Project Report: Language-Guided Object Placement in 3D Indoor Scenes Using Vision Language Models

Jialu Gao, Yifei Liu, Grace Su, Haochen Zhang
Carnegie Mellon University

1 Introduction

Figure 1: Ikea's Kreativ tool for visualizing furniture in a room [1]

The goal of this project is to develop a pipeline where, given an arbitrary object in 2D or 3D format and an existing indoor 3D scene, a user can customize scene augmentation by specifying with natural language where they want the object to be placed within the scene. The specification from the user is in the form of a natural language statement describing the desired spatial position of the object to be placed. While scene customization and generation in the 2D space is relatively more explored, there are fewer works addressing customization of 3D scenes. Being able to edit 3D scenes however, has many applications in the real world, such as viewing how an item or piece of furniture looks in your room, testing out different layouts for interior design, or generating augmented 3D scene data for learning other robotics tasks such as indoor navigation or manipulation. While apps and tools like Ikea Kreativ [1] (Fig. 1), Crate & Barrel's View-In-My-Room [2], and Amazon's AR View exist [3] and partially inspire this idea, these applications are not open-sourced, do not use language to specify placement, and do not allow for users to upload their own images. In real-world application areas in furniture viewing and beyond, making such a pipeline as flexible and customizable as possible is important, as well as only requiring input information easy to obtain from everyday human users. Thus, the goal of the pipeline is to be able to take in 2D images of any arbitrary household object and any single-room 3D scene.

To achieve this goal, we first start with a collection of existing 3D household assets and 3D indoor scenes. We then develop a training-free pipeline that leverages vision-language models (VLMs) and foundation models to first place and localize the object in a 2D rendering of the scene, then translate this to 3D using relative positioning techniques. We qualitatively demonstrate the scene editing results of our method across diverse language instructions, scenes, and objects, while also evaluating the final edited scene layout quantitatively using state-of-the-art metrics. We find that our method allows for natural and consistent scene layout edits, while accurately following spatial language instructions containing relative references to pre-existing objects in the scene. The effectiveness of the method at zero-shot scene

editing given a user instruction is promising in both it's generalizability and usability for future applications in online shopping, interior design, and scene generation.

2 Related Work

A highly related research area to our proposed problem is **3D Indoor Scene Design**. With the recent advances of Diffusion Models [4, 5] in generative tasks, Tang et al. [6] propose DiffuScene which parameterize a 3D scene using {Location, Size, Orientation, Class, Shape code} of objects within the scene, and train a diffusion model to generate 3D scenes from text. Similarly, Fang et al. [7] propose Ctrl-room, which only trains a diffusion model to generate 3D layouts, and uses Control-net [8] to generate panoramas conditioned on the layout. In InstructScene, Lin and Mu [9] adopt a different parametrization approach using semantic graph, and trains diffusion models for semantic graph prior and layout decoder. In a different approach, previous studies have also explored using Large Language Models (LLMs) and Vision-Language Models (VLMs) for scene design and generation. Most recently, LayoutGPT [10] uses exemplar data retrieved according to user input to perform in-context learning, which guides GPT to produce desirable layouts in both 2D and 3D space. LLPlace [11] further builds on the inherent conversational capabilities of LLMs [12] and trains a LoRA [13] on dialogue-based training data to achieve iterative 3D scene generation and editing. LayoutVLM [14] prompts VLMs [15, 16] with language instructions, asset groups, and scenes with visual marks to generate code representation of a scene, which can further be optimized to output the final object placements.

The main difference between our work and the 3D Indoor Scene Design domain is that we want to allow customized objects and scene inputs. Many of the existing methods only operates on an empty scene (LayoutVLM), or only support object retrievals from a specified dataset (LLplace, DiffuScene, InstructScene), or can only perform layout-to-scene generation without supporting conditional generation given certain objects (Ctrl-room). The challenge of our task lies in how to unify the scales of the given object and scene, and ground 3D spatial placement given natural language input. The only existing work that can support conditional generation based on scene and object input is LayoutGPT, but since it doesn't explicitly align the scale of different inputs, it will likely fail when faced diverse customized inputs.

3 Method

The key idea this project will explore is leveraging the prior spatial and object-relevant knowledge of a pre-trained text-to-image model to insert an arbitrary 3D object into a realistic location within an existing indoor 3D scene, based on a natural language instruction.

Most existing works on 3D indoor scene layout generation and editing rely on fine-tuning a LLM to generate layout representations that respect object bounding boxes. Thus, this project evaluates the effectiveness of a training-free 3D scene layout editing approach that utilizes existing foundation models. The overall pipeline of our method is shown in Figure 2 and each component is further described below.

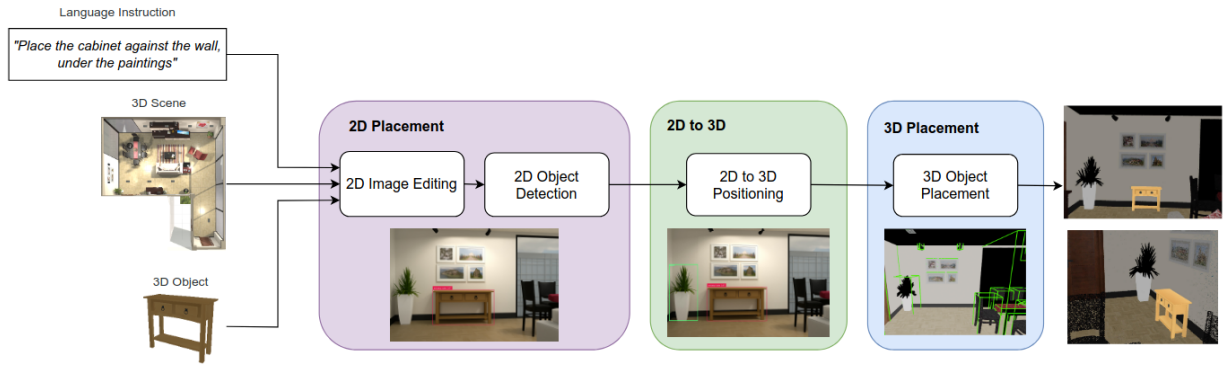


Figure 2: Overall diagram of the proposed pipeline

3.1 2D Image Editing & Object Detection

The first step in our pipeline is to render the scene with the object of interest and obtain a 2D bounding box to place the object. We experiment with two ways of rendering the 2D image and two corresponding methods to generate the bounding box.

Learned diffusion prior. We adopt SDEdit [17] and the Flux-dev [18] text-to-image diffusion model to edit a 2D image of the indoor scene by inserting an object based on a natural language instruction. For instance, for the instruction "place the cabinet against the wall under the paintings", the prompt given to the text-to-image diffusion model would include the original BLIP [19] caption combined with the instruction: "A photo of a white wall with a cabinet against the wall under the paintings." This enables the diffusion model to interpret the instruction and generate a plausible object location. While this method does not support inserting a specific object instance into the image, it leverages learned priors over object categories to infer reasonable placements and sizes. Future work could involve extracting attention maps from customizable text-to-image models to insert the target object while preserving its identity.

To extract the object location, we first apply DDIM inversion to obtain the inverted noise map corresponding to the generated scene image. We then extract the diffusion model's cross-attention maps corresponding to the target object's text description [20, 21, 22]. A 2D bounding box is derived from the aggregated maps.

Vision Language Models. To overcome SDEdit's limitation of single-image editing, we leverage the recent advances in image generation from vision language model that allows multiple image inputs. In particular, we use GPT-image-1 model [23] from OpenAI. We provide both the background scene and the object image as input, along with a prompt such as "Inpaint the object in the second image into the first image (background scene) without changing the background scene. Place it in front of the curtain." Initial experiments were conducted using the web interface, followed by automated image generation and saving pipeline with API calls.

To localize the inserted object in the output image, we apply the foundation model Grounding DINO 1.5 [24], which detects the object using the generated image and a corresponding text query. The model returns the 2D bounding box of the inserted object in pixel coordinates.

3.2 2D to 3D Positioning

The second step is to convert the 2D bounding box location into 3D coordinates. To achieve this, we identify existing anchor objects in the scene and use them as references. Since the scenes already include 3D bounding boxes for these anchor objects, we can leverage this information to map between 2D and 3D coordinates. For instance, given the prompt "Place the table next to the potted plant," we would select the potted plant as our anchor object. We apply the semantic segmentation model Grounding DINO 1.5 [24] to the edited 2D image to obtain a bounding box for the anchor object. Then, we calculate the relative position of the inserted object's bounding box with respect to the anchor object's bounding box to determine the x and y coordinates in the camera frame. Since 2D images do not contain depth (z-axis) information, and considering typical object placement should align, we assign the inserted object the same z-coordinate as the anchor object. Ultimately, this procedure transforms the 2D bounding box data into corresponding 3D coordinates within the point cloud space.

3.3 3D Object Placement

Once the location for object placement is determined in 3D, the object point cloud can be merged with the scene point cloud at that location and rendered to obtain the final scene.

The proposed method has an additional advantage of ensuring the final insertion of the object is scaled and position appropriately, relative to the input 3D scene, because the pre-trained text-to-image model has knowledge of how to generate realistic images of objects within indoor scenes. Meanwhile, existing works on 3D indoor scene layout generation and editing assume the 3D object is already scaled appropriately relative to the 3D scene.

4 Dataset and Evaluation

We use the Amazon Berkeley Objects (ABO) dataset [25] as a primary resource for objects to be placed, in 2D or 3D formats. Custom scenes from the Unity game engine [26] serve as the foundational dataset for 3D indoor scenes into which users can place objects, due to the customizability of the scenes and ease of rendering 2D images automatically.

For qualitative assessment, we visually inspect the placement of objects in the scene and whether it aligns with real-world physics. To evaluate our method quantitatively, we employ metrics presented by other works, such as Fréchet Inception Distance (FID) scores and Object Overlap Rate. We plan to explore additional object placement metrics with semantic segmentation. We can also query a multi-modal large vision language model (VLM) to score the placement of objects or conduct human studies to score the placement.

4.1 Datasets

The **ABO** dataset is a collection of Amazon products with product metadata, catalog images, and 3D models. It contains 147,702 products representing common household objects such as furniture, appliances, and clothing (fig. 3). For approximately 8000 of these products, the dataset provides 72-frame 360-degree image sequences and high-quality 3D models. Each 3D model includes multiple modalities (fig. 4), such as rendered images for 91 different viewpoint, camera parameters, object segmentation, dense normals, depth maps, and texture maps.

Figure 3: ABO Dataset: Various products with 3D models.

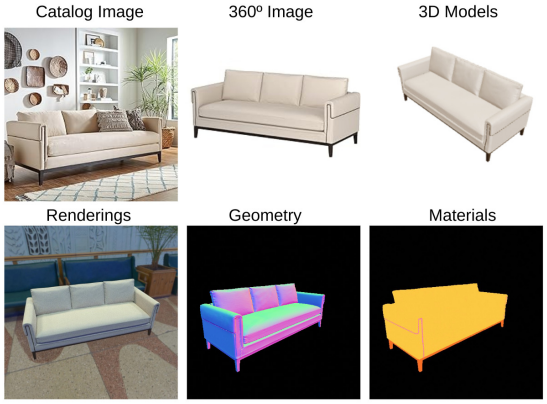


Figure 4: ABO Dataset: Modalities for each 3D model.

For the 3D indoor scenes in which to place objects, custom indoor scenes from the **Unity** game engine are used, which primarily feature environments such as bedrooms, living rooms, and offices (fig. 5), as well as multi-room homes. Each scene includes 3D surface mesh reconstructions, point clouds, and the object instance-level segmentations.

The **3D-FRONT** (3D Furnished Rooms with layOuts and semaNTics) dataset provides a large collection of synthetic indoor scenes with detailed room layouts, semantic annotations, and textured 3D furniture objects. It serves as an additional resource for testing object placement within diverse, structured environments.

4.2 Metrics

We calculate CLIP [27] text-image alignment scores to evaluate how well the images of the generated layouts follow the input text instructions. The CLIP score is the cosine similarity between the text and image's CLIP embeddings, with a score close to 1 indicating that the image matches the text description very closely.

We also leverage the VLM GPT-4o [28] to assess layout realism, basing our evaluation prompt template on prior work [11]:

"Give a grade from 0 to 10 to the following room renders based on how well they correspond together to the user instruction (in triple backquotes) in the following aspects:

- Adherence to real world physics

Figure 5: Unity Dataset: Sample scene

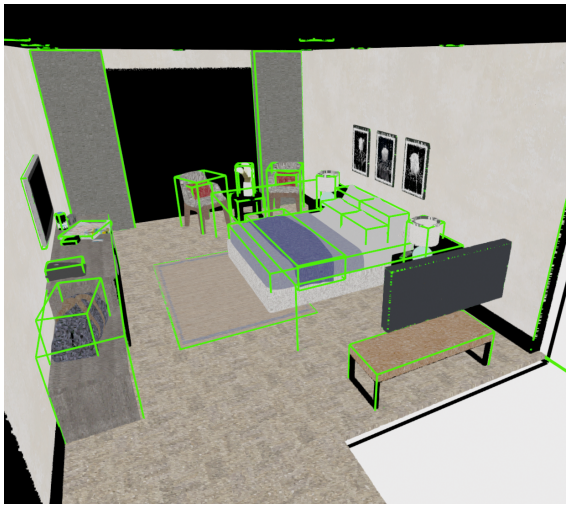


Figure 6: Unity Dataset: Point cloud with object detections

- Functionality and Activity-based Alignment
- Layout and furniture
- The user instruction: [User instruction]

Return the results in the following JSON format: {"adherence": 4, "functionality": 8, "layout": 7, "instruction": 7}"

5 Results

We evaluate our 3D scene layout editing method using 20 different user instructions for indoor scene editing. In Sec 5.1, we report the qualitative evaluation results, and in Sec 5.2, we report the quantitative metrics.

5.1 Qualitative Results

The first approach we tried was using a learned diffusion prior for the 2D image editing and placement step. We found that applying SDEdit to a pre-trained text-to-image model generally struggled to insert new objects into the input image without modifying existing objects. In addition, because the text-to-image model we used does not accept multiple images as input, it inserts objects with arbitrary appearances. We show an example editing result in Figure 7.

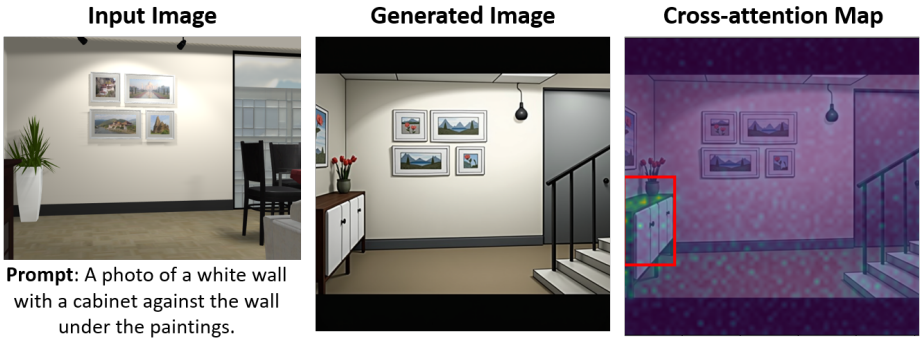


Figure 7: Example result from learned diffusion prior. The cross-attention map corresponds to the text token for the inserted object, "cabinet". The bounding box is calculated from the positions of peak values in the cross-attention map.

To effectively utilize multiple input images, we instead employed pre-trained vision-language models to insert the object into a given scene within the 2D space. Examples of edited scenes are presented in Table 1. The results indicate that although VLMs may modify the details of the background or the inserted object, the overall structure of the scene is preserved, and the placement of the object remains contextually meaningful.

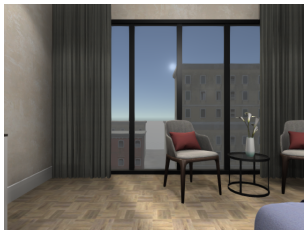
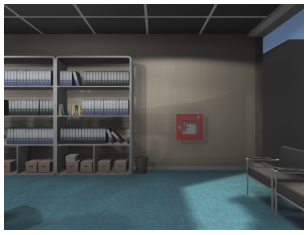
Prompt: Inpaint the object in the second image into the first image (background scene) without changing the background scene, place it in front of the curtain.	**Prompt:** Inpaint the object in the second image into the first image (background scene), place it against the wall under the paintaings.	**Prompt:** Inpaint the object in the second image into the first image (background scene), place it next to the door under the paintings.	**Prompt:** Inpaint the object in the second image into the first image (background scene), place it next to the table.
			
			
			

Table 1: Top row: text prompt to VLM, second row: input scene image, third row: object to be placed, fourth row: generated image with object localized with GroundingDINO. All prompts are followed by: *Do not modify the background scene, keep it original aspect ratio.*

Using the edited 2D scene images, our method infers the 3D location of the inserted object. Then it merges the point clouds of the inserted object and the original scene according to this computed 3D offset. We present the qualitative results illustrating the effectiveness of our semantic object insertion pipeline in Figure 8 with a portion of the evaluated user instructions. Each example contains the natural-language instruction describing where to insert the object in the scene, the initial state of the scenes, an isolated view of the object intended for insertion, and multiple views of each 3D scene after the object insertion.

From Figure 8, we can see that the objects range widely in type, including furniture such as tables, chairs, armchairs, cabinets, ladders, and office chairs. The diverse set of objects tests our method's versatility in handling various object geometries and semantic categories. From the output scenes, several observations can be made:

- Objects are inserted accurately following the textual prompts, correctly interpreting spatial instructions such as "under," "next to," "in front of," and "on the right of."
- The inserted objects align naturally with the original scene context, respecting environmental structures such as walls, doors, furniture placements, and room layout.
- Even in more challenging cases (e.g., placing objects near curtains or close to existing furniture), the method maintains coherent placement, showing robustness against potential overlaps or misplacements.

The consistent preservation of scene structure and logical placement demonstrates the effectiveness of the semantic and geometric reasoning employed by our pipeline.

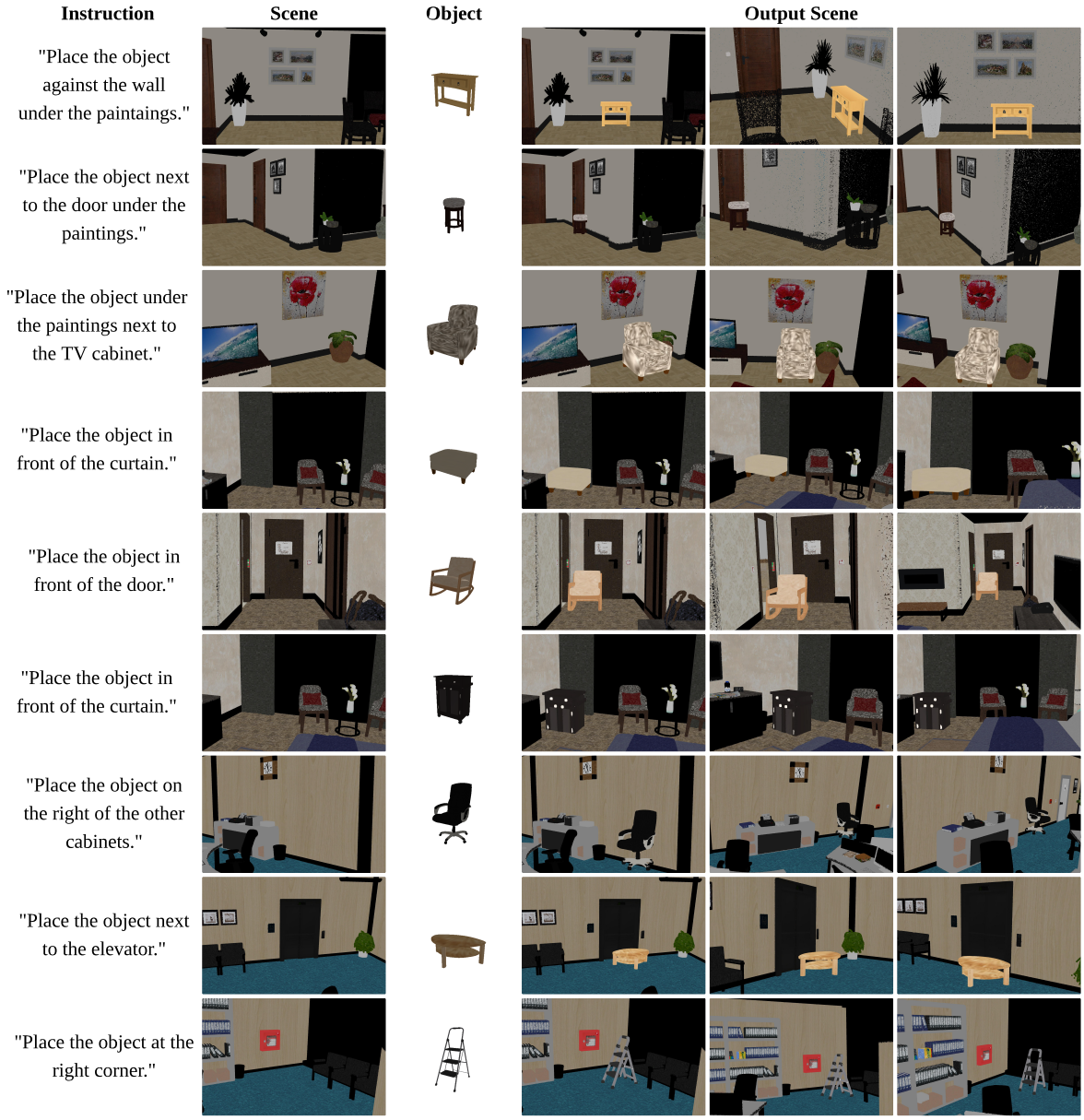


Figure 8: Qualitative results of our pipeline. Our method can semantically insert diverse objects into the scene according to the text instructions.

5.2 Quantitative Results

To calculate quantitative metrics, we generate edited scene layouts for 20 different user instructions, then render 3 views per layout to create a total of 60 images.

We find that the average CLIP text-image alignment score is 0.2335, compared to an average score of 0.2155 for the alignment between the un-edited input scenes and the target prompts. This indicates that our scene editing approach helps improve the adherence of the 3D scene to the user's natural language instruction.

In addition, we use GPT-4o to evaluate layout realism, as described in Section 4.2, and compute average scores in Table 2. The average Instruction score is 9.55 out of 10, demonstrating that our method closely follows the user's editing instructions. The other layout realism scores are near 6, indicating that there are still some prompts for which the inserted object's placement could be improved.

Metric	Adherence	Functionality	Layout	Instruction
Mean	6.70	6.35	6.50	9.55

Table 2: GPT-4o Evaluation Results. Each score is on a scale of 0 to 10.

6 Conclusion

In conclusion, we demonstrated the effectiveness of a training-free 3D scene layout editing approach by using VLMs such as GPT-image-1 to edit 2D images first, then using 2D object detection and relative positioning to position the target object into the 3D scene. We show that this works across a diverse set of object types and different scenes, with natural language instructions that contains spatial language references.

Despite the promising results of our pipeline, we acknowledge that the method has limitations as well. During our experiments, we observed several limitations to the VLM approach for 2D image editing. After a few consecutive prompts, the model often fails to preserve the original aspect ratio of the scene image, instead generating square or vertical rectangle images. This occurs despite explicitly including instructions such as "Do not modify the background scene, keep it original aspect ratio." We found that clearing the prompt history or restarting the API session can mitigate this issue, suggesting that prompt accumulation may lead to context drift or hidden state contamination.

Another failure in preserving aspect ratio occurs when attempting to insert large objects into spatially constrained scenes. In such cases, the model tends to stretch or distort the background in an effort to accommodate the object, leading to inconsistent output shape and layout. The scale of the final inserted 3D object may then be inconsistent with the scale in the 2D rendering. An area of future work to mitigate this would be to integrate geometric reasoning or layout constraints into the VLM's inference process, either by combining VLMs with 3D priors or by using auxiliary models to validate spatial consistency.

References

[1] I. Group, "Ikea launches new ai-powered experience; ikea kreativ," https://www.ingka.com/newsroom/ikea-launches-new-ai-powered-experience-empowering-customers-to-create-lifelike-room-designs/, 2022, [Accessed 26-02-2025].

[2] C. . Barrel, "View in your room using augmented reality," https://www.crateandbarrel.com/special-features/ar-living-room-furniture/1, [Accessed 26-02-2025].

[3] Amazon, "Amazon ar view," https://www.amazon.com/products, 2020, [Accessed 26-02-2025].

[4] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[5] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022. [Online]. Available: https://arxiv.org/abs/2010.02502

[6] J. Tang, Y. Nie, L. Markhasin, A. Dai, J. Thies, and M. Nießner, "Diffuscene: Denoising diffusion models for generative indoor scene synthesis," 2024. [Online]. Available: https://arxiv.org/abs/2303.14207

[7] C. Fang, Y. Dong, K. Luo, X. Hu, R. Shrestha, and P. Tan, "Ctrl-room: Controllable text-to-3d room meshes generation with layout constraints," 2024. [Online]. Available: https://arxiv.org/abs/2310.03602

[8] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2302.05543

[9] C. Lin and Y. Mu, "Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior," 2024. [Online]. Available: https://arxiv.org/abs/2402.04717

[10] W. Feng, W. Zhu, T. jui Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang, "Layoutgpt: Compositional visual planning and generation with large language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.15393

[11] Y. Yang, J. Lu, Z. Zhao, Z. Luo, J. J. Yu, V. Sanchez, and F. Zheng, "Llplace: The 3d indoor scene layout generation and editing via large language model," *ArXiv*, vol. abs/2406.03866, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270286118

[12] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[13] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[14] F.-Y. Sun, W. Liu, S. Gu, D. Lim, G. Bhat, F. Tombari, M. Li, N. Haber, and J. Wu, "Layoutvlm: Differentiable optimization of 3d layout via vision-language models," 2024. [Online]. Available: https://arxiv.org/abs/2412.02193

[15] OpenAI, "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[16] F. Li, R. Zhang, H. Zhang, Y. Zhang, B. Li, W. Li, Z. Ma, and C. Li, "Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models," 2024. [Online]. Available: https://arxiv.org/abs/2407.07895

[17] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "SDEdit: Guided image synthesis and editing with stochastic differential equations," in *International Conference on Learning Representations*, 2022.

[18] B. F. Labs, "Flux," https://github.com/black-forest-labs/flux, 2024.

[19] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*. PMLR, 2022, pp. 12 888–12 900.

[20] Y. Kim, J. Lee, J.-H. Kim, J.-W. Ha, and J.-Y. Zhu, "Dense text-to-image generation with attention modulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7701–7711.

[21] S. Ge, T. Park, J.-Y. Zhu, and J.-B. Huang, "Expressive text-to-image generation with rich text," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7545–7556.

[22] G. Parmar, K. Kumar Singh, R. Zhang, Y. Li, J. Lu, and J.-Y. Zhu, "Zero-shot image-to-image translation," in *ACM SIGGRAPH 2023 conference proceedings*, 2023, pp. 1–11.

[23] "Introducing our latest image generation model in the api," Apr 2025. [Online]. Available: https://openai.com/index/image-generation-api

[24] T. Ren, Q. Jiang, S. Liu, Z. Zeng, W. Liu, H. Gao, H. Huang, Z. Ma, X. Jiang, Y. Chen, Y. Xiong, H. Zhang, F. Li, P. Tang, K. Yu, and L. Zhang, "Grounding dino 1.5: Advance the "edge" of open-set object detection," 2024.

[25] J. Collins, S. Goel, A. Luthra, L. L. Xu, K. Deng, X. Zhang, T. F. Y. Vicente, H. Arora, T. L. Dideriksen, M. Guillaumin, and J. Malik, "Abo: Dataset and benchmarks for real-world 3d object understanding," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21 094–21 104, 2021.

[26] Unity Technologies, "Unity," 2023, game development platform. [Online]. Available: https://unity.com/

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[28] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.