Navigate Complex Physical Worlds via Geometrically Constrained LLM

Yongqiang Huang[1], Wentao Ye[2], Liyao Li[2], Junbo Zhao[2,*]
[1]College of Energy Engineering, Zhejiang University
[2]College of Computer Science and Technology, Zhejiang University
Emails: {hyq.cee, yewt01, liliyao, j.zhao}@zju.edu.cn
*** Corresponding author:** j.zhao@zju.edu.cn

Abstract

This study investigates the potential of Large Language Models (LLMs) for reconstructing and constructing the physical world solely based on textual knowledge. It explores the impact of model performance on spatial understanding abilities. To enhance the comprehension of geometric and spatial relationships in the complex physical world, the study introduces a set of geometric conventions and develops a workflow based on multi-layer graphs and multi-agent system frameworks. It examines how LLMs achieve multi-step and multi-objective geometric inference in a spatial environment using multi-layer graphs under unified geometric conventions. Additionally, the study employs a genetic algorithm, inspired by large-scale model knowledge, to solve geometric constraint problems. In summary, this work innovatively explores the feasibility of using text-based LLMs as physical world builders and designs a workflow to enhance their capabilities.

1 Introduction

LLMs acquire extensive world knowledge embedded in textual data through pre-training. This raises an intriguing question: can LLMs reconstruct and simulate the physical world using this textual knowledge? The physical world, characterized by complex geometric and physical constraints, can be abstracted into fundamental geometric shapes. Utilizing a custom-designed engine, we simplify the 3D world's geometric content into basic cube combinations. This work pioneers the exploration of text-only LLMs as potential builders of the physical world, leveraging their pre-trained knowledge to understand and generate 3D spatial representations purely from textual descriptions.

Some preliminary work on world-building has explored constructing 3D worlds at the image level. Techniques like 3D-VAE-GAN (Wu et al., 2016) and Pix2Vox (Xie et al., 2019) combine Variational Autoencoders (VAEs) (Kingma and Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) to generate high-quality 3D models with precise shape and pose control. AtlasNet (Groueix et al., 2018) approximates 3D surfaces by learning a set of 2D textures, effectively handling irregular topologies. Despite their impressive quality, these models struggle with simulating complex physical interactions and maintaining spatial consistency due to intricate and dynamic geometric constraints(Li et al., 2024).

Some methods rely on high-precision geometric libraries or external knowledge bases for human-level prior knowledge. For instance, Sun et al. (2023) and Zhou et al. (2024) use LLMs to generate 3D scene images by calling Blender APIs based on user requirements. Wu et al. (2024) proposes combining external knowledge bases to generate 3D scenes from sketches. However, these methods heavily depend on external libraries and interfaces, which lack flexibility and face challenges like resource maintenance, copyright disputes, and network security issues(Gao et al., 2014).

We explored how to leverage LLM pre-training knowledge to autonomously guide complex geometric constraints. Our evaluation compared the spatial construction and geometric relationship understanding abilities of GPT-3.5-turbo and GPT-4, revealing that GPT-4 excels in spatial construction tasks due to its superior performance. we also introduced an innovative multi-agent approach for 3D scene construction, establishing geometric conventions at three levels (center, axis, and surface) to standardize the spatial relationships of 3D objects as understood by LLMs. This multi-level graph-driven approach enhances the spatial understanding and reasoning capabilities of LLMs. The workflow ensures information consistency and uniformity, mitigating data silos and redundancy issues, while enabling LLMs to explore their ability to understand geometric relationships of physical world.

2 Related Work

2.1 Generation Based On 3D Graphics

The application of GANs and VAEs in 3D scene generation has made notable progress in recent years. Chan et al. (2022) provides a method which can synthesize high-resolution, multi-view consistent images in real-time and also generate high-quality 3D geometry. Xie et al. (2019) proposes a context-aware convolutional neural network to reconstruct 3D voxel models from single and multi-view images. This method uses GANs to enhance the detail and structural accuracy of the generated 3D models. Wu et al. (2016) combines GAN for generating and controlling 3D objects, producing high-quality 3D models with shape control. Groueix et al. (2018) introduces a 3D surface generation method by learning a collection of 2D maps to approximate 3D surfaces, handling irregular topologies.Besides, Tang et al. (2024) find a method to use 2D diffusion model which can further control the generated content and inject reference-view information for unseen views.

These works typically offer high quality and realism, creativity, and diversity in generated content. However, they also face challenges such as high data dependency, complexity, and computational intensity.Moreover, such work often overlooks the complex geometric relationships between objects in the physical world.

2.2 Generation Based On External Libraries

The quality and availability of numerous 3D models have significantly improved. Tang et al. (2024) provide a large amount of 3D materials. And Zhou et al. (2018) provide an open-source library that supports rapid development of software for processing 3D data.It benefits research that utilizes LLMs to invoke open-source models and achieve scene graph construction. Sun et al. (2023), based on a multi-agent system, call the Blender interface to generate 3D scene images according to user requirements. SceneX (Zhou et al., 2024) employs LLMs to drive procedural modeling, utilizing Blender APIs and a vast array of procedural assets. Wu et al. (2024) offer an approach that combines user sketches with external knowledge, progressively generating 3D scenes through a scene diffusion model. Their work demonstrates how these agents can leverage external tools and model libraries to automate the construction and understanding of scene graphs.

Utilizing existing model libraries offers significant advantages in terms of efficiency, scalability, and flexibility in scene generation. However, due to the heavy reliance on external libraries and external materials, the work in question exhibits inconsistent material quality, poses high maintenance complexity, demonstrates insufficient flexibility, and involves copyright challenges.

3 Method

3.1 Graph Runs Through the Entire Workflow

Multi-agent systems have demonstrated effective performance in segmenting complex problems into numerous sub-problems and resolving them (Grossi et al., 2023), aligning with the step-by-step decomposition of three-dimensional scene concepts and the meticulous refinement of generated content at each stage in this work. And implementing information alignment between proxy groups is a huge challenge(Han et al., 2024). Inspired by Qi et al. (2023) and Ranasinghe et al. (2024), we choose graph database as the medium. In our work, we use GPT-4 (OpenAI, 2023b) as the basis for the agent and Neo4j (Neo4j, 2023) database to store our graph. By employing a graph database to capture spatial information and representing shapes and their geometric relationships with nodes and edges, complex geometric relationships can be managed flexibly. The graph database records scene information, providing a comprehensive overview of user objectives and scene graphs throughout the workflow. This ensures that generated scenes align with predefined spatial constraints and design specifications by integrating relational processing with large model generation capabilities, offering a flexible and efficient solution for managing complex spatial data and scene generation.

3.1.1 Scenery Designer

Graph databases can stably and comprehensively record object information in existing scenes, thereby reducing scene graph generation errors caused by illusions or memory problems in LLMs, such as reconstructing existing objects or using non-existent objects as reference points. By providing detailed scene information to LLMs, the graphics database helps to develop plans that are consistent

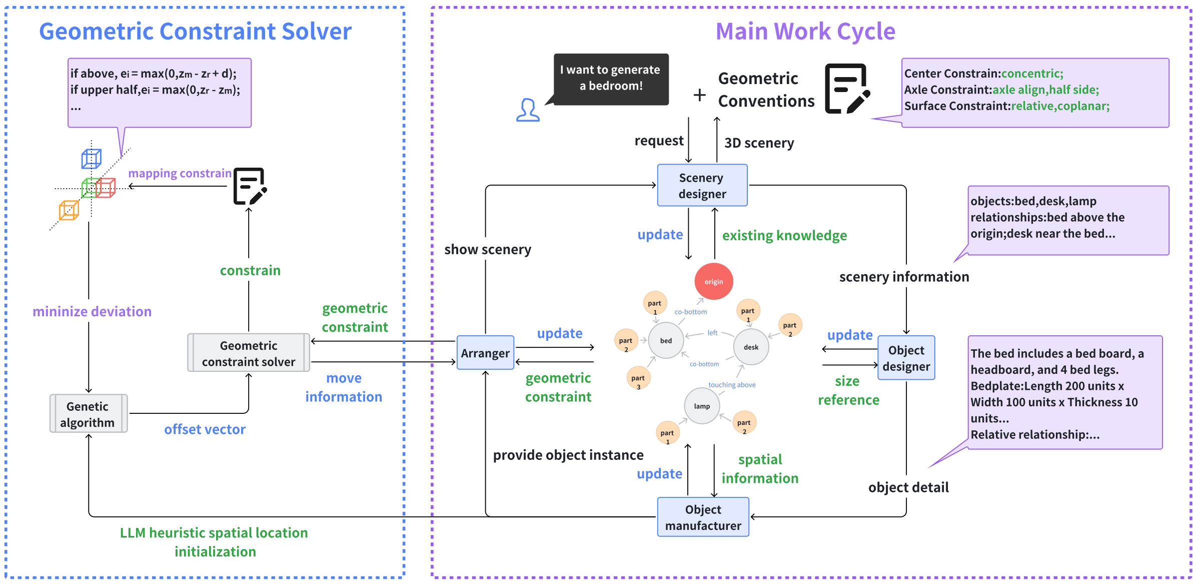

Geometric Constraint Solver
if above, $e_i = \max(0, z_m - z_r + d)$;
if upper half, $e_i = \max(0, z_r - z_m)$;
...
mapping constrain
constrain
mininize deviation
geometric constraint
Geometric constraint solver
move information
Genetic algorithm
offset vector
LLM heuristic spatial location initialization
Main Work Cycle
I want to generate a bedroom!
+ Geometric Conventions
Center Constrain:concentric;
Axle Constraint:axle align,half side;
Surface Constraint:relative,coplanar;
request
3D scenery
Scenery designer
update
existing knowledge
show scenery
objects:bed,desk,lamp
relationships:bed above the origin;desk near the bed...
scenery information
origin
part 1
co-bottom
part 1
part 2
part 2
bed
left
desk
co-bottom
part 3
touching above
lamp
part 1
part 2
Arranger
update
geometric constraint
update
size reference
Object designer
The bed includes a bed board, a headboard, and 4 bed legs.
Bedplate:Length 200 units x Width 100 units x Thickness 10 units...
Relative relationship:...
provide object instance
update
spatial information
object detail
Object manufacturer

Figure 1: The entire workflow is based on geometric conventions and relies on multiple agents to carry out 3D scene construction work around the graph. The user's demand information will be refined layer by layer by designers and used to generate object instances. Finally, the arranger will use the mapping from geometric constraints to deviations and a genetic algorithm solver to determine the correct placement position of the object.

with the given semantics and do not conflict with the current scene graph. Based on this, the scene designer will mobilize their internal world knowledge to design a scene that is semantically consistent with the input, including the main objects in the scene and the spatial geometric relationships between objects.

3.1.2 Object Designer

After the scene planning is completed, the object designer needs to design objects with appropriate structure and size based on the existing reference objects in the scene. On the one hand, image databases are needed to provide background information, and on the other hand, LLMs themselves require a certain level of common sense knowledge and reasoning ability to lay a more detailed foundation for the next step of object creation.

3.1.3 Object Manufacturer

After completing the object design phase, we proceed to the construction phase. At this stage, LLMs require a thorough understanding of the descriptive statements used by object designers, particularly those describing the interrelationships between internal modules of the object. This ensures alignment between the generated objects and their descriptive statements. We have observed that models with weaker performance, such as the GPT-3.5 turbo(OpenAI, 2023a), often have poor performance in this step, regardless of the level of detail provided by the designer. Additionally, to minimize the risk of spatial divergence when using genetic algorithms in later permutation calculations, the initial position of the object should be proximate to its main reference object, typically adhering to their relative spatial relationships. Here, a graphical database becomes crucial, as it offers detailed information about the size and position of reference objects, as well as their approximate relative relationships. This information is essential to guide LLMs in utilizing their internal knowledge effectively.

3.1.4 Arranger

Following the construction of the object, further optimization of its spatial position is required to meet specific spatial requirements, such as those related to smaller particle sizes. Initially, the relationship information between the newly constructed object and the reference object must be extracted from the graph database. This information is then used to perform further inference and to supplement any missing spatial constraints. Based on these completed spatial constraints, the appropriate constraint equations can be selected for positional optimization.

The graph database provides a comprehensive understanding of global scene information at each layer of the workflow and provides necessary information for each layer to complete tasks. It can efficiently manage complex relationships and dependencies, enabling each level to accurately locate and process relevant information in complex scenarios.

3.2 Geometric Conventions

Inspired by the work of Hedau (2011) and Klein (1998), we recognize that clearly and systematically representing the relative positions of objects in space is beneficial for enhancing the spatial reasoning capabilities of LLMs. Consequently, we have devised a spatial convention that encompasses three levels of constraint relationships: geometric center, axis, and surface, with varying degrees of constraint strength.By integrating different spatial conventions, we can flexibly and accurately determine the positions of objects within a reasonable range. This set of spatial conventions is integral to our entire workflow. Through the implementation of a unified spatial convention system, we ensure consistency and standardization throughout the workflow.

An example of the spatial convention we designed is as follows:

3.2.1 Geometric Center Relationship Constrain

- Concentric relationship:

$$x_m^c = x_r^c, \quad y_m^c = y_r^c \quad \text{and} \quad z_m^c = z_r^c \quad (1)$$

3.2.2 Axle Relationship Constraint

- x align:

$$x_m^c = x_r^c \quad (2)$$

- front half:

$$x_{\mathrm{r}}^c > x_{\mathrm{m}}^c \quad (3)$$

3.2.3 Surface Relationship Constraint

- front:

$$x_r^b - x_m^f = d \quad (4)$$

- coplanar front:

$$x_r^t = x_m^t \quad (5)$$

To avoid misunderstandings, we briefly declare the following symbols:

- x, y and z represent the projections of the corresponding parts of the object on that axis
- In superscripts, f, b and t, etc. respectively represent the corresponding surfaces of the object, such as the front, back/bottom, and top surfaces. And c represents the geometric center.
- In the subscript, r and m represent the reference object and the object to be moved, respectively. And d stands for distance.

3.3 Graph Driven LLM Spatial Inference

The final layer of the workflow is called the arranger, responsible for the spatial arrangement of generated objects in the scene. Wei et al. (2024)discussed Detailed introduction on how to construct a knowledge graph of geographic spatial data, as well as how to express and infer spatial relationships. Inspired by this, this work maps the relative positional relationships of objects to a graphics database. By setting strong and weak reference objects, we provide different levels of constraints for the object to be moved. With the continuous enrichment of graphic information, our framework will provide increasingly accurate spatial constraints. After determining the spatial constraints, the LLM inspired genetic algorithm is used to solve the spatial constraints, which is used to update the spatial position of the object to be moved and dynamically update the graphic data. This layer utilizes a graphical database to store entities and their spatial relationships, establishing and updating spatial constraints at the granularity of objects. The process specifically includes several steps:

3.3.1 Graph Database Interaction

Arranger interacts with graphical databases to generate more detailed relationship information and select the correct constraint equation according to it. Based on the provided rough relationship pairs, the arranger select the strong reference object which will provides 1 to 3 constrains from the graph database and return the weak reference objects which provides 0 to 2 constrains and be associated with the strong reference object. In this way, the computational complexity of constraints can be reduced. The LLM agent will obtain various types of information about the reference object, including its dimensions and spatial positions. It will then infer and add new spatial constraints within the basic spatial constraint framework and select the correct constraint equation for genetic algorithm calculation of accurate spatial positioning.

3.3.2 Genetic Algorithm for Solving Geometric Relationships

Given the global optimization capabilities of the genetic algorithm and its effective use with heuristic initialization, we ultimately opted for the genetic algorithm to address the spatial constraints. When LLM completes spatial constraints and selects the correct geometric equation, the permutator pass the parameters to the genetic algorithm(Shapiro, 1999) solver to optimize the geometric relationships and further adjust and update the spatial position of the objects initialized by LLM.

Each object is composed of multiple blocks, with each block represented by its centroid coordinates and three-dimensional dimensions. The specific representation is as follows:

Single block representation:

$$b_i = \{c_i, d_{i1}, d_{i2}, d_{i3}\}$$

where $c_i = (x_i, y_i, z_i)$ is the centroid coordinates, and d_{i1}, d_{i2}, d_{i3} represent the length, width, and height, respectively.

Object representation:

$$O_i = \{b_{i1}, b_{i2}, \ldots, b_{in}\}$$

where O_i represents an object composed of multiple blocks b_{ij}. In addition, the spatial information of objects can also be represented as follows:

$$O_i = \{C_i, D_{i1}, D_{i2}, D_{i3}\}$$

where C_i is the centroid coordinates, and D_{i1}, D_{i2}, D_{i3} represent the length, width, and height of O_i respectively.

We define various types of spatial constraints to describe the relative spatial relationships between objects. Below are examples of above, and upper half:

$$\text{above} : z_m^b \geq z_r^t + d$$

$$\text{upper half} : z_m^c \geq z_r^c$$

To generate appropriate constraint equations, we abstract the reference object as a block and generate movable object pairs with reference part relationships for each object. Then, based on the generated relationship pairs, we generate appropriate constraint equations and pass them to the genetic algorithm for solution.

Assume we have multiple reference objects R_k and a movable object M, each pair $(R_k, \text{relation}, M)$ can be represented as a set of constraint formations:

$$e_i = \begin{cases} \max\left(0, z_m^c - z_r^c + d\right), \text{if above} \\ \max\left(0, z_r^c - z_m^c\right), \text{if upper half} \end{cases} \tag{6}$$

The optimization goal is to minimize the total error:

$$\min E = \min \sum_{i=1}^{N} e_i^2$$

To determine effective motion vectors, we employed a genetic algorithm inspired by LLM initialization. Objects are generated at specific positions based on global and reference content, partially fulfilling constraint requirements. The algorithm's initialization is then refined based on the size of both the reference object and the object to be moved, enhancing the optimization process. Each genome consists of three XYZ coordinates representing motion vectors. The total error E of each individual is calculated to assess fitness, with top-performing individuals selected for crossover and mutation. During crossover, parent DNA combines to produce new offspring, and mutations make fine adjustments to coordinates. This process iterates until a set number of generations or error convergence is achieved, gradually approaching the optimal solution.

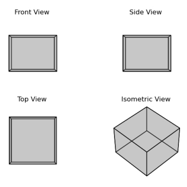
4 Experiment

In this section, we will discuss the factors affecting the quality of the 3D scene graph generated by the LLM from two aspects. The first influencing factor is the model's ability. We test the generation performance of the base models GPT-3.5-Turbo and GPT-4 without using the framework. The second influencing factor is the degree of integration with the work framework. We set up three sets of experiments to explore the complete use of the work framework, including ablation experiments to analyze the impact of removing certain components.

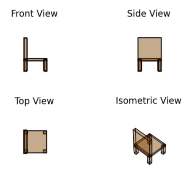
4.1 Model Performance Impact

Our experiment found a strong correlation between LLM performance and spatial understanding. Evaluating GPT-3.5-Turbo and GPT-4-0125 on object and scene generation tasks, we observed that GPT-3.5 had poor spatial comprehension and simplistic outputs. In contrast, GPT-4 showed improved spatial concepts and multi-object scene generation but still used simple blocks with limited detail.

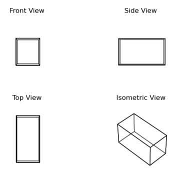
4.1.1 Object Generation



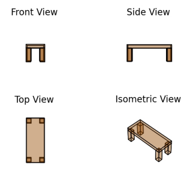
(a) Chair(3.5)



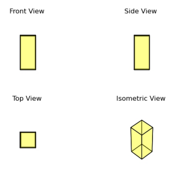
(b) Chair(4)



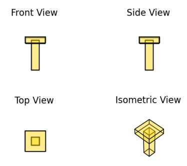
(c) Table(3.5)



(d) Tabel(4)



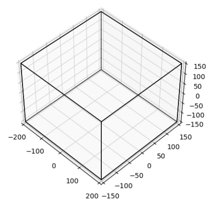
(e) Lamp(3.5)



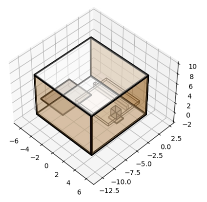
(f) Lamp(4)

Figure 2: GPT-4 produces complex structures and details and achieves better semantic alignment than GPT-3.5.

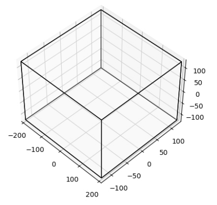
4.1.2 Scenery Generation



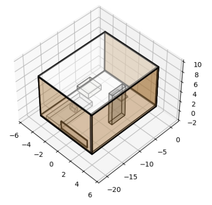
(a) Bedroom(3.5)



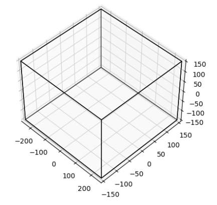
(b) Bedroom(4)



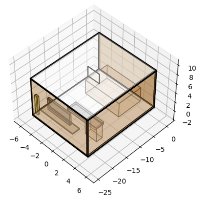
(c) Kitchen(3.5)



(d) Kitchen(4)



(e) Living(3.5)



(f) Living(4)

Figure 3: GPT-4 shows better spatial comprehension and multi-object scene generation than GPT-3.5, but still uses simple blocks with limited detail.

4.2 Analysis And Comparison

Metric: We choose CLIP (Radford et al., 2021)to calculate the similarity between the generated object and scene images and text, in order to evaluate the alignment between the text and the generated content. In addition, during the experimental process, there is often a large amount of overlap or object isolation in the generated failed scene images. Therefore, for the scene, we additionally introduced overlap score and isolation score, corresponding to the proportion of overlapping volume to the total volume of all objects and the proportion of isolated blocks to the total block, respectively.

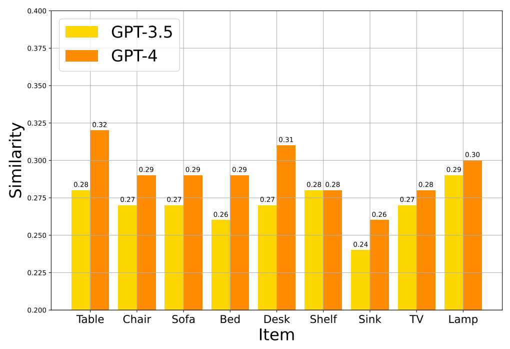


Figure 4: In object level generation tasks, the clip index of agents based on GPT-4 is generally better than ones based on GPT-3.5.

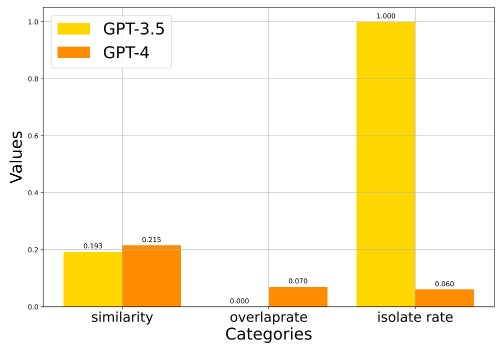


Figure 5: In the scenario level generation task, the clip index of GPT-4 group is **10.1%** higher than that of GPT-3.5 group, and its isolation rate is much better than that of GPT-3.5 group.

4.3 Framework Impact

Baseline Methods: The baseline we have chosen is a single agent without designed agents or graph driven methods, which showed in Figure 3. The base model of each agent is gpt-4-0125 preview with default temperature.

4.3.1 Ablation Study

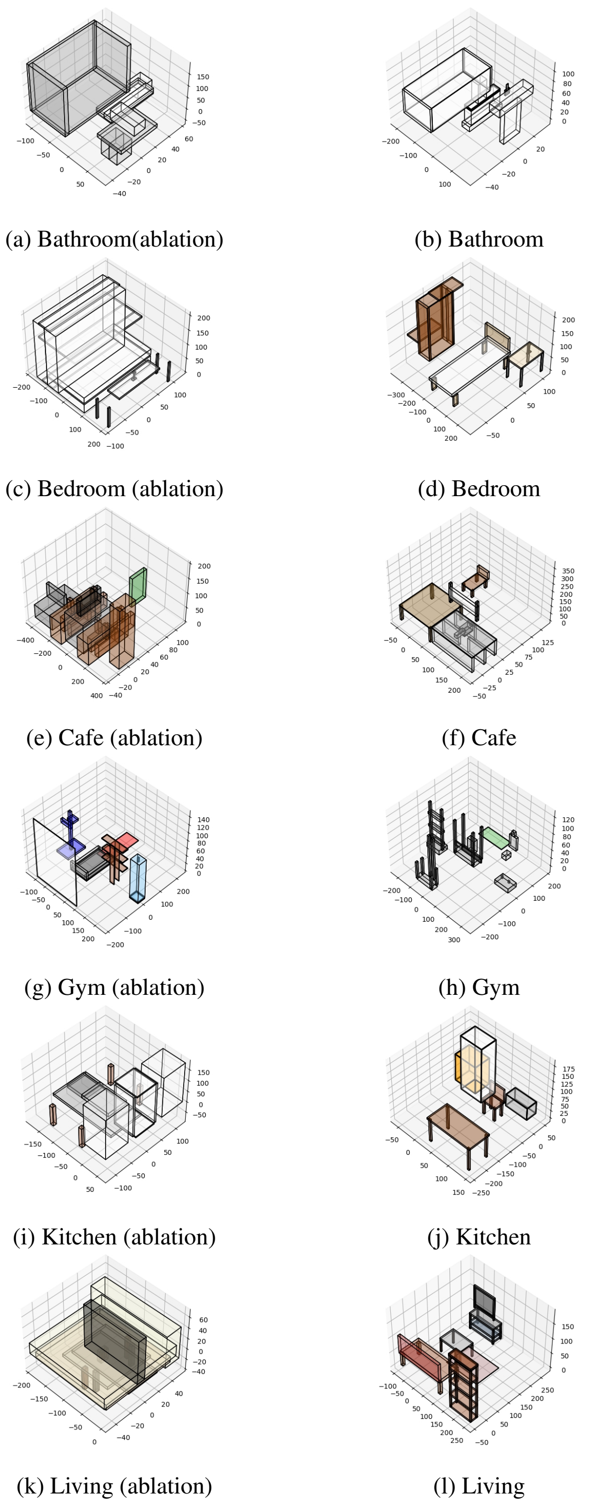


Figure 6: The ablation group showed detailed structures, but lacked reasonable spatial planning The non-ablated group can not only represent details of objects but also have a reasonable plan for the placement of objects.

In the ablation group experiment, we eliminated the interaction process between the graphical database and the workflow, while retaining the workflow of multi-agent collaboration. The non ablated group completely retained the graph reasoning framework.

4.4 Analysis And Comparison

The schematic diagram illustrates the performance of LLM scene graph generation in three modes. Images produced by the baseline method neglect object details but exhibit some overall spatial planning capability. The ablation group attempts to emphasize object details but lacks spatial planning, leading to overcrowded scenes. The non-ablated group excels in both object details and proper object placement.

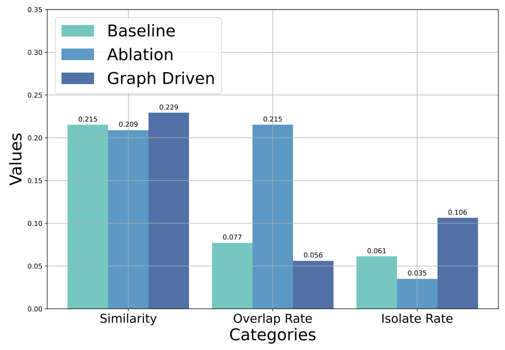


Figure 7: Comparison of metrics across different work modes indicates the following information: using graph driven workflows improves the similarity between images and text, with a decrease in spatial overlap rate but an increase in isolation rate

According to the above Figure 7, we found that in terms of clip similarity, the graph driven group performed better than both the baseline and ablation groups, and was generally better than both in a single task, with mean values **6.3%** and **8.7%** higher than the baseline and ablation groups, respectively. In terms of object overlap rate, it is lower than both, but in terms of isolation rate, it is higher than both.

5 Conclusion And Limitation

Our research provides an intuitive demonstration of the spatial understanding capabilities of LLMs and quantitatively evaluates the spatial comprehension of two distinct models. Additionally, we enhance the geometric understanding and spatial reasoning abilities of LLMs in complex physical environments by implementing well-defined geometric conventions and a graph-driven framework.

This study is conducted using a custom-developed sandbox platform, designed to present the spatial concepts understood by LLMs in a more intuitive and flexible manner. However, due to resource constraints, we are unable to test higher-performing models, which limits our ability to fully showcase the framework's potential in improving the spatial understanding of LLMs.

References

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133.

Lin Gao, Yan-Pei Cao, Yu-Kun Lai, Hao-Zhi Huang, Leif Kobbelt, and Shi-Min Hu. 2014. Active exploration of large 3d model repositories. *IEEE transactions on visualization and computer graphics*, 21(12):1390–1402.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Davide Grossi, Ariel Rosenfeld, and Nimrod Talmon. 2023. Advances in multi-agent systems research: Eumas 2021 extended selected papers. *SN Computer Science*, 4(5):587.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 216–224.

Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, Zhaozhuo Xu, and Chaoyang He. 2024. Llm multi-agent systems: Challenges and open problems.

Varsha Chandrashekhar Hedau. 2011. *3D spatial layout and geometric constraints for scene understanding*. University of Illinois at Urbana-Champaign.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Rüdiger Klein. 1998. The role of constraints in geometric modelling. In *Geometric Constraint Solving and Applications*, pages 3–23. Springer.

Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. 2024. Advances in 3d generation: A survey. *arXiv preprint arXiv:2401.17807*.

Inc. Neo4j. 2023. *Neo4j Documentation*.

OpenAI. 2023a. Gpt-3.5 turbo.

OpenAI. 2023b. Gpt-4 technical report. *OpenAI*.

Jianzhong Qi, Zuqing Li, and Egemen Tanin. 2023. Maasdb: Spatial databases in the era of large language models (vision paper). In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '23. ACM.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Kanchana Ranasinghe, Satya Narayan Shukla, Omid Poursaeed, Michael S. Ryoo, and Tsung-Yu Lin. 2024. Learning to localize objects improves spatial reasoning in visual-llms.

Jonathan Shapiro. 1999. Genetic algorithms in machine learning. In *Advanced Course on Artificial Intelligence*, pages 146–168. Springer.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 2023. 3d-gpt: Procedural 3d modeling with large language models. *arXiv preprint arXiv:2310.12945*.

Zhenyu Tang, Junwu Zhang, Xinhua Cheng, Wangbo Yu, Chaoran Feng, Yatian Pang, Bin Lin, and Li Yuan. 2024. Cycle3d: High-quality and consistent image-to-3d generation via generation-reconstruction cycle.

Bo Wei, Xi Guo, Xiaodi Li, Ziyan Wu, Jing Zhao, and Qiping Zou. 2024. Construct and query a fine-grained geospatial knowledge graph. *Data Science and Engineering*, 9(2):152–176.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in Neural Information Processing Systems*, 29.

Zijie Wu, Mingtao Feng, Yaonan Wang, He Xie, Weisheng Dong, Bo Miao, and Ajmal Mian. 2024. External knowledge enhanced 3d scene generation from sketch. *arXiv preprint arXiv:2403.14121*.

Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. 2019. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2690–2698.

Mengqi Zhou, Jun Hou, Chuanchen Luo, Yuxi Wang, Zhaoxiang Zhang, and Junran Peng. 2024. Scenex: Procedural controllable large-scale scene generation via large-language models. *arXiv preprint arXiv:2403.15698*.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3d: A modern library for 3d data processing.

A Geometric conventions

A.1 Geometric Center Relationship Constrain

1. **concentric**: Concentric.
    - Calculation: The Euclidean distance between the centers of the two objects.

A.2 Axle Relationship Constraint

A.2.1 Align Relationship

1. **x aligned**: X-aligned.
    - Calculation: The alignment error in the x direction between the two objects.
2. **y aligned**: Y-aligned.
    - Calculation: The alignment error in the y direction between the two objects.
3. **z aligned**: Z-aligned.
    - Calculation: The alignment error in the z direction between the two objects.

A.2.2 Half Side Relationship

1. **left half**: Left half.
    - Determine if the `ref center` object is in the left half of the `mov center` object.
2. **right half**: Right half.
    - Determine if the `ref center` object is in the right half of the `mov center` object.
3. **upper half**: Upper half.
    - Determine if the `ref center` object is in the upper half of the `mov center` object.
4. **lower half**: Lower half.
    - Determine if the `ref center` object is in the lower half of the `mov center` object.
5. **front half**: Front half.
    - Determine if the `ref center` object is in the front half of the `mov center` object.
6. **back half**: Back half.
    - Determine if the `ref center` object is in the back half of the `mov center` object.

A.3 Surface Relationship Constraint

A.3.1 Relative Positioning Relationship

1. **left**: `mov center` object is to the left of the `ref center` object.
    - Calculation: The distance between the left edge of the `ref center` object and the right edge of the `mov center` object minus the given `distance`.
2. **right**: `mov center` object is to the right of the `ref center` object.
    - Calculation: The distance between the left edge of the `mov center` object and the right edge of the `ref center` object minus the given `distance`.
3. **above**: `mov center` object is above the `ref center` object.
    - Calculation: The distance between the bottom edge of the `mov center` object and the top edge of the `ref center` object minus the given `distance`.
4. **below**: `mov center` object is below the `ref center` object.
    - Calculation: The distance between the bottom edge of the `ref center` object and the top edge of the `mov center` object minus the given `distance`.
5. **front**: `mov center` object is in front of the `ref center` object.
    - Calculation: The distance between the back edge of the `mov center` object and the front edge of the `ref center` object minus the given `distance`.
6. **back**: `mov center` object is behind the `ref center` object.
    - Calculation: The distance between the back edge of the `ref center` object and the front edge of the `mov center` object minus the given `distance`.

A.3.2 Coplanar Relationship Constrain

1. **coplanar top**: Coplanar on top.
    - Determine if the top edges of the two objects are coplanar.
2. **coplanar bottom**: Coplanar on the bottom.

- Determine if the bottom edges of the two objects are coplanar.

3. **coplanar left**: Coplanar on the left.
    - Determine if the left edges of the two objects are coplanar.
4. **coplanar right**: Coplanar on the right.
    - Determine if the right edges of the two objects are coplanar.
5. **coplanar front**: Coplanar in front.
    - Determine if the front edges of the two objects are coplanar.
6. **coplanar back**: Coplanar in the back.
    - Determine if the back edges of the two objects are coplanar.

B Objects Generated With Workflow

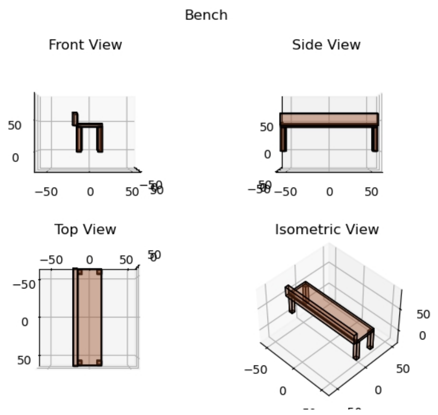


Figure 8: Bench

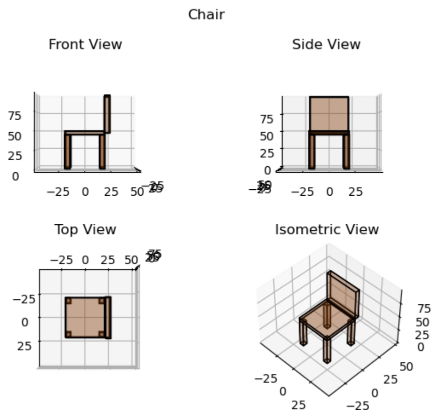


Figure 9: Chair

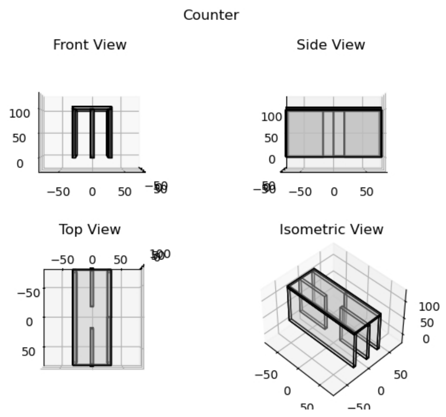


Figure 10: Counter

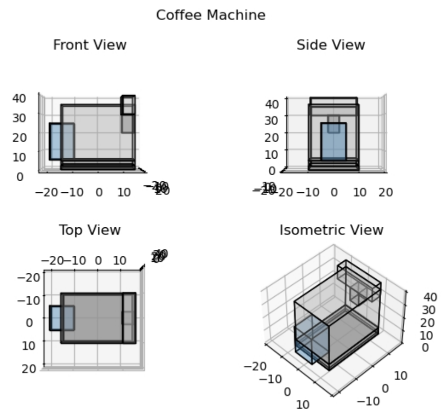


Figure 11: Coffee Machine

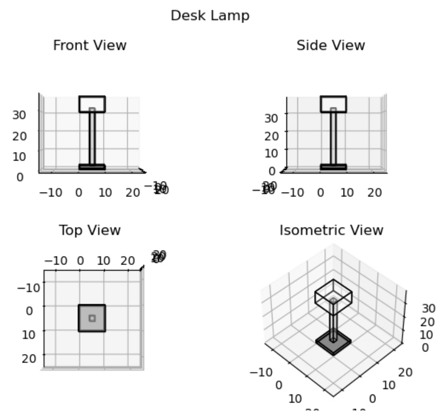


Figure 12: Lamp

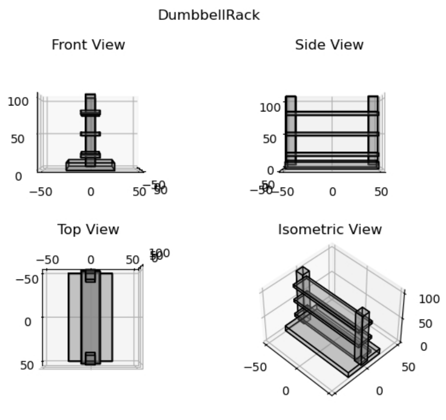


Figure 13: Dumbbell Rack

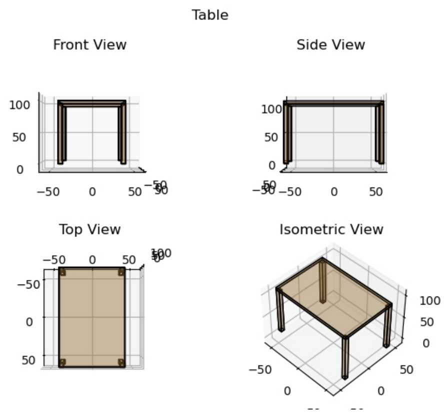


Figure 16: Table

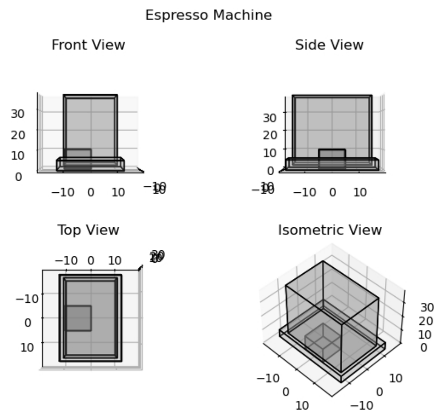


Figure 14: Espresso Machine

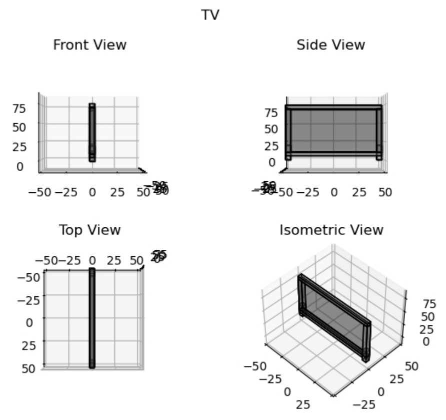


Figure 17: TV

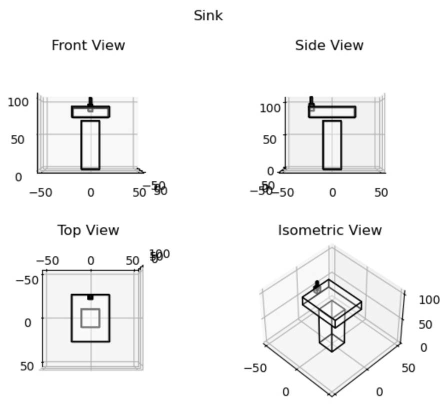


Figure 15: Sink

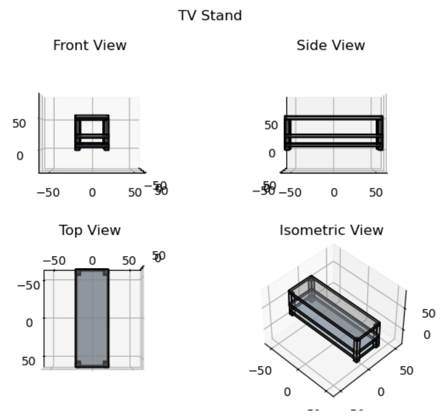


Figure 18: TV Stand