 2022.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents, April 2022.

Royi Rassin, Eran Hirsch, Daniel Glickman, Shauli Ravfogel, Yoav Goldberg, and Gal Chechik. Linguistic Binding in Diffusion Models: Enhancing Attribute Correspondence through Attention Map Alignment. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems*, May 2022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, November 2023a.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023b.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. 2023. doi: 10.48550/ARXIV.2303.04671.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, December 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*, September 2022b.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *Transactions on Machine Learning Research*, August 2022. ISSN 2835-8856.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

A Appendix

B Effect of background and texture on evaluation for UFO

We look into how the background and texture differences in images created by text-to-image models and LLM-based methods might affect human evaluations. To test this, we change prompt styles with these text-to-image models. For DALL-E-3, we use "[object name] Make sure the background is black, and the object is a gray-colored 3D shape." For Stable Diffusion XL, we use "[object name], black background, gray-colored 3D shape." Additionally, we alter the guidance scale for Stable Diffusion XL, which determines how closely the diffusion model follows the text prompts. In both scenarios, we observe that L3GO outperforms text-to-image models in terms of human preference, as illustrated in Figure 9. We also conducted an initial test to determine if GPT-4V could serve as an evaluator for UFO. However, we observed that in over 20% of the cases, GPT-4V refuses to provide an answer, which might be related to the characteristics of the images generated. We refer to Table 3 in Appendix for more details.

C Algorithm

The pseudo algorithm of L3GO is given in Figure 10.

D GPT-4V as an evaluator for UFO

We also conducted an initial test where we used GPT-4V to evaluate generated images using prompts from UFO. This experiment involved showing GPT-4V two sets of 10 images from two models, similar to our human study, and asking GPT-4V which set of images more accurately represent the corresponding text caption. We ask it choose either the 'top row', 'bottom row', or 'no preference', along with providing a reason for its choice. The images were generated using L3GO and DALL-E-3.

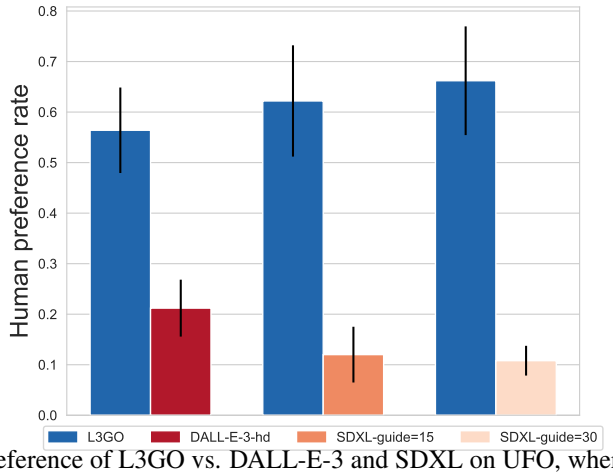


Figure 9: Human preference of L3GO vs. DALL-E-3 and SDXL on UFO, where we attempt to make the background of generated images to be simple gray shading. We also vary guidance scales for SDXL to see if better prompt following improves the performance, denoted as 'guide=15, 30'. For DALL-E-3, we use 'quality=hd' option for enhanced detail.

Algorithm 1: 3D-LCoT: Loopy-Chain-of-3D-Thought agent

```
Input: object name, max iter
Output: 3D mesh object
prev parts = EmptyList; currnt iter = 0
while current iter < max iter do
    part name := PartNameGen(object name, prev parts)
    if not isValidName(part name) then
        redo PartNameGen
    part size := PartSizeGen(part name, object name, prev parts)
    spatial req := SpatialRequirementGen(part name, part size, prev
     parts, object name)
    part coord := CoordinateCalc(part name, part size, prev parts,
     spatial req, object name)
    scene := AddMesh(part name, part size, part coord)
    env feedback := BlenderEnv(scene)
    if not isValidSpatial(env feedback) then
        redo CoordinateCalc
    prev parts := append(part name)
    if isComplete(object name, prev parts) then
        break
    currnt iter += 1
```

Figure 10: Pseudo algorithm of L3GO.

We carried out this experiment twice. In both cases, for more than 20% of the prompts (13 and 18 out of 50), the response we received was "I cannot assist with this request". Due to this, we decided to rely on human evaluators instead of GPT-4V for our main experiments.

	L3GO	DALL-E-3	No pref	Refuse	No images
1st trial	12	8	9	18	3
2nd trial	14	12	11	13	0

Table 3: GPT-4V's evaluation of UFO, where GPT-4V compares the image generation by L3GO and DALL-E-3. "Refuse" indicates instances where GPT-4V declined to respond, for example by saying "I cannot assist with this request." "No images" refers to cases where GPT-4V incorrectly perceived the images as being invisible.

E DETAILS FOR HUMAN EVALUATION

We recruit participants through Prolific.com, an online platform used by researchers to conduct behavioral studies and gather participants. The requirement for participants was English fluency, as the study's materials were all in English. The study was made accessible to all qualified participants on the platform. We ensure $15 hourly wage.

At the beginning of the experiment, we give participants an online form detailing the experiment's aims, a summary, the methodology, confidentiality assurances, and the voluntary nature of their involvement. The participants are assured of the confidentiality of their responses and their right to withdraw at any time. Participants are asked to acknowledge their understanding of the provided information and their consent to participate by clicking the "start" button. The experiment commences only after this button was clicked.

Despite efforts to minimize bias by keeping participant criteria to a minimum, there remains an inherent bias towards English-speaking individuals with access to computers and the internet. We note that this bias also tends to reflect characteristics of the WEIRD (Western, Educated, Industrial, Rich, Democracies) demographic.

F ADDITIONAL DETAILS FOR EXPERIMENTS

Wrapper APIs for action commands in SimpleBlenv are shown in Figure 11.

Example environment feedback from SimpleBlenv is shown in Figure 12.

Full prompt for the baseline GPT-4 is shown in Figure 13. This prompt is taken from BlenderGPT[3], which is the first open-source code that uses GPT-4 inside Blender to our knowledge.

The list of prompts in UFO a chair with two legs, a chair with three legs, a chair with five legs, a chair with six legs, a chair with seven legs, a chair with one armrest, a chair with three armrests, a chair with two backrests, a chair with three backrests, a stool with two legs, a stool with three legs, a stool with five legs, a stool with six legs, a stool with seven legs, a stool with one armrest, a stool with three armrests, a stool with two backrests, a stool with three backrests, a desk with two legs, a desk with three legs, a desk with five legs, a desk with six legs, a desk with seven legs, a table with two legs, a table with three legs, a table with five legs, a table with six legs, a table with seven legs, a pair of eyeglasses with one round lens and one square lens, a pair of eyeglasses with one round lens and one triangle lens, a pair of eyeglasses with one square lens and one triangle lens, a pair of eyeglasses with three lenses, a pair of eyeglasses with four lenses, a pair of eyeglasses with five lenses, a sofa with one leg, a sofa with two legs, a sofa with three legs, a sofa with five legs, a sofa with legs that are longer than its backrest, a sofa with armrests that are longer than its backrest, a lamp with two legs, a lamp with four legs, a lamp with five legs, a bottle whose lid is twice as wide as its mouth, a bottle with a lid that is three times wider than its mouth, a bottle with a lid that is four times wider than its mouth, a mug with two handles, a mug with three handles, a mug with four handles, a mug with five handles

F.1 ADDITIONAL GENERATED EXAMPLES

are shown in Figure 14 and 15.

The code is uploaded in the following url.[4]

G ABLATION STUDIES

We ablate three 3 system design choices to see which component most impacts the overall performance for 3D mesh generation. We use our automatic evaluation via GPT-4V to compare the performance in

[3] https://github.com/gd3kr/BlenderGPT

[4] https://u.pcloud.link/publink/show?code=kZ4e550Z2VnyJfJO8VHyivUkfVdKhb5B8SEk

```python
def create_cube(name, location, scale):
    bpy.ops.mesh.primitive_cube_add(size=1, location=location)
    cube = bpy.context.object
    cube.name = name
    cube.scale = scale
    return cube


def create_cylinder(name, radius, z_axis_height, location):
    bpy.ops.mesh.primitive_cylinder_add(
        radius=radius, depth=z_axis_height, location=location
    )
    cylinder = bpy.context.object
    cylinder.name = name
    return cylinder


def create_cone(name, radius, z_axis_height, location):
    bpy.ops.mesh.primitive_cone_add(radius1=radius, depth=z_axis_height, location=location)
    cone = bpy.context.object
    cone.name = name
    return cone


def create_uv_sphere(name, location, scale):
    """
    segments: int
    rings: int
    """
    bpy.ops.mesh.primitive_uv_sphere_add(radius=0.5, location=location)
    sphere = bpy.context.object
    sphere.name = name
    sphere.scale = scale
    return sphere


def create_torus(name, major_radius, minor_radius, location, scale):
    bpy.ops.mesh.primitive_torus_add(
        major_radius=major_radius, minor_radius=minor_radius, location=location
    )
    torus = bpy.context.object
    torus.name = name
    torus.scale = scale
    return torus


def rotate_object(object_name, rotation, orient_axis):
    """
    Rotate object by rotation.

    object_name: string, name of the object
    rotation: float, rotation in degrees
    orient_axis: string, axis to rotate around. Should be one of 'X', 'Y', 'Z'.
    Example:
    rotate_object("Cube", 90, 'X')
    """
    ob = bpy.context.scene.objects[object_name]  # Get the object
    bpy.ops.object.select_all(action="DESELECT")  # Deselect all objects
    bpy.context.view_layer.objects.active = ob  # Make the cube the active object
    ob.select_set(True)  # Select the cube
    bpy.ops.object.mode_set(mode="EDIT")  # Enter Edit Mode
    bpy.ops.transform.rotate(value=math.radians(rotation), orient_axis=orient_axis)
    bpy.ops.object.mode_set(mode="OBJECT")  # Exit Edit Mode
```

Figure 11: Full list of action wrapper APIs for SimpleBlenv.

```
{"fuselage": {"center_coordinate": {"x": 0.0, "y": 0.0, "z": 0.0}, "bbox_corners": {"bottom-left-back":
[-2.5, -22.5, -2.5], "top-right-front": [2.5, 22.5, 2.5]}, "dimensions": {"width": 5.0, "depth": 45.0,
"height": 5.0}},
"left wing": {"center_coordinate": {"x": -10.0, "y": 0.0, "z": 0.0}, "bbox_corners": {"bottom-left-back":
[-17.5, -0.25, -0.75], "top-right-front": [-2.5, 0.25, 0.75]}, "dimensions": {"width": 15.0, "depth": 0.5,
"height": 1.5}},

"front landing gear": {"center_coordinate": {"x": 0.0, "y": 22.0, "z": -4.5}, "bbox_corners": {"bottom-left-
back": [-0.5, 21.5, -6.5], "top-right-front": [0.5, 22.5, -2.5]}, "dimensions": {"width": 1.0, "depth": 1.0,
"height": 4.0}},

"spatial_relations": "fuselage and front landing gear are intersecting (or touching each other) OR "There is
a gap between fuselage and front leading gear, even though they should be attached to each other." OR "front
landing gear is fully inside fuselage and not visible to us.")
```

Figure 12: An example environment feedback from SimpleBlenv.

3 ablations: 1) without spatial critic, 2) without program-based coordinate calculator, and 3) choice of LLMs. For each setting, we generate 10 objects per category, and render 10 images from different angles. For 1) and 2), the mean accuracy across the 13 categories of ShapeNet (evaluated by GPT-4V) are 0.515 and 0.585, respectively. In comparison, L3GO achieves a higher score of 0.6. While the average scores for L3GO and without coordinate calculator achieve similar scores, the latter scores 0 in both the cabinet and car categories. In contrast, when L3GO employs program-based coordinate calculation, it achieves scores of 0.5 and 0.4 for these categories, respectively.

```
System prompt: """You are an assistant made for the purposes of helping the user with Blender, the 3D
software.
    - Respond with your answers in markdown (```).
    - Preferably import entire modules instead of bits.
    - Do not perform destructive operations on the meshes.
    - Do not use cap_ends. Do not do more than what is asked (setting up render settings, adding cameras,
etc)
    - Do not respond with anything that is not Python code.

    Example:

    user: create 10 cubes in random locations from -10 to 10
    assistant:
    ```
    import bpy
    import random
    bpy.ops.mesh.primitive_cube_add()

    #how many cubes you want to add
    count = 10

    for c in range(0,count):
        x = random.randint(-10,10)
        y = random.randint(-10,10)
        z = random.randint(-10,10)
        bpy.ops.mesh.primitive_cube_add(location=(x,y,z))
    ```"""
 User prompt: "Can you please write Blender code for me that accomplishes the following task: Build a 3D
model of " + object_name + "? \n. Do not respond with anything that is not Python code. Do not provide
explanations"
```

Figure 13: The full prompt for the baseline GPT-4 used in our experiments.

Figure 14: Additional generated examples on UFO.

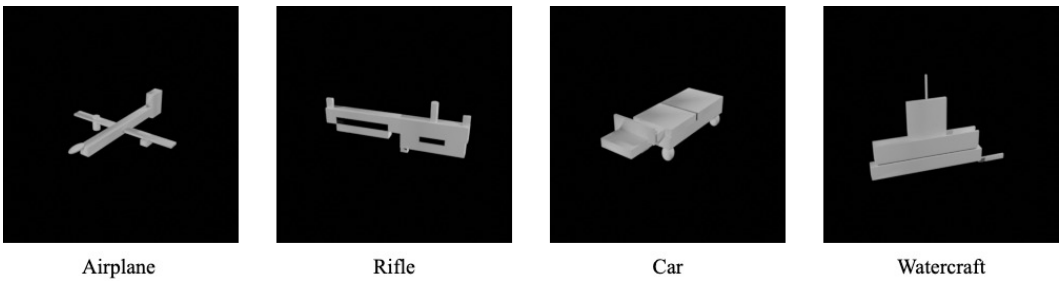


Figure 15: Additional generated examples of L3GO on ShapeNet.

Open-sourced LLMs We explore the use of open-source LLMs in place of GPT-4. For this purpose, we use Mixtral-8x7B, a sparse mixture-of-experts model Jiang et al. (2024), that is known to either match or surpass the performance of Llama-2 70B and GPT-3.5 in most benchmark tests. We carry out experiments using ReAct-B and ShapeNet-13, with GPT-4V serving as our evaluation tool. While the accuracy for the ReAct-B(Mixtral-8x7B) for most shape categories ranged between 0 and 0.1, the categories of sofas, lamps, and tables achieved higher scores of 0.3, 0.3, and 0.7, respectively. This is likely due to their simpler shapes and easier recognizability. The average accuracy score is 0.138. This result is significantly lower than the 0.423 accuracy achieved by ReAct-B(GPT-4), as indicated in Table 1. This indicates that the task of constructing mesh objects, which demands a precise understanding of 3D space, still needs the reasoning abilities found at the level of GPT-4.

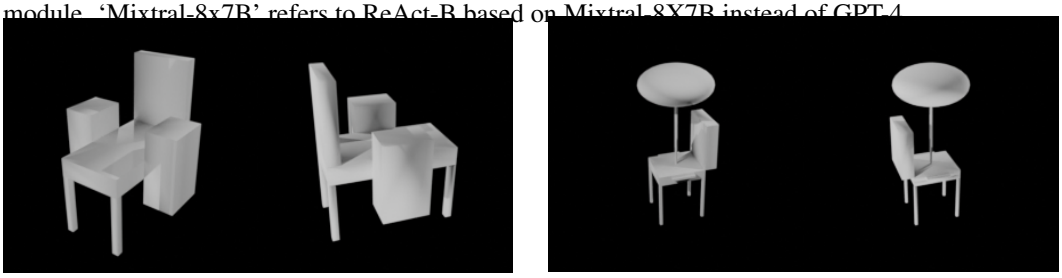
H Applications and Future Work

H.1 Creative Combinations of Object Parts

In UFO, we explored how our approach could adapt to unique variations in common object characteristics, such as creating a chair with an atypical number of legs. While legs are a standard feature of

Mixtral-8x7B	w/o spatial critic	w/o program-based	L3GO
0.138	0.515	0.585	**0.6**

Table 4: Ablation studies. We evaluate the performance based on ShapeNet's 13 categories using GPT-4V as an evaluator; each cell is an average over 130 trials. 'w/o spatial critic/program-based' refers to L3GO based on GPT-4 without spatial critic and without program-based coordinate calculation module. 'Mixtral-8x7B' refers to ReAct-B based on Mixtral-8X7B instead of GPT-4.



Chair with two wings

Chair with an umbrella attached

Figure 16: Because Large Language Models can interpret creative prompts, L3GO can create unique items such as a chair with wings or a chair that has an umbrella attached.

chairs, our focus was on experimenting with unconventional leg counts. In this section, we extended this question to see if L3GO can handle even more creative combinations, like integrating components not typically associated with the object, for instance, a chair with wings.

Displayed in Figure 16 are instances of these creative prompts and their resulting models. Even though the forms for "wings" and "umbrella" are basic, such as rectangles and elongated spheres with thin cylinders, L3GO adeptly figures out that "wings" should attach to the sides of the chair, and an "umbrella" should be positioned on top. (See Appendix for more examples.) Our results demonstrate that L3GO can successfully interpret and construct these unusual designs.

H.2 Language interface for complex softwares

Most 3D modeling software programs have a steep learning curve. Creating sophisticated 3D models often involves a sequence of operations using complex interfaces, which can be challenging for non-experts to navigate. Our method can be used as a language interface for these complex software programs, making them more accessible to beginners. As illustrated in Figure 17, a user can input a prompt, and L3GO will generate an initial draft of the object. While our approach is still in its early stages and needs further development for practical use, we hope our research inspires future advancements.

I Limitations

The quality of 3D mesh objects generated using LLM-based methods lag behind those produced by diffusion-based text-to-3D methods, and far from being on par with human standards. Moreover, creating simple objects takes roughly 3 to 5 minutes, while more complex ones can take 10 to 20 minutes, depending on the number of retry operations needed after the feedback. We believe that the future work should explore more efficient LLM-based approaches to create 3D mesh as this is a promising direction to better control the eventual 3D object generation.

Impact Statement Our research indicates the vast potential for integrating language models with 3D modeling, potentially revolutionizing design processes and the creation of digital environments. This convergence aims at making generative AI tools more intuitive and capable of supporting creative endeavors. With L3GO, we begin to tap into the untapped possibilities in this domain, setting the stage for extensive future exploration and development.

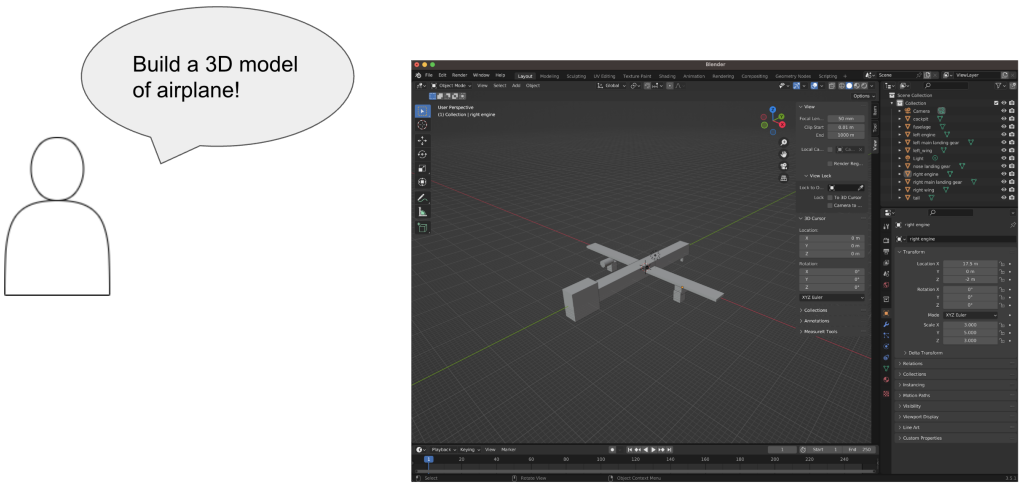


Figure 17: The Blender user interface can be daunting for newcomers. An LLM agent can be helpful by creating a draft of an object to assist beginners.

The positive societal impacts of our work could be substantial, particularly in design, engineering, and the arts, by enabling the visualization and prototyping of ideas that were previously difficult or impossible to achieve. Furthermore, our approach could enhance educational tools, making complex concepts more accessible through interactive and visually intuitive representations. However, we must also consider the ethical implications of advancing image generation technology, such as the potential for creating misleading or harmful content. It underscores the necessity for ongoing research into mechanisms that ensure these powerful tools are used responsibly and ethically. We advocate for a balanced approach that emphasizes innovation alongside responsibility and ethical considerations.