CS326. Rodionov Fedor. Final report.

“Using Visual Language Model for Room Layout Generation Problem.”

Problem definition

Room Layout Generation Problems (RLGP) involve creating optimal interior layouts by placing various objects in a room while adhering to specific constraints, such as non-overlapping placements, object size restrictions, aesthetic coherence, comfort and cozy, etc. While humans can easily distinguish between good and bad layouts, deep learning models struggle to capture the complex spatial relationships between objects and ensure that they are placed logically. Even advanced architectures like Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformers face challenges when generating room layouts.

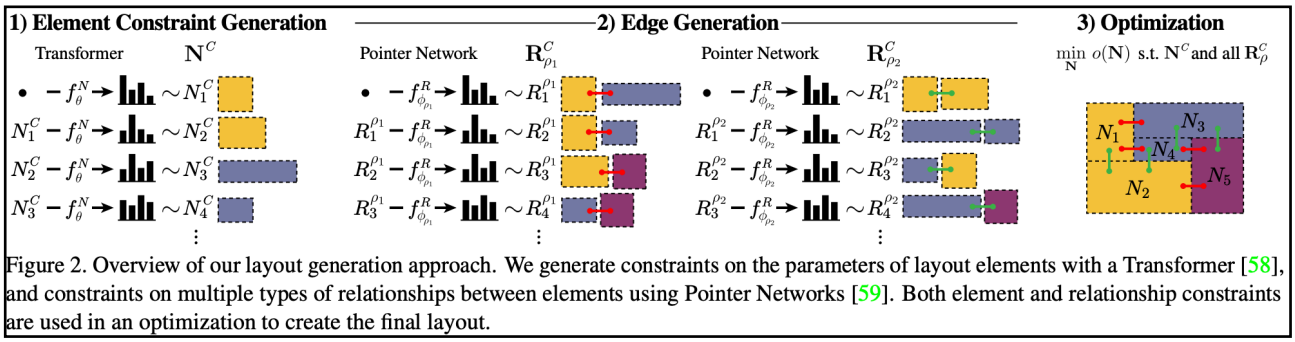


Figure 2. Overview of our layout generation approach. We generate constraints on the parameters of layout elements with a Transformer [58], and constraints on multiple types of relationships between elements using Pointer Networks [59]. Both element and relationship constraints are used in an optimization to create the final layout.

Figure 1. Example scheme of an algorithm utilizing Transformers **[1]**

RLGP can be approached in two main ways:

1. **Direct Generation of Bounding Boxes**: This involves directly generating bounding boxes representing each room object. For example, each object corresponds to a tuple with the following fields: (length, width, x_position, y_position, rotation_angle).

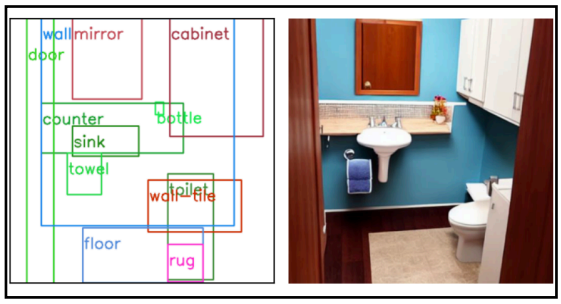


Figure 2. Bounding Boxes

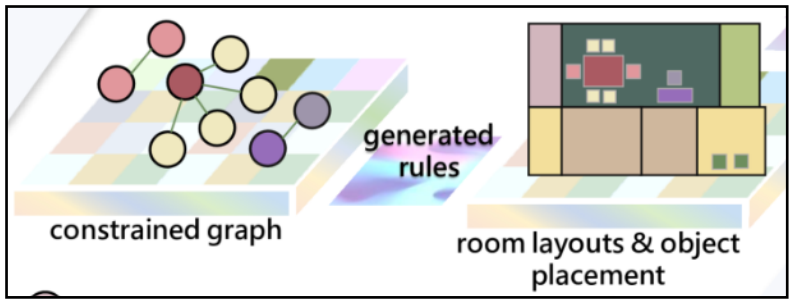


Figure 3. Constrained Graph

2. **Constraint Graph Representation**: A more sophisticated approach where a constraint graph models the relationships between objects (position relationships: in front of, side of, above, on top of; alignment: center-aligned; rotation: face to; etc.). This representation enables more accurate and flexible layout generation, though it may be computationally more expensive. Typically, brute-force methods provide better results compared to optimization

algorithms because optimization often leads to local minima, where objects cluster near walls due to constraints in the graph's high degree of freedom.

An example of constraints could be obtained from the generation model and used in an optimization solver (Gurobi) or backtracking algorithm :

$$\begin{cases} \underline{w}_i \leq w_i \leq \overline{w}_i \\ \underline{d}_i \leq d_i \leq \overline{d}_i, \end{cases}$$

Figure 4. Size control

$$\begin{cases} x_i - w_j \geq x_j - M \cdot (1 - \sigma_{i,j}^R) \\ x_i + w_i \leq x_j + M \cdot (1 - \sigma_{i,j}^L) \\ y_i - d_j \geq y_j - M \cdot (1 - \sigma_{i,j}^F) \\ y_i + d_i \leq y_j + M \cdot (1 - \sigma_{i,j}^B) \\ \sum_{d=1}^{4} \sigma_{i,j}^d \geq 1, \end{cases}$$

Figure 5. Non overlapping

Related Works

I am currently exploring layout generation using LLMs, a method that has gained significant popularity and delivers promising results. For my analysis, I have selected three notable works as baselines.

- **LayoutGPT [2]**: One of the first approaches using LLM for layout generation and rather popular. This approach generates bounding boxes by leveraging the power of ChatGPT, followed by rendering these boxes in a scene. It shows good potential for automated layout generation but still struggles with complex relationships between objects. The article covers many layout generation problems, but we are interested only in indoor scene synthesis.

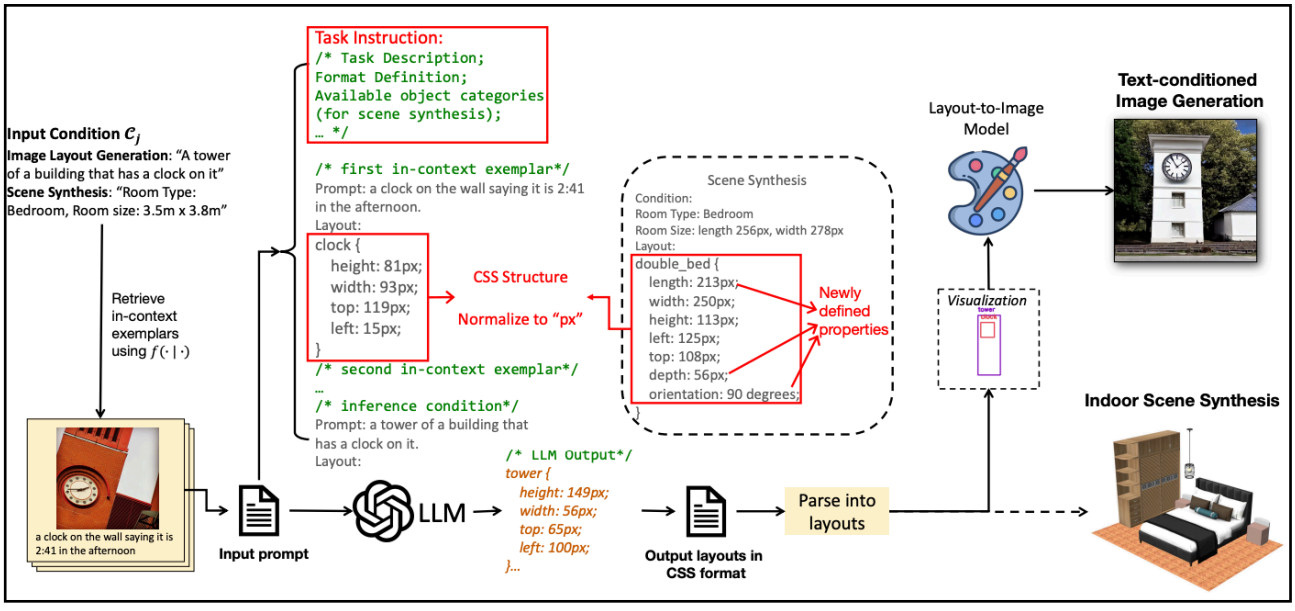


Figure 6. LayoutGPT scheme

- **I-Design [3]**: This method constructs a constraint graph, uses a brute-force algorithm to generate a layout, and then refines the scene by fetching objects based on text descriptions. While promising, this method also faces challenges with object placement accuracy and error-free output.

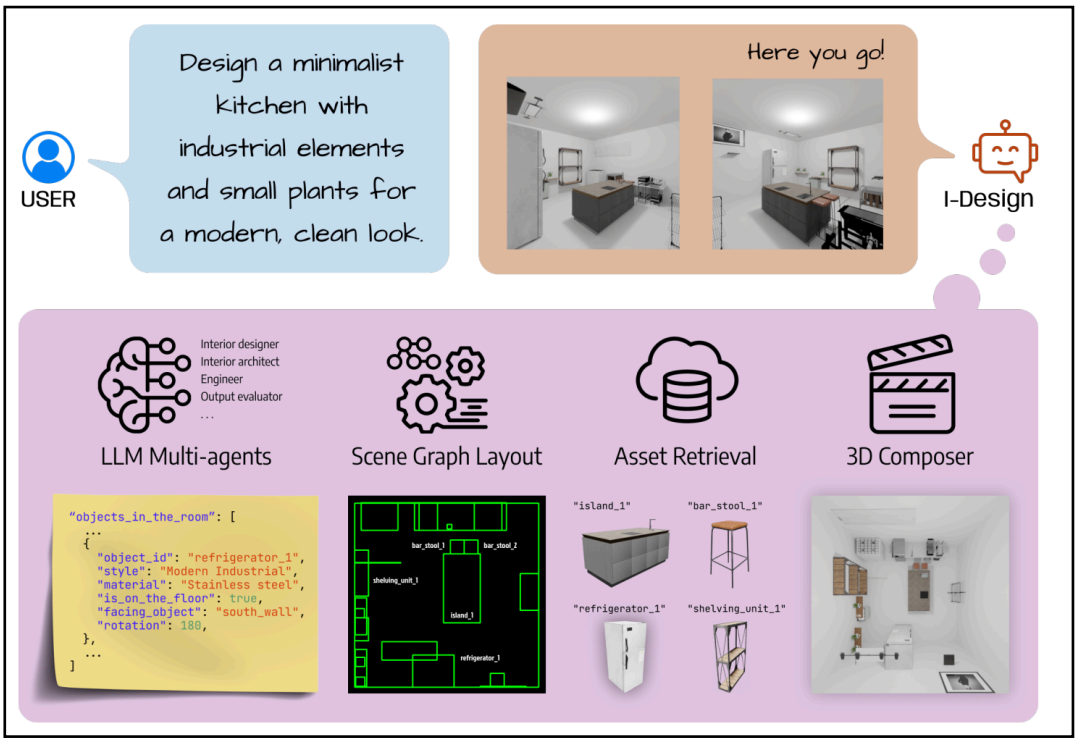


Figure 7. I-Design scheme

- **Holodeck [4]**: Holodeck is a promising recent approach. It achieves strong results through iterative, step-by-step interactions with the LLM. However, it often encounters issues such as object overlap and incorrect placements.

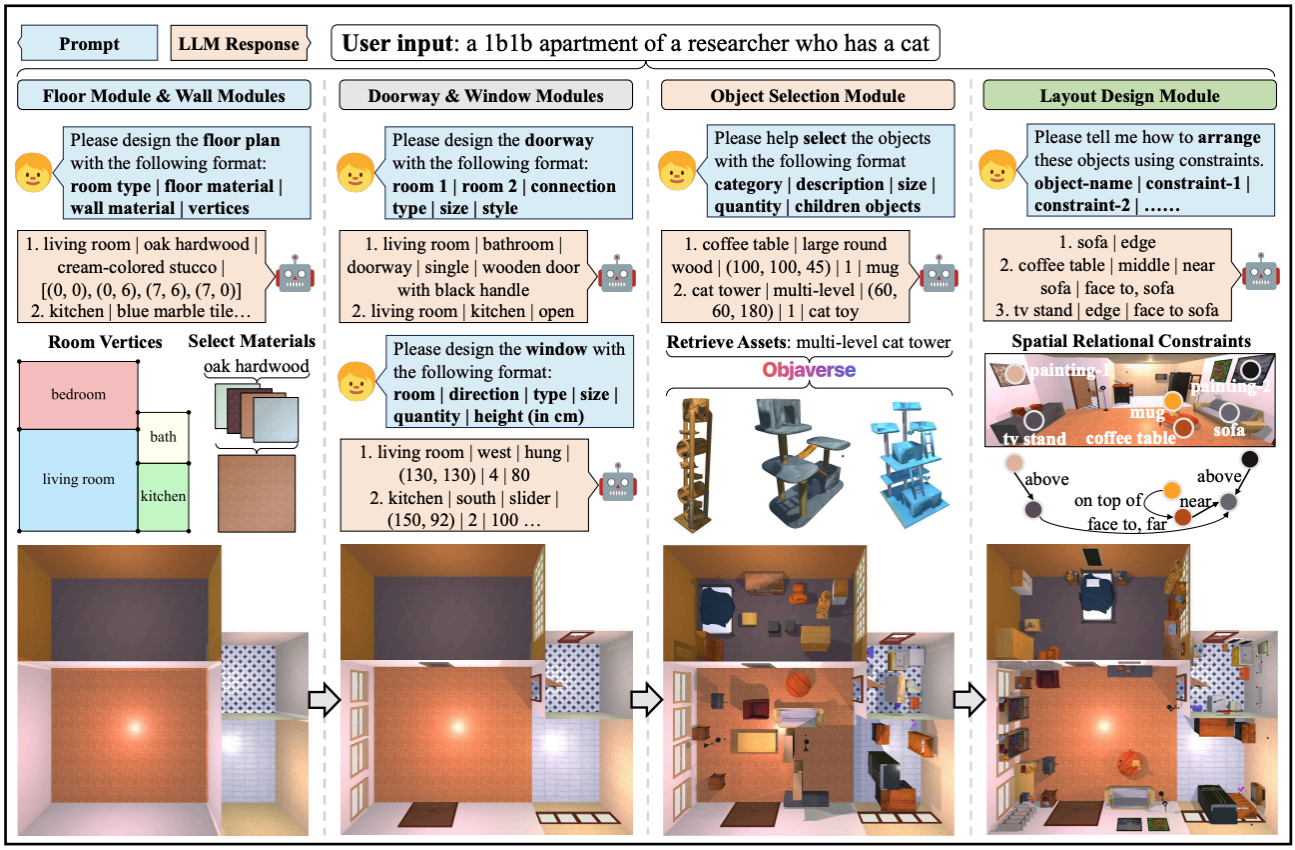


Figure 8. Holodeck scheme

Baseline results

I collected all baseline results for these methods during the first checkpoint. While each approach shows potential, it often requires significant human intervention to address layout inconsistencies and errors.

- **LayoutGPT**

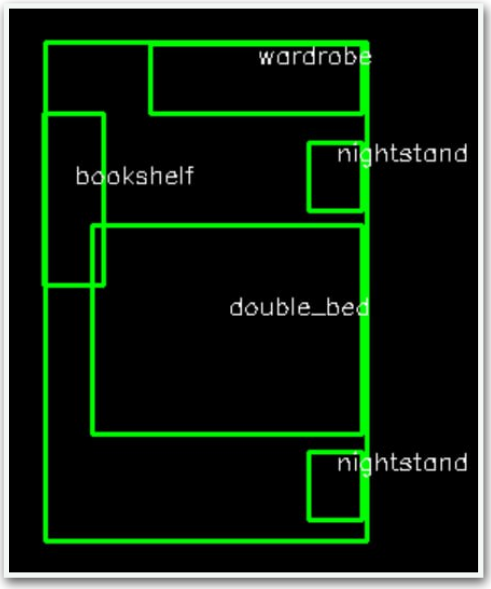


Figure 9. LayoutGPT - Bedroom

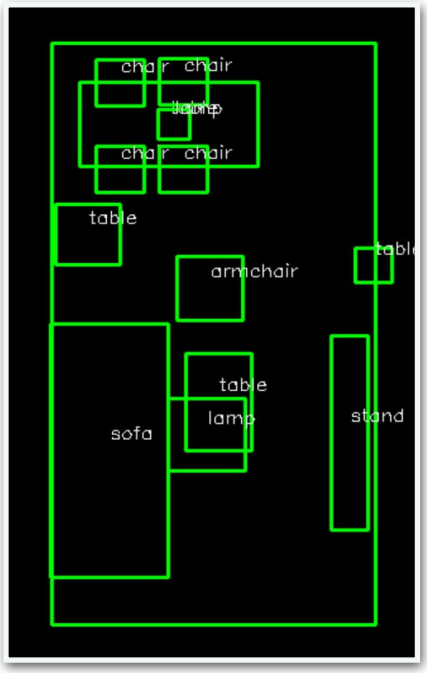


Figure 10. LayoutGPT - Living room

- **I-Design**

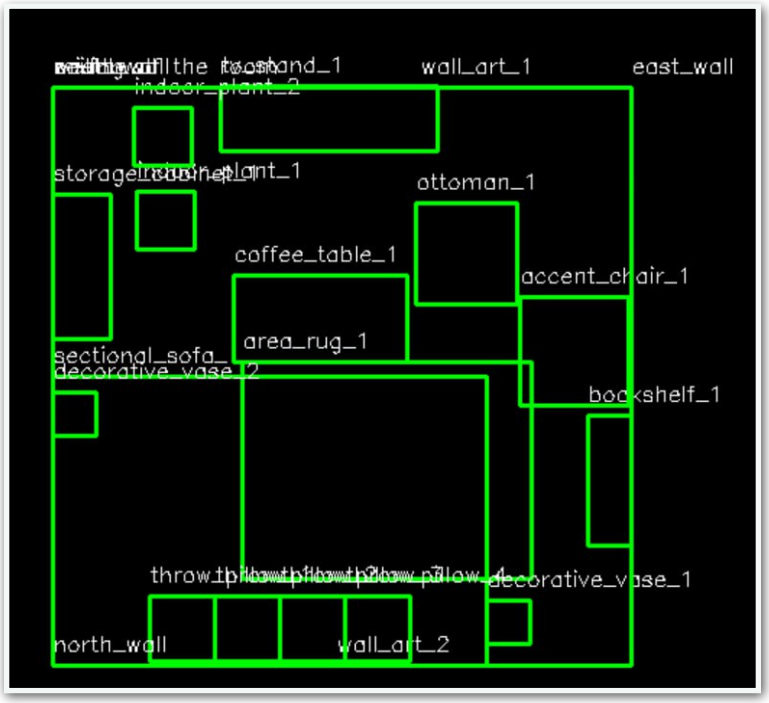


Figure 11. I-Design - Living room

- **Holodeck**

Figure 12. Holodeck - Bathroom

Figure 13. Holodeck - Living room

In LayoutGPT, object intersection errors occasionally occur. While using I-Design, I encountered instances where the optimization algorithm successfully found solutions, but at times, it was overloaded and failed to maintain proper proportions. In Holodeck, some objects appeared in front of windows, and a toilet was positioned in a corner, which did not align with real-life design principles.

Metrics

To compare generation results based on these articles, it's possible to use the following metrics for this task

- FID Score
- KL Divergence
- Out-of-Boundary (OOB) Error
- Bounding Box Loss (BBL)
- Average Number of objects (Nobj)
- GPT-4V Score
- Number of Shots - k (examples)
- Generalization to New Classes (B - bedroom, LR - living room, K - kitchen, Bathroom - Bath)

I obtained the following metrics using an evaluation technique based on open-source code. Initially, I planned to compare only the first two algorithms but later discovered Holodeck. While I did not collect metrics for Holodeck, I generated images and observed that it effectively avoids out-of-boundary issues and produces more diverse results. Holodeck can generate various room types (e.g., bedroom, living room, kitchen, bathroom) and even arbitrary scenes. It employs zero-shot prompting using internal LLM knowledge and achieves good results through step-by-step interaction. However, coherence can occasionally be lacking in complex scenes.

	FID score	KL div.	OOB error	BBL	Nobj	GPT-4V	K	Classes
Layout-GPT	56,66	0,14	38,1	5,15	6,9	-	8	B, LR
I-Design	60,12	-	0,0	0,33	14,1	-	8	B, LR

In I-Design, a brute-force algorithm is used to avoid out-of-boundary (OOB) errors, and the LLM is prompted to generate more objects. A common feature of both algorithms is that increasing the number of prompts helps generalize the concept and improves results. However, their application is limited to only two room types. The solutions are often infeasible for both methods or require multiple iterations to achieve a correct layout. There is no true distribution for I-Design.

I have not yet measured the GPT-4V score, which evaluates image realism based on ChatGPT's assessment. While obtaining this metric would be valuable, I have not implemented the necessary techniques, and the cost of API access makes it impractical for my current needs.

Motivation and Idea

These methods share a common challenge: extensive postprocessing due to strict constraints on object positioning and LLMs' limited ***spatial reasoning*** capabilities. This postprocessing is necessary because LLMs often struggle with spatial tasks. I came across an interesting study **[5]** that highlights how LLMs fail to determine relative positions when navigating random points and edges of figures. Several studies consistently emphasize these limitations.

The **goal** was to improve *spatial reasoning*, particularly for interior design. I aimed to incorporate additional visual information to generate more accurate spatial relationships. The first step involved collecting a high-quality dataset and leveraging VLLMs to achieve better results.
And later use some of the VLLM to generate better results.

There were three approaches to obtaining a suitable dataset for enhancing spatial reasoning:

1. **Use existing layouts**: Select layouts from previous approaches
2. **Generate a synthetic dataset with VLLM**: Use VLLMs to create datasets based on generated room layouts.
3. **Utilize existing synthetic datasets**: Apply generated layouts and create question-answer pairs to support training.

The first attempt was to collect a dataset using the current VLLMs

I chose not to use existing layout generation algorithms to avoid training the model on incorrect layouts. Instead, I aimed to generate a dataset using robust VLLM models. For example, feeding an image of a room along with its depth map could provide valuable additional information.
I initially developed a hybrid approach using a Visual-Language Model (VLLM) with the following steps:

- **Extract** text descriptions of a scene from multiple angles.
- **Combine** these descriptions to generate a comprehensive room layout, including brief item descriptions.

This method leverages both visual data and textual descriptions, enabling more accurate object retrieval and layout generation. Further improvements could be achieved by incorporating depth maps and object segmentation.

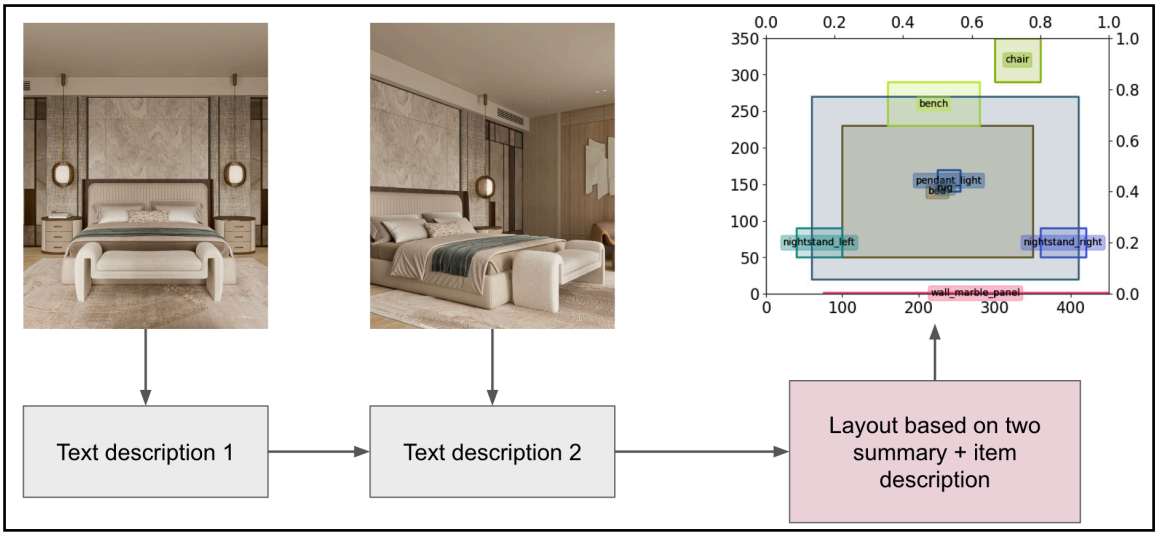


Figure 14. VLLM Layout extraction - 1

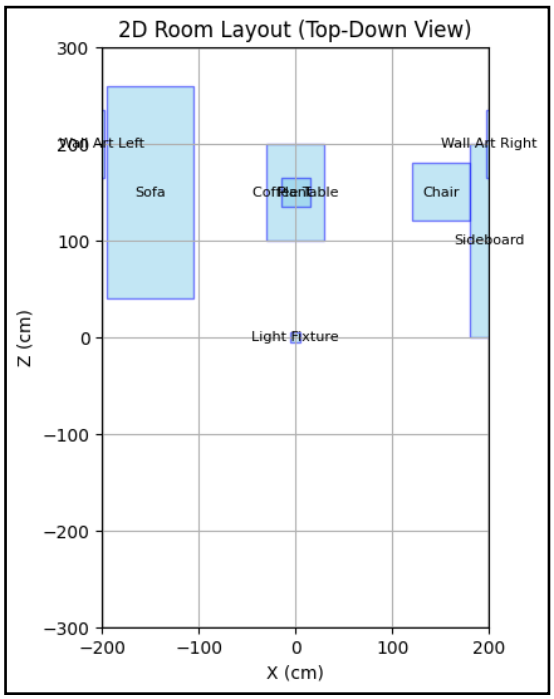
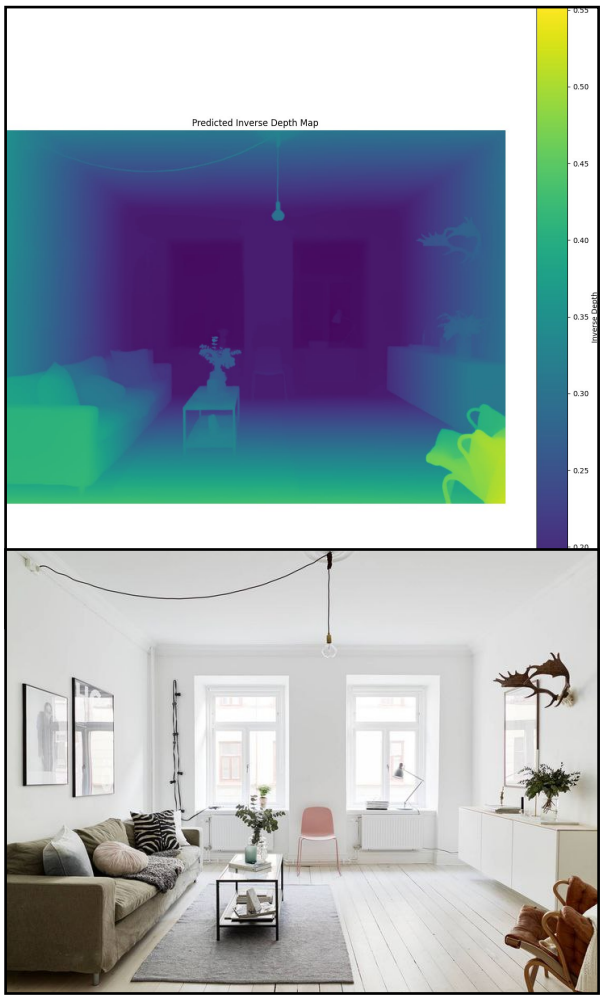


Figure 15. VLLM Layout extraction - 2



While the model described objects and their qualities well, it struggled significantly when asked about real sizes or distances between them, often generating incorrect layouts that didn't match the accompanying images. This limitation prompted me to explore another approach: using synthetic data.

I found valuable inspiration from a developer who improved the performance of small VLLMs in question-answering tasks by leveraging curated datasets and adding custom question-answer

pairs (Moondream **[6]**). Drawing from this approach, I decided to follow similar steps and focus on generating high-quality synthetic data for layout generation.

There are two excellent synthetic datasets available:
1. Hypersim **[7]**.
2. Infinigen **[8]**.

The Hypersim dataset includes 461 scenes created by professional architects and designers, offering multiple viewpoints, synthetic scene images, object segmentation, and depth maps. This data makes it possible to generate bounding boxes and 2D top-view layouts.

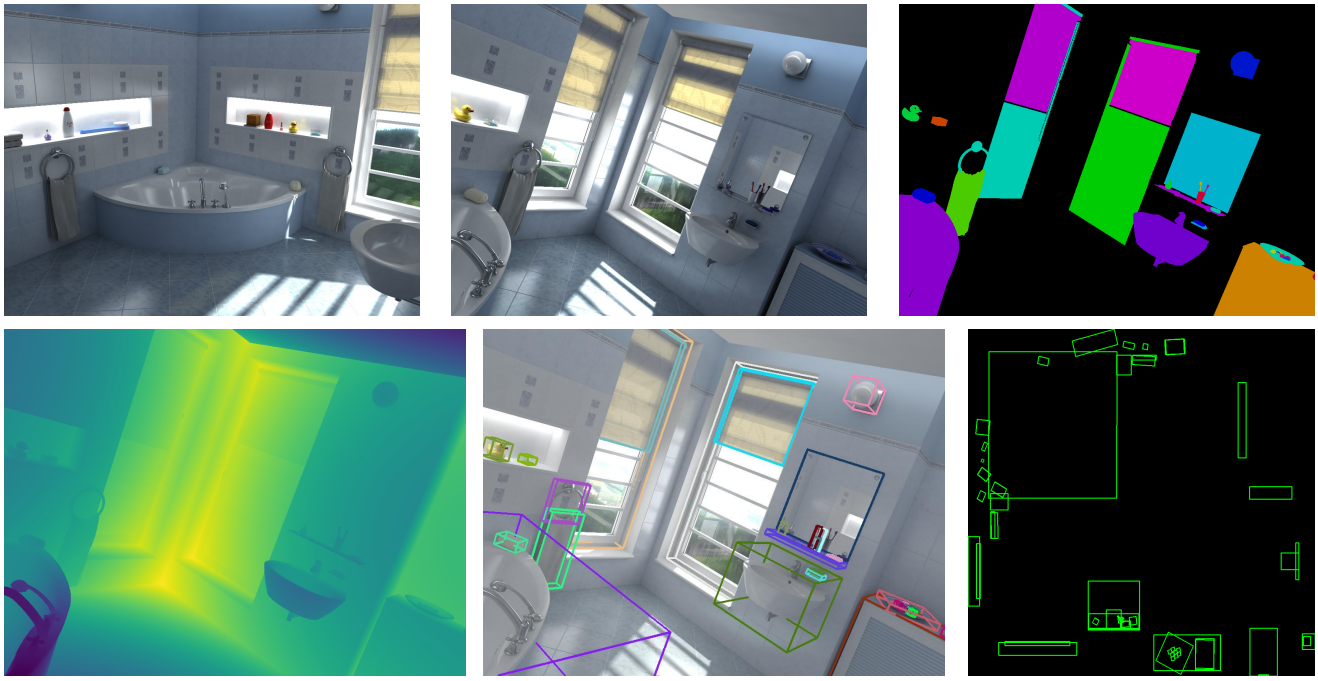


Figure 16. Example of Hypersim images

Fine-Tuning Vision Models - Part 1

The first step in my exploration was to assess the spatial reasoning abilities of smaller models. Building on the Moondream approach mentioned earlier. I aimed to evaluate how effectively these models could handle question-answering tasks related to object positions and sizes within a given scene.
To facilitate this, I collected two datasets designed to test the models' understanding of spatial relationships and their ability to answer questions regarding objects' relative positioning and dimensions. These datasets would serve as the foundation for evaluating the performance of the models in this specific context.

	Question	Answer	Image Path
0	Where is the bathtub located in relation to th...	The bathtub is positioned to the left of the w...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
1	What is the size of the sink compared to the b...	The sink is smaller in size compared to the ba...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
2	How far is the towel rack from the bathtub?	The towel rack is positioned a short distance ...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
3	Where are the shelves with objects located rel...	The shelves with objects are located above and...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
4	What is the distance between the window and th...	The window is positioned far enough away that ...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...

Figure 17. Question Answer (relative position) Dataset

Question: Where is the lamp located in relation to the bookshelf?

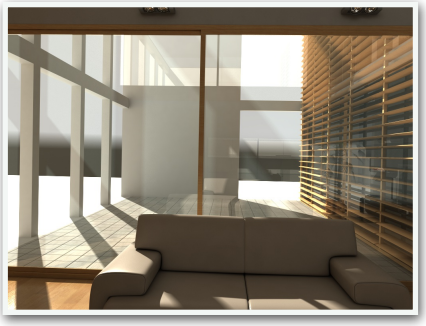
Ground Truth: The lamp is positioned to the left of the bookshelf.

Moondream: The lamp is located on the left side of the bookshelf.

Figure 18. Image example

	Question	Answer	Image Path
0	What is the size of the duck in cm? Length, wi...	Object duck has the following size:\nLength: 1...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
1	What is the size of the cotton_bud in cm? Leng...	Object cotton_bud has the following size:\nLen...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
2	What is the size of the towel1 in cm? Length, ...	Object towel1 has the following size:\nLength:...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
3	What is the size of the towel1 in cm? Length, ...	Object towel1 has the following size:\nLength:...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...
4	What is the size of the roller2_blind in cm? L...	Object roller2_blind has the following size:\n...	/home/rodionfa/3d_box_data/ml-hypersim/data/ai...

Figure 19. Question Answer (sizes) Dataset



Question: What is the size of the *coach* in cm? Length, width, and height.

Ground Truth: Object *coach* has the following size:
length: 162.84 cm, width: 83.69 cm, height: 85.5 cm

Moondream: Object *coach* has the following size:
length: 126.17 cm, width: 73.56 cm, height: 82.69 cm

Figure 20. Image example

Moondream already performed well with these types of questions, and fine-tuning didn't seem to improve its performance. However, the model appeared to be memorizing all possible sizes rather than understanding or deriving them.
Despite its proficiency in answering, Moondream cannot generate more complex tasks, such as simple layouts. This inverse task proved to be particularly challenging for the model, and as a result, I was unable to obtain meaningful outcomes.

Fine-Tuning Vision Models - Part 2

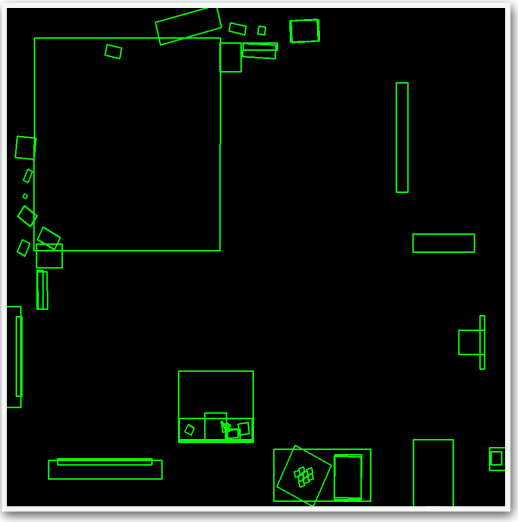
I decided to experiment with LLaMA, a more powerful vision model with enhanced capabilities. Here are the models I tested:

1. **Meta LLaMA-3.2-11B-Vision**: With 11 billion parameters and built-in vision instructions, LLaMA is a highly versatile model for fine-tuning. It provides strong responses, is freely accessible, and is relatively straightforward to fine-tune, making it a good option for this task.

2. **Microsoft Phi 3.5**: This model supports multi-image input but did not perform as well as LLaMA in terms of output quality and overall results.
3. **ChatGPT 4.0**: The API-based version of ChatGPT 4.0 supports generating responses from multiple images and can be fine-tuned. However, it is a paid service.
4. **LLaVa (from the original LLaVa paper [10])**: While this model offers decent outputs, its performance is slightly lower in quality compared to the other models mentioned above.

Fine-Tuning Process

I attempted to fine-tune **LLaMA-3.2-11B-Visual-Instruct** using a single NVIDIA A100 GPU with 80GB of memory, focusing specifically on the query and value layers through **LoRA (Low-Rank Adaptation)**. To streamline the fine-tuning process, I utilized the **PEFT (Parameter-Efficient Fine-Tuning)** and **TRL (Transformers for Reinforcement Learning)** libraries, which allowed for efficient and flexible model adaptation.




```
{
  "furniture_list": {

    "duck": {
      "left_bot_corner": [
        17.15569923648713,
        151.0821522573944
      ],
      "top_left_corner": [
        11.636712048088743,
        158.87425406141867
      ],
      "top_right_corner": [
        22.399569984576324,
        166.49737393389137
      ],
      "bottom_right_corner": [
        27.918557437917354,
        158.7052721298671
      ]
    },
...
```


Figure 21. Example of pair: image - description

For fine-tuning, I used input pairs consisting of layout images and their corresponding JSON descriptions, including the furniture's locations and sizes. However, I was only able to collect 80 pairs due to the prevalence of non-standard designs, which made data collection more difficult. Unfortunately, the fine-tuning process did not yield any notable improvements over the pre-trained model.

Metrics

	FID score	KL div.	OOB error	BBL	Nobj	GPT-4V	K	Classes
LLaMa-FT-3.2-V	58,45	0,12	35,2	5,04	8,3	-	2	B, LR, K, Bath

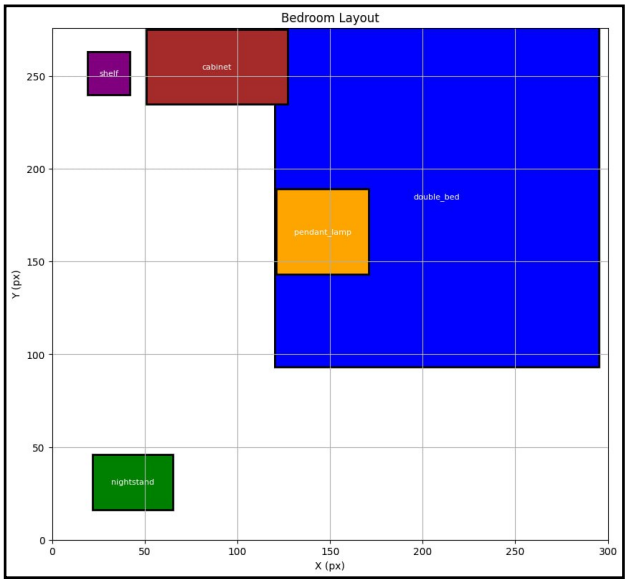
The output strictly depends on a prompt and resembles LayoutGPT.

Possible reasons how to improve the result

- **Layout Generation**: This inverse problem could benefit from additional constraints or specialized loss functions.
- **Loss Function Enhancement**: Currently using simple language modeling loss. Future improvements could include adding a loss to compare discrepancies between numbers (sizes) for more accurate predictions

- **Dataset Expansion**: The current dataset lacks mesh data, but Infinigen offers control over and variation of scenes, providing an opportunity to expand the dataset for better training.

Example output, generated with the same prompt as LayoutGPT.



The code

All the code regarding experiments was uploaded to GitHub - https://github.com/OldDeLorean/CS_326_DL.

References:

[1] *Generative Layout Modeling using Constraint Graphs*, Wamiq Para et al.
[2] *LayoutGPT: Compositional Visual Planning and Generation with Large Language Models*, Weixi Feng et al.
[3] *I-Design: Personalized LLM Interior Designer*, Ata Çelen et al.
[4] *Holodeck: Language Guided Generation of 3D Embodied AI Environments*, Yue Yang et al.
[5] *Evaluating Spatial Understanding of Large Language Models*, Yutaro Yamada et al.
[6] Moondrem
[7] *Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding,* Mike Roberts et al.
[8] *Infinite Photorealistic Worlds using Procedural Generation,* A. Raistrick et al.
[9] *The Llama 3 Herd of Models*, Meta AI
[10] *Visual Instruction Tuning,* Haotian Liu et al