ARCHITECT: Generating Vivid and Interactive 3D Scenes with Hierarchical 2D Inpainting

Yian Wang[*]
Umass Amherst
yianwang@umass.edu

Xiaowen Qiu[*]
Umass Amherst
xiaowenqiu@umass.edu

Jiageng Liu[*]
Umass Amherst
jiagengliu@umass.edu

Zhehuan Chen
Umass Amherst
zhehuanchen@umass.edu

Jiting Cai
Shanghai Jiao Tong University
caijiting@sjtu.edu.cn

Yufei Wang
Carnegie Mellon University
yufeiw2@andrew.cmu.edu

Tsun-Hsuan Wang
MIT
tsunw@mit.edu

Zhou Xian
Carnegie Mellon University
zhouxian@cmu.edu

Chuang Gan
Umass Amherst
chuanggan@umass.edu

Abstract

Creating large-scale interactive 3D environments is essential for the development of Robotics and Embodied AI research. However, generating diverse embodied environments with realistic detail and considerable complexity remains a significant challenge. Current methods, including manual design, procedural generation, diffusion-based scene generation, and large language model (LLM) guided scene design, are hindered by limitations such as excessive human effort, reliance on predefined rules or training datasets, and limited 3D spatial reasoning ability. Since pre-trained 2D image generative models better capture scene and object configuration than LLMs, we address these challenges by introducing ARCHITECT, a generative framework that creates complex and realistic 3D embodied environments leveraging diffusion-based 2D image inpainting. In detail, we utilize foundation visual perception models to obtain each generated object from the image and leverage pre-trained depth estimation models to lift the generated 2D image to 3D space. While there are still challenges that the camera parameters and scale of depth are still absent in the generated image, we address those problems by "controlling" the diffusion model by *hierarchical inpainting*. Specifically, having access to ground-truth depth and camera parameters in simulation, we first render a photo-realistic image of only back-grounds in it. Then, we inpaint the foreground in this image, passing the geometric cues in the back-ground to the inpainting model, which informs the camera parameters. This process effectively controls the camera parameters and depth scale for the generated image, facilitating the back-projection from 2D image to 3D point clouds. Our pipeline is further extended to a hierarchical and iterative inpainting process to continuously generate placement of large furniture and small objects to enrich the scene. This iterative structure brings the flexibility for our method to generate or refine scenes from various starting points, such as text, floor plans, or pre-arranged environments. Experimental results demonstrate that ARCHITECT outperforms existing methods in producing realistic and complex environments, making it highly suitable for Embodied AI and robotics applications.[2]

[*] Equal Contribution

[2] Project page: https://vis-www.cs.umass.edu/ARCHITECT

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

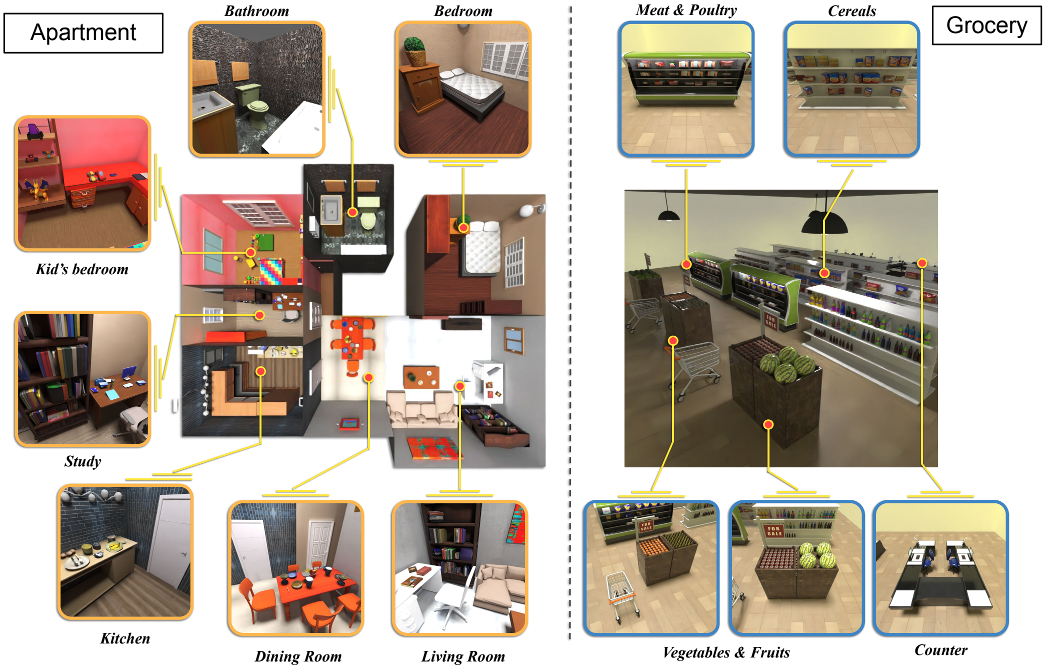


Figure 1: We present ARCHITECT, a generative framework to create *diverse*, *realistic*, and *complex* Embodied AI scenes. Leveraging 2D diffusion models, ARCHITECT generates scenarios in an open-vocabulary manner. Here, we showcase two cases in detail: an apartment and a grocery store.

Collecting or generating large-scale training data has recently emerged as a promising direction for advancing Robotics and Embodied AI research. A major focus in recent works pursuing this direction advocates for data generation in simulated environments [Wang et al., 2023b,a, Ha et al., 2023, Dalal et al., 2023], as simulation offers a cost-effective approach to data collection that scales naturally with computational resources; this thrust holds the potential for producing realistic physics and rendering data, and meanwhile grants access to valuable ground-truth state information for speeding up policy learning. Among the types of data needed for training Embodied AI agents, diverse and realistic *environments* with the possibility of interacting with surrounding entities is crucial. However, obtaining vivid interactive scene and environment data remains challenging. Recent studies have attempted to tackle this problem by developing generative models for environment creation via various approaches, including procedural generation with predefined rules [Deitke et al., 2022], diffusion-based scene generation [Tang et al., 2023a, Yang et al., 2024b, Feng et al., 2024], and large language model (LLM) guided scene population and design [Wang et al., 2023b, Yang et al., 2024c, Wen et al., 2023].

Despite these recent efforts, generating *diverse*, *realistic*, and *complex* Embodied AI environments still remains a challenging problem due to the inherent drawbacks and assumptions made in the pipeline designs of existing methods. For example, manually designed environment datasets [Ramakrishnan et al., 2021, Weihs et al., 2021, Li et al., 2023a, Fu et al., 2020a,b] require excessive human effort and are hence inherently difficult to scale. Procedural generation methods [Deitke et al., 2022, Khalifa et al., 2020, Earle et al., 2021, Zhao et al., 2021] rely on predefined rules, which are limited in their ability to learn from and resemble real-world distributions, and struggle to generate open-vocabulary scenes. Large language model (LLM) guided scene generation process [Wang et al., 2023b, Yang et al., 2024c, Wen et al., 2023, Lin et al., 2023, Aguina-Kang et al., 2024, Feng et al., 2024] also presents its own limitations, as LLMs operate in language space and have limited 3D understanding and spatial reasoning capabilities. Moreover, existing LLM-based scene generation methods still rely on certain simplifications and hand-designed rules, such as primarily focusing on placements of large furniture pieces on the floor or against walls, and only considering simple inter-object relationships such as random placement of small items *on top of* large background furniture. Therefore, these methods struggle to generate more complex and cluttered object arrangements that are often encountered in daily life, such as “an organized dining table”, “an office desk drawer cluttered with objects”, or “a shelf of toys”, which typically require nuanced object placement and context-aware positioning.

As a result, there still exists a gap in current literature for generating interactive 3D scene with detailed and complex configurations that closely resemble real-world distributions. To this end, we propose ARCHITECT, a generative framework for creating realistic and interactable 3D scenes via diffusion-based 2D inpainting [Podell et al., 2023]. Our pipeline leverages controllable and hierarchical generation in 2D image space. Compared to LLMs which operate in language space, pre-trained image-based generative models are able to better capture scene and object configurations from massive image data readily available, both at the scene level and in fine-grained inter-object spatial information. Pre-trained depth estimation models [Ke et al., 2024, Bhat et al., 2023, Yang et al., 2024a] can then be used to lift the generated 2D static image to 3D environments. However, images created from 2D generative models do not provide accurate camera parameters, which are crucial for reconstructing accurate 3D environments. In addition, the predicted depth images also present scale ambiguity. To address these challenges, we propose to “control” the 2D generative models with 3D constraints via *hierachical inpainting*. First, we render a photo-realistic image in a simulated empty scene with only a static background, where we have access to ground-truth depth and camera parameters. We then use this image as a *template* for inpainting the foreground using 2D diffusion models. During this process, the generation respects the camera parameters informed by the geometric cues in the background image and ensures that the inpainted components are both semantically and spatially consistent with existing components in the input image. By generating images this way, we effectively control the camera parameters and depth scale for the generated image, which allows us to project it back to 3D point clouds. Subsequently, utilizing visual recognition models [Kirillov et al., 2023, Liu et al., 2023, Ren et al., 2024], we segment the 2D image to obtain the semantics and geometric information of each generated object. These objects are then instantiated in the actual simulated environments, by either retrieving from large-scale asset databases [Deitke et al., 2023a, Mo et al., 2019] or generating using image-to-3D generative models [Xu et al., 2024].

While the pipeline described above is able to generate 3D scene configurations described from a single camera view, our goal is to generate complete scenes observable from *multiple* views. In addition, we aim to generate scenes with real-world complexity, where objects of different scales together form a holistic environment (e.g. ideally we want to also generate small items placed on a shelf or in a drawer). Therefore, we further extend the pipeline to perform iterative and hierarchical inpainting, during which we continuously render new image patches of different locations of the scene to further enhance the complexity when needed. Specifically, given a text description of a target scene, we (i) generate the floor plan following previous works [Yang et al., 2024c, Wen et al., 2023], (ii) add background assets such as walls and floors into a simulated environment, (iii) render images of this empty scene and perform the aforementioned, proposed iterative inpainting process for scene-level generation from multiple camera views, (iv) hierarchically, apply inpainting again at a finer level to place small objects in various semantically plausible locations in the interior space, and (v) finally resulting in a complex 3D scene. Note that such iterative process results in a flexible generative pipeline that can handle different levels of inputs: text descriptions, floor plans, or even pre-arranged scenes.

Our pipeline is able to generate complex scenes that are fully interactable, with detailed asset placement and configurations at multiple scales, as shown in Figure 1. Note that since we make use of powerful prior knowledge encoded in 2D pre-trained generative models, we are able to generate open-vocabulary scenes in a zero-shot manner, for not only diverse room types in home settings, but also non-home environments such as grocery stores. Our experiments show that our framework outperforms prior scene creation approaches in generating interactable scenes that are more complex and realistic. We summarize our main contributions as follows:

- We introduce ARCHITECT, a zero-shot generative pipeline that creates diverse, complex, and realistic 3D interactive scenes to advance Embodied AI agents and Robotics research.
- We propose to leverage 2D prior from vision generative models to facilitate the 3D interactive scene generation process, and make such process *controllable* by initializing from simulation-rendered image for hierarchical inpainting, ensuring consistent spatial features and controllable camera parameters and depth scale, allowing accurate 2D to 3D lifting.
- The experimental results show that our method outperforms previous approaches in generating more complex and realistic interactive 3D scenes, both quantitatively and qualitatively.

Methods	No Train	No Human Effort	Interactive	Organized Small Objects	Open Vocab
Behavior-1k			✓	✓	
ProcTHOR	✓		✓		
Holodeck	✓	✓	✓		✓
AnyHome	✓	✓			✓
RoboGen	✓	✓	✓		✓
PhyScene		✓	✓		
DiffuScene		✓			
LayoutGPT		✓			
Text2Room	✓	✓			✓
Ours	✓	✓	✓	✓	✓

Table 1: We compare our work with previos works that also aims to generate large scale 3D scenes in 5 aspect. Here, **No Train** means no need for training data of indoor layouts.

Our code will be made publicly available.

2 Related Works

Indoor Scene Generation A large body of works have focused on automatic indoor scene generation. Some works generate only the static mesh of the scene [Höllein et al., 2023, Schult et al., 2023]; in contrast, ours generates interactive scenes that can be used for downstream embodied AI and robotics tasks. One line of research generates interactive indoor scenes via procedure generation with manually defined rules [Deitke et al., 2022]. The quality and diversity of the generated scenes highly depend on the predefined rules, which demands huge human efforts. Another group of works train a generative model (e.g., transformers or diffusion models) on in-door scene datasets [Fu et al., 2021] and use the trained models for scene generation [Yang et al., 2024b, Feng et al., 2024, Tang et al., 2023a, Paschalidou et al., 2021], and the quality and diversity of the scenes are bounded by the training dataset. Ours differ from these two lines of work as we do not use any manually defined rules nor pre-collected datasets, which might constrain the diversity of the generated scenes. Instead, we achieve higher diversity in the generated scenes by leveraging 2D image generative models that are trained with abundant internet data, which cover a wider distribution of scenes than those generated by manually defined rules or a fixed dataset. Recently, several works employ a Large Language Model (LLM) for indoor scene generation [Wang et al., 2023b, Yang et al., 2024c, Wen et al., 2023], such as floor plan, layout, and object placements. Since the 3D spatial reasoning abilities of LLMs are still limited, the quality and diversity of the generated scenes are still bounded. By combining 2D image generative models, simulation rendering and controlled image impainting, our method achieves more coherent 3D layouts and higher scene diversity. We make a comparison between us and previous works in Table 2 .

Text-to-Image Diffusion models Text-to-image models based on diffusion model, such as DALL-E2 [Ramesh et al., 2021] and LDM [Rombach et al., 2021], have become dominant in text-to-image generation. These text-to-image diffusion models have been trained on billions of images, giving them strong visual and 3D priors in addition to their image generation capabilities. Through SDS (Score Distillation) loss proposed by DreamFusion [Poole et al., 2022], the 3D prior can be leveraged for a variety of downstream tasks like 3D object generation [Poole et al., 2022, Tang et al., 2023b, Wang et al., 2023c, Liang et al., 2023]. In our observations, in addition to visual and 3D priors, these text-to-image diffusion models also have strong priors for the layout of furniture and objects in a room. Therefore, we designed a pipeline to exploit these layout priors from inpainting diffusion models to generate realistic indoor scenes.

3 Method

We automatically generate complex and realistic embodied environment scenes through a hierarchical and iterative process of rendering, inpainting, and visual reasoning. Specifically, as shown in Figure 2, starting from an empty room, our pipeline first iteratively generates the layout of large furniture items (Figure 2 *Large Furniture*). Subsequently, we place smaller objects inside or upon these large furniture pieces, as depicted in Figure 2 *Small Objects*. In detail, our process entails the following

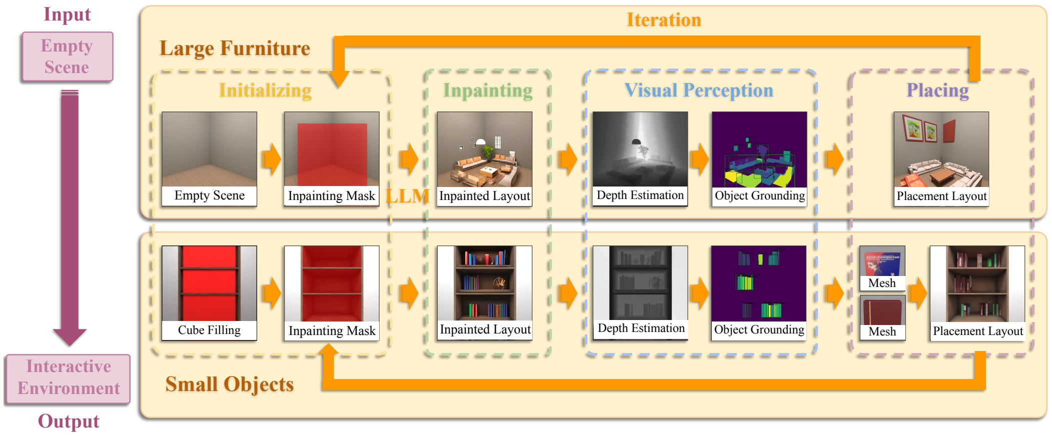


Figure 2: Demonstration of our pipeline that generate complex interactive environment starting from empty scenes, including **Initializing**, **Inpainting**, **Visual Perception** and **Placing** modules.

steps: (1) **Initializing**: We begin by selecting a view in the scene and rendering an image using Pyrender [Matl, 2019] and LuisaRender [Zheng et al., 2022], and generating an inpainting mask; (2) **Inpainting**: Given a text prompt generated by LLMs, along with the rendered image and masks from step 1, we inpaint the image according to the text prompts, leveraging Latent-Diffusion [Rombach et al., 2021]; (3) **Visual Perception**: Upon recognizing the inpainted image, we generate 3D bounding boxes for objects using GPT4v, Grounded-SAM, and Marigold [OpenAI, 2023, Kirillov et al., 2023, Liu et al., 2023, Ren et al., 2024, Ke et al., 2024], as well as the rendering parameters from step 1; (4) **Placing**: We place objects into the scene according to the 3D bounding boxes and return to step 1 to continue generating new objects in the next iteration.

3.1 Initializing Module

Overview In this module, we initialize the viewpoint for the iteration, rendering an image and obtaining the inpainting mask and ground-truth depth for the following steps.

Large Furniture Given an empty room, we heuristically choose the first view that spans from one corner to the opposite corner, maximizing the visible space of the room to render an image, as in Figure 2 *Empty Scene*. Setting a light source in the middle top of the room, we use a ray-tracing based method to render a high-quality image and a raster-based method to obtain the ground-truth depth and object segmentation. Next, we generate an inpaint mask for the image, which is centered within the frame, as shown in Figure 2 *Inpainting Mask*. If objects are already present in the scene, we utilize the object segmentation mask obtained from the rasterizer to filter out those pixels from the inpaint mask, ensuring that any furniture or objects placed within the room will not disappear from the newly inpainted images.

Small Objects As illustrated in Figure 2 *Small Objects*, to place small objects on large furniture, we first heuristically choose a front-top view or a front view depending on whether we are placing on top of a large object (e.g., a table) or inside an object (e.g., a shelf). To obtain the inpainting mask, we place a cube within or atop the bounding box of the larger furniture object, with a size slightly smaller than that of the furniture object itself, as shown in Figure 2 *Cube Filling*. In cases placing small objects on top of larger ones, we position the cube on top of the larger object's bounding box, with a fixed height and the other two dimensions slightly smaller than those of the larger object. Similar to large furniture, we remove the pixels of existing small objects from the inpainting mask.

3.2 Hierarchical Inpainting Module

Overview In this module, we utilize the image and inpainting mask provided from the previous step to first generate text prompts using LLMs. Subsequently, we use these prompts for image inpainting.

The inpainting process for both large furniture and small objects is depicted in Figure 2 *Inpainting*. To ensure smoother inpainting results, we apply techniques such as erosion and Gaussian blur to the mask before commencing the inpainting process. This preparation allows for more effective filling of the contents within the mask. To enhance the diversity and ensure the generation of reasonable objects, we prompt LLMs to automatically generate suitable text prompts and negative prompts for the inpainting model. For example, when we input the configuration of a partially generated living room that includes a TV set into an LLM, the LLM will place the word "TV" in the negative prompt, reasoning that a living room normally has only one TV set.

During this step, we generate multiple inpainted images. If the number of recognized objects in a generated image falls below a predefined criterion, we filter out this image and generate new ones.

3.3 Visual Perception Module

Overview This module takes the inpainted image, ground-truth depth, and camera parameters to recognize and segment objects, estimate their depth, back-project them into 3D, and finally output 3D bounding boxes for each object.

For object recognition, we initially utilize GPT4v to detect all objects present in the image. The identified object names then serve as tags for Grounding-Dino, which performs object detection and provides the output bounding boxes. Following this, we use the SAM to obtain instance-level segmentation masks based on these bounding boxes, as depicted in Figure 2 *Object Grounding*.

After that, we estimate the relative depth of the generated image and rescale the predicted depth using reference depth. Specifically, giving n reference pixel set $P_r = \{(i_0, j_0), ..., (i_{n-1}, j_{n-1})\}$, referenced depth map D_r in $R^{W \times H}$ where W and H are the resolution of the image, estimated depth map D_e in $R^{W \times H}$, we rescale the estimated depth map to $D_{rescaled}$ in the following formulas:

$$\text{max}_r = \max_{t \in [0,n-1]} \left(D_r^{(i_t,j_t)} \right), \text{min}_r = \min_{t \in [0,n-1]} \left(D_r^{(i_t,j_t)} \right), \text{max}_e = \max_{t \in [0,n-1]} \left(D_e^{(i_t,j_t)} \right)$$

$$\text{min}_e = \min_{t \in [0,n-1]} \left(D_e^{(i_t,j_t)} \right), \text{scale} = \frac{\text{max}_r - \text{min}_r}{\text{max}_e - \text{min}_e}$$

$$D_{rescaled} = D_e \cdot \text{scale} - \frac{1}{n} \sum_{t \in [0,n-1]} D_e^{(i_t,j_t)} \cdot \text{scale} + \frac{1}{n} \sum_{t \in [0,n-1]} D_r^{(i_t,j_t)}$$

We employ different strategies when selecting reference pixels P_r. For placing large furniture, we utilize all the non-masked parts of the image as reference pixels to ensure general consistency with the room's floors and walls. For small objects, we focus on the non-masked areas of large furniture as reference pixels, aiming for consistency specifically with the inpainted object. This approach is adopted because the depth information outside the inpainted objects exhibits discontinuities that are challenging to predict accurately. For instance, as shown in Figure 2 *Depth Estimation*, predicting the depth of the wall behind the shelf is difficult and could introduce noise if considered.

Once the depth estimation is acquired, we use the camera parameters from the rendering process to back-project the depth into a 3D point cloud. Utilizing the 2D masks provided by SAM, we extract the point cloud for each object instance. To eliminate any outliers, we apply DBSCAN[Khan et al., 2014] clustering to each segmented object, which allows us to derive axis-aligned bounding boxes for each object.

3.4 Placing Module

Overview Equipped with the 3D bounding boxes of objects, this module places the objects into the simulation and returns to the **Initializing** phase to commence the next iteration.

Large Furniture To incorporate large furniture into a scene utilizing 3D bounding boxes, we initially retrieve each piece of furniture according to a text description generated by GPT4v. After extracting a list of instances from the datasets, we select them based on feature similarity and the proportionality of their scale in three dimensions. The scale of each item is adjudicated by large language models using common sense knowledge.

Owing to the possibility that the retrieved furniture may not precisely conform to the 3D bounding boxes and minor errors in depth estimation, directly placing furniture at the center of the bounding box can lead to issues such as collisions or complications arising from partial view observations. To mitigate these issues, we adopt an alternative approach by generating constraints derived from the 3D bounding boxes. We employ search methods akin to those used in Holodeck [Yang et al., 2024c] to determine optimal placement. Unlike Holodeck, which utilizes LLMs to generate all constraints, we derive ours directly from the generated images and 3D bounding boxes. This search process enables us to avoid collision conflicts while simultaneously ensuring alignment with the generated image. Further details about these constraints can be found in Appendix A.

Small Objects To position small objects within a scene as defined by 3D bounding boxes, we first generate 3D instances based on the semantic content of each object and then place them at the specified positions within the 3D bounding boxes. Subsequently, we adjust the orientation and scale of the small objects to match the orientation and size of the bounding boxes. Notably, due to the partial nature of point clouds, it is impractical to uniformly apply the scale of the bounding box across all three dimensions. Instead, we focus on utilizing the scale along the dimensions perpendicular to the viewing direction. The placement process for small objects is depicted in Figure 2.

4 Experiments

Dataset We retrieve objects from Objaverse. Deitke et al. [2023b,a] Objaverse is a dataset that contains massive annotated 3D objects. It includes various objects, including manually designed objects, everyday items, historical and antique items, *etc*. In the process of generating indoor scene objects, we retrieve suitable furniture from the Objaverse dataset and place them in the scene.
We also retrieve articulated objects from PartnetMobility [Xiang et al., 2020]. PartnetMobility contains 2346 3D articulated models from 46 categories, with articulation annotations.

Implementation We use Marigold [Ke et al., 2024] as the depth estimation model. We use Grounded-Segment-Anything as our segementation model. We use the SD-XL [Podell et al., 2023] inpainting model provided by diffusers as the image inpainting diffusion model. We use LuisaRender [Zheng et al., 2022] as our renderer. For text-to-3D generation, we first use MVdream [Shi et al., 2023] to generate a image and then feed the image to InstantMesh [Xu et al., 2024] to generate the 3D asset. All experiments, including qualitative evaluation, quantitative evaluation and robotics task are all conducted on an A100 GPU.

Baseline We compare ARCHITECT with state-of-the-art indoor scene generation approaches: (1) Holodeck [Yang et al., 2024c], leveraging common sense from LLMs to generate floor plans and place objects; (2) Text2Room [Höllein et al., 2023], utilizing the knowledge from image diffusion models and depth estimation models to generate the entire mesh of a scene; (3) DiffuScene [Tang et al., 2023a], learning a diffusion model to generate the layout of 3D objects in a scene. In addition to the methods mentioned above, AnyHome [Wen et al., 2023] is another baseline we would like to compare with. However, the method is not yet fully open-sourced, so we will leave it for future work.

Metric To evaluate the semantic correctness of the generated scenes, we use the following metrics that calculate the similarity between a rendered image and a given caption: (1) CLIPScore [Hessel et al., 2021], computing the correlation between image feature of the rendered image extracted by CLIP image encoder and text feature of the caption extracted by CLIP text encoder; (2) BLIPScore, using the image-text-matching head of BLIPv2 [Li et al., 2023b] to compute alignment between the rendered image and caption; (3) VQAScore [Lin et al., 2024], feeding the rendered image and caption into a VQA model, returning the probability that the answer to the question "Does the image show caption" is "Yes" as the score; (4) GPT4o Ranking, asking GPT4o to rank rendered images of generated scenes, and calculate the average ranking as the score. We also conduct a user study to evaluate various aspects of the generated scenes. We ask users to rate the following four indicators from 1 to 5: Visual Quality (VQ), Semantic Correctness (SC), Layout Correctness (LC) and Overall Preference (OP).

4.1 Scene Generation

Qualitative Evaluation We compare rooms generated by our model with those generated by other methods. Figure 3 shows the results for both household and non-household scenes. The comparisons with Text2Room and DiffusScene are provided in Appendix B, as these methods generate either non-interactive scenes or lack diversity.

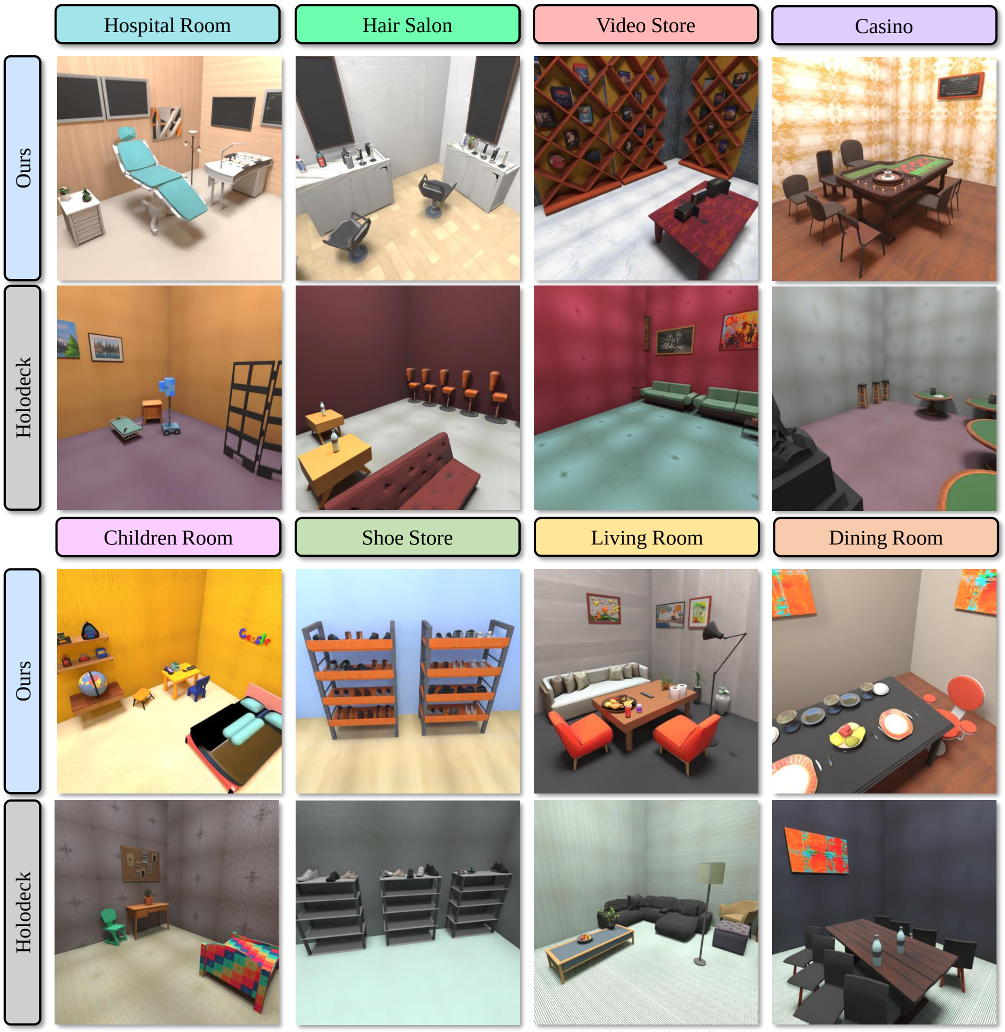


Figure 3: We compare ARCHITECT with other methods in both household scenes(living room and dining room) and other non-household scenes. We only compared the household scene generated by Diffuscene due to its limitations in Figure 7 and compared with Text2Room in Figure 8.

Compared to Holodeck, our scenes are more realistic, leveraging the capabilities of 2D diffusion models. For example, in the hair salon scenario, Holodeck fails to generate semantically correct scenes because the spatial constraints and objects are entirely generated by LLMs. Additionally, our work demonstrates the ability to generate more complex and detailed placements of small objects. For instance, the shoe store filled with paired shoes, toys on the children's room shelf, and the organized placement of items on the dining room table all exceed the generative abilities of Holodeck. This comparison highlights the effectiveness of our iterative and hierarchical inpainting process, showing that the use of 2D generative models indeed brings more spatial priors compared to LLMs.

Quantitative Evaluation We compare ARCHITECT to other state-of-the-art indoor scene generation results using 2D image scores and user studies. The results are shown in Table 2 and Table 4. ARCHITECT outperforms others in CLIP score, BLIP score, and GPT-4o ranking, while achieving a relatively high VQA score (only slightly lower than Text2Room). It demonstrate that our generated scene are generally better aligned with the room caption (with better semantic and layout coherence). In GPT-4o's explanations of it's ranking, we found some common points of our previous analysis. Rooms generated by Text2Room are often criticized for having artifacts and distortion and as a result are often ranked lower; rooms generated by Holodeck are often described as simply

Method	Text-Image Scores				User Study			
	CLIP↑	BLIP↑	VQA↑	GPT4o↓	VQ↑	SC↑	LC↑	OP↑
Diffuscene	0.6785	0.4310	0.7561	-	3.76	3.52	3.37	3.50
Text2Room	0.6491	0.1223	**0.8149**	2.64	2.79	3.62	3.04	3.12
Holodeck	0.6502	0.3463	0.5696	1.91	3.34	3.14	3.11	3.07
Ours	**0.7173**	**0.5859**	0.8073	**1.36**	**3.87**	**3.76**	**3.65**	**3.71**

Table 2: Quantitative Comparison. We evaluate 2D image metrics, including CLIP Score, BLIP Score, VQA Score and GPT4o ranking. We also conducted a user study, reporting visual quality(VQ), semantic correctness(SC), layout coherence(LC) and overall preference(OP). GPT-4 ranking involves ranking and therefore does not include Diffuscene which can only generate a limited number of household scenes.

arranged; rooms generated by ARCHITECT are more favored by GPT-4o evaluator. In user study, ARCHITECT outscored other methods in all four aspects. It is worth noting that the visual quality score of Text2Room is significantly lower than other methods, which is likely due to the artifacts and distortions in Text2Room-generated scenes.

4.2 Embodied/Robotic task

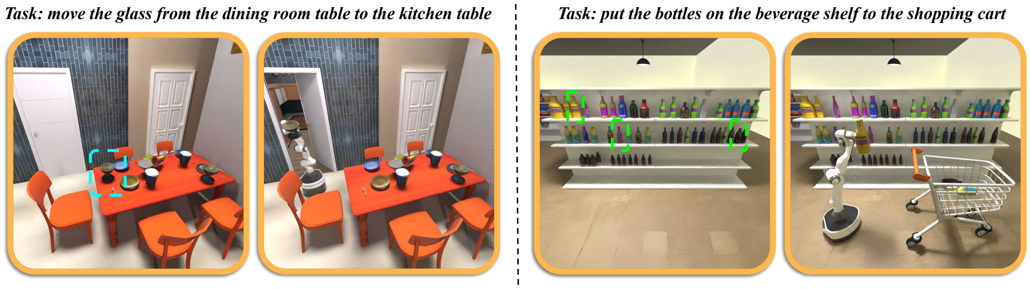

Task: move the glass from the dining room table to the kitchen table
Task: put the bottles on the beverage shelf to the shopping cart

Figure 4: Two robot manipulation tasks generated in our scene setting.

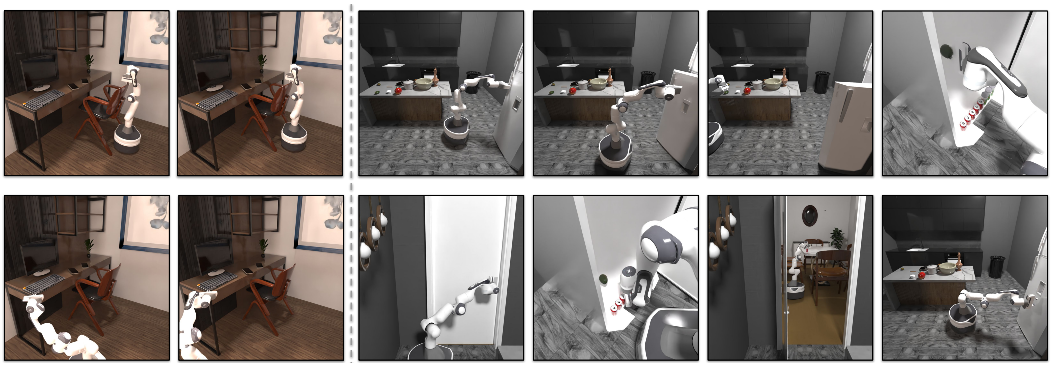


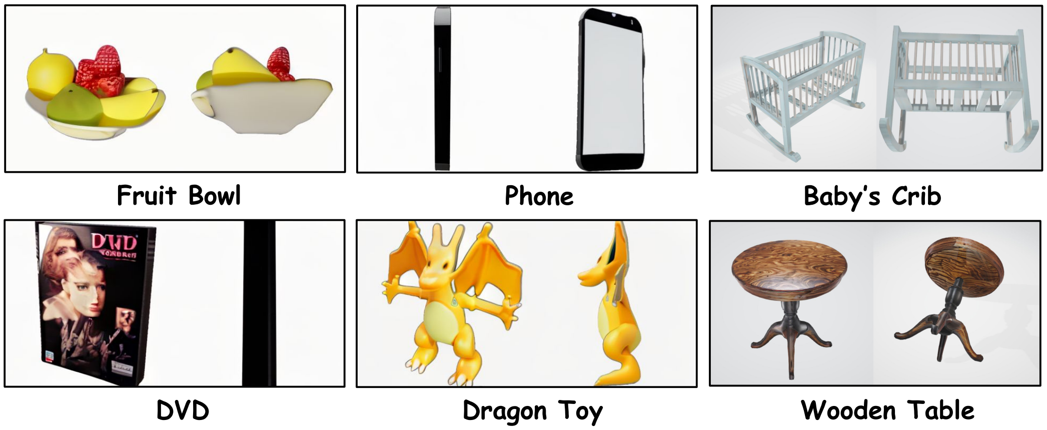
Figure 5: **Left:** the robot organizes the room by pushing the chair under the table and pushing the keyboard inside the table. **Right:** the robot opens the fridge door, grasps the mango and puts it into the fridge, opens the kitchen-dining room door, grasps the soda can and puts it on the dining room table, and finally closes the fridge.

Inspired by previous work RoboGen [Wang et al., 2023b], we are making efforts to collect large-scale data for long-term embodied or robotics tasks in our generated scene.

A significant challenge we face is that the inclusion of all the detailed small objects in the simulation significantly slows down the speed of the embodied environment. To address this, after generating the scene, task and task decomposition, we use LLMs to select relevant objects for each substep, while all other objects are designated as background objects and will not be physically simulated during this substep.

Given our house-level scene generation pipeline, we can now extend RoboGen to generate action trajectories for skills that require long-distance navigation and more complex tasks. Specifically, by inputting the floor plan, large furniture, and small objects into GPT-4, it first generates a task related to the existing objects in the scene. Then, as in RoboGen, it decomposes the task and filters out irrelevant objects to the background. Finally, we leverage action primitives and training supervision generated by LLMs to obtain a trajectory of actions to solve the task. We demonstrate two example generated tasks in Figure 4, and two other task with corresponding collected trajectories in Figure 5. The comparison of diversity of is shown in Table 3 by the self-BLEU score of task discription.

4.3 Object Generation


Fruit Bowl

Phone

Baby’s Crib

DVD

Dragon Toy

Wooden Table

Figure 6: Examples of generated small objects and large furniture.

To address one of our limitations, the dependence on a large furniture database, we apply a pipeline to generate high-quality large furniture. It optimizes a differentiable tetrahedron mesh with SDS loss [Guo et al., 2024], using the normal-depth diffusion model and albedo diffusion model provided by RichDreamer [Qiu et al., 2024] as the main supervision signal. This pipeline is capable of generating high-quality object meshes from text guidance, specifically large furniture in our case. Some results are shown in the right part of Figure 6 right part. We also show some qualitative resutls about small object generation in Figure 6 left part, which is another crucial factor of the quality of generated scenes.

5 Conclusion and Future Work

In this paper, we propose ARCHITECT, a generative framework capable of creating *diverse*, *realistic*, and *complex* Embodied AI environments. Leveraging pre-trained 2D image inpainting diffusion models that better capture scene and object configurations compared to LLMs, ARCHITECT iteratively extracts diverse and realistic layouts from image inpainting results. We also propose to *control* this inpainting process by processing geometric cues in the background of a rendered image. This process effectively controls the camera parameters and depth scale for the generated image, allowing us to project it back into 3D point clouds. The scenes generated by ARCHITECT provide realistic and complex environments for downstream Embodied AI and robotics applications. In qualitative and quantitative comparisons, ARCHITECT outperformed baseline methods in both realism and diversity. We believe ARCHITECT is an important step towards creating large-scale interactive 3D environments.

Limitation and Future work Currently, ARCHITECT retrieves furniture and large objects from datasets. This means that the diversity of furniture in our results is inherently limited by the dataset. In the future, we will explore generative methods to create more high-quality and articulated objects to further enhance the diversity of the generated scenes.

Acknowledgement

We thank the anonymous reviewers for their helpful suggestions. This work is funded in part by grants from Microsoft Accelarate Foundational Models Research Initiative, Cisco, and NSF IIS-240438.

References

Rio Aguina-Kang, Maxim Gumin, Do Heon Han, Stewart Morris, Seung Jean Yoo, Aditya Ganeshan, R Kenny Jones, Qiuhong Anna Wei, Kailiang Fu, and Daniel Ritchie. Open-universe indoor scene generation using llm program synthesis and uncurated object databases. *arXiv preprint arXiv:2403.09675*, 2024.

Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023.

Murtaza Dalal, Ajay Mandlekar, Caelan Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv:2305.16309*, 2023.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023a.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023b.

Sam Earle, Maria Edwards, Ahmed Khalifa, Philip Bontrager, and Julian Togelius. Learning controllable content generators. In *2021 IEEE Conference on Games (CoG)*, pages 1–9. IEEE, 2021.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Qixun Zeng, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li, Yi Liu, Peng Liu, Lin Ma, Le Weng, Xiaohang Hu, Xin Ma, Qian Qian, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3d-front: 3d furnished rooms with layouts and semantics. *arXiv preprint arXiv:2011.09127*, 2020a.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *arXiv preprint arXiv:2009.09633*, 2020b.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

Minghao Guo, Bohan Wang, Kaiming He, and Wojciech Matusik. Tetsphere splatting: Representing high-quality geometry with lagrangian volumetric meshes. *arXiv preprint arXiv:2405.20283*, 2024.

Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *CoRR*, abs/2104.08718, 2021. URL https://arxiv.org/abs/2104.08718.

Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7909–7920, 2023.

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 95–101, 2020.

Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023a.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023b.

Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching, 2023.

Yiqi Lin, Hao Wu, Ruichen Wang, Haonan Lu, Xiaodong Lin, Hui Xiong, and Lin Wang. Towards language-guided interactive 3d generation: Llms as layout interpreter with generative feedback. *arXiv preprint arXiv:2305.15808*, 2023.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. *arXiv preprint arXiv:2404.01291*, 2024.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

M. Matl. PyRender. https://github.com/mmatl/pyrender, 2019.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.

R OpenAI. Gpt-4v (ision) system card. *Citekey: gptvision*, 2023.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34:12013–12026, 2021.

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, 2023.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9914–9925, 2024.

Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021. URL https://arxiv.org/abs/2102.12092.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Jonas Schult, Sam Tsai, Lukas Höllein, Bichen Wu, Jialiang Wang, Chih-Yao Ma, Kunpeng Li, Xiaofang Wang, Felix Wimbauer, Zijian He, et al. Controlroom3d: Room generation using semantic proxy rooms. *arXiv preprint arXiv:2312.05208*, 2023.

Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv:2308.16512*, 2023.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis. *arXiv preprint arXiv:2303.14207*, 2023a.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023b.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. In *Arxiv*, 2023a.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023b.

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, 2023c.

Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5922–5931, 2021.

Zehao Wen, Zichen Liu, Srinath Sridhar, and Rao Fu. Anyhome: Open-vocabulary generation of structured and textured 3d homes. *arXiv preprint arXiv:2312.06644*, 2023.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*, 2024a.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. *arXiv preprint arXiv:2404.09465*, 2024b.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)*, volume 30, pages 20–25. IEEE/CVF, 2024c.

Yizhou Zhao, Kaixiang Lin, Zhiwei Jia, Qiaozi Gao, Govind Thattai, Jesse Thomason, and Gaurav S Sukhatme. Luminous: Indoor scene generation for embodied ai challenges. *arXiv preprint arXiv:2111.05527*, 2021.

Shaokun Zheng, Zhiqian Zhou, Xin Chen, Difei Yan, Chuyan Zhang, Yuefeng Geng, Yan Gu, and Kun Xu. Luisarender: A high-performance rendering framework with layered and unified interfaces on stream architectures. *ACM Transactions on Graphics (TOG)*, 41(6):1–19, 2022.

6 Appendix

A Implementation Details

A.1 Constraint and Search

Furniture can be devided into floor objects and wall objects. In detail, for the floor objects, we rely on the following type of constraints for furniture: *Global*, *Location*, *Distance*, *Relation*, *Alignment* and *Rotation*.

/* Global Constraint */
Edge: at the edge of the room, close to the wall.
Middle: not close to the edge of the room.
Corner: at the corner of the room.
Horizontal/Vertical: the global direction.

/* Distance Constraint */
Near, object: near to the other object.
Far, object: far away from the other object.

/* Relation Constraint */
In front of, object: in front of another object.
Behind, object: behind of another object object.
Left of, object: to the left of another object.
Right of, object: to the right of another object.

/* Alignment Constraint */
Center aligned, object: aligned with another object .

/* Soft Location Constraint */
Location, (x, y): predicted bounding-box location.

/* Rotation Constraint */
Face to, object: face to the center of another object.

Different from Holodeck that is using LLMs to generate all the constraints, all above constraints are generated by sorting floor objects by size and traversing them based on the position of their bounding boxes except for the rotation constraints. Specifically, for each floor object's bounding box, constraints of each type are assigned based on the distance and directional relationship between its boundaries and those of other floor object bounding boxes. In particular, rotation constraints cannot be solely determined by the bounding box, so an LLM is consulted to obtain the rotation constraint based on common sense (e.g., chairs facing a table). After obtaining the complete constraints, a DFS algorithm is utilized to explore possible placements for each item. Placements that do not meet the hard constraints are filtered out, and the highest scoring placements are selected based on the soft constraint scores. Here, hard constraints refer to mandatory constraints, such as global constraints and position constraints. Soft constraints refer to cumulative scores, where the highest scoring options are prioritized, such as location constraints. In practice, we applied a greedy pruning mechanism to the DFS algorithm, exploring only 3 nodes with highest score at each time. The score S_p for each placement is calculated as follows:

$$S_p = W_{loc} \cdot \left(\sum_{i \neq i} \Delta_i \cdot w_i + \frac{w_{cur}}{\Delta_{cur}} + C \right) + W_{rotation} \cdot \left(\sum_{i=1}^{n} \mathbb{1}_{r_i} \right)$$

For each current object, W represents the weight of constraints; w represents the weight of each object; Δ represents the deviation from the reference, which hopes to be close to the reference and away from other items already placed; C represents constants to keep result positive; $\mathbb{1}$ is a indicator function that equals 1 if the rotation satisfies the constraint; r represents the rotation of the item.

As for wall objects, most of its constraint comes from floor. Wall objects constraints are as follows:

> /* Global Constraint */
> **Above, object:** close to the wall, above a specific floor object.
>
> /* Soft Location Constraint */
> **Location, (x, y):** predicted bounding-box location.
>
> /* Position Constraint */
> **Height:** The height of the object.

The placement of wall objects is relatively simpler because it does not require specifying an orientation; by default, they face away from the wall. Additionally, the likelihood of conflicts on the wall is lower, and using soft location constraint suffices for effective arrangement. And the searching process of wall objects is just almost the same as floor objects.

A.2 Large Furniture Retrieving

Following Holodeck, for each piece of large furniture, we first retrieve multiple candidates from the dataset using text descriptions of the assets. Then, we select one asset from the retrieved candidates based on scale similarity, which is calculated as the L1 difference between the scale of 3D bounding box of object point cloud and the 3D bounding box of object mesh. Additionally, we integrated image similarity using the cosine similarity of CLIP features in the selection process in our latest pipeline. Here, scale similarity and image similarity are used only in the candidate selection process rather than the retrieval process, since there could be significant occlusions in the image (e.g., a chair behind a table) that could greatly influence the accuracy of retrieving.

A.3 Small Objects Generation and Selection

We use a text-to-3D pipeline (text-to-image and image-to-3D) to generate 3D assets for small objects. To make the scene more reasonable and resemble the inpainted image, we generate multiple candidates for each type of object and use the cosine similarity of DINO features to select from the candidates. We also experimented with an image-to-3D pipeline, starting from the object image segmented from the inpainted image. However, the resolution of the segmented image is low, resulting in sub-optimal 3D shapes and textures.

A.4 View Selection

For large furniture placement, we heuristically select up to three views (right-back corner to left-front corner, front middle to back middle, and left-back corner to right-front corner) that can cover the whole room area for inpainting. Assuming the room ranges from $(0, 0)$ to (x, y), the three views would be looking from $(x, y, 1.8)$ to $(0, 0, 0.5)$, from $(\frac{x}{2}, 0, 1.8)$ to $(\frac{x}{2}, y, 0.5)$, and from $(0, y, 1.8)$ to $(x, 0, 0.5)$. We stop inpainting from new views when the occupancy of the room is larger than 0.7 or it has been inpainted from all three views.

Additionally, we use an 84-degree FOV for our camera during rendering, a standard parameter for real-world cameras. Consequently, for a square room, this setup results in approximately 95 percent of the room being visible from a single corner-to-corner view.

For small object placement, we first ask LLMs to determine which objects can accommodate small objects on or in them, and then inpaint each of them with heuristic relative views. For objects like tables or desks on which we are placing items, we use a top-down view. For shelves or cabinets in which we are placing objects, we use a front view. The distance of the camera from the object is adjusted according to the scale of the object and the camera's FOV, ensuring the full object is visible during inpainting.

B More Experiments

B.1 More Comparison Cases

As shown in Figure 7 and 8, compared to Text2Room, our method generates more photorealistic scenes. However, since Text2Room directly projects RGB pixels into 3D space using depth maps, artifacts are unavoidable, the geometry of the generated 3D mesh is distorted, and it can't serve as an embodied environment since it's not interactive. These issues are addressed in retrieval-based methods. Compared to Diffuscene in Figure 7, our method generates more reasonable and detailed scenes. Our results exhibit more detail and precision. Additionally, the dataset lacks information on the placement of small objects, limiting the diversity of small object placement in Diffuscene-generated scenes.

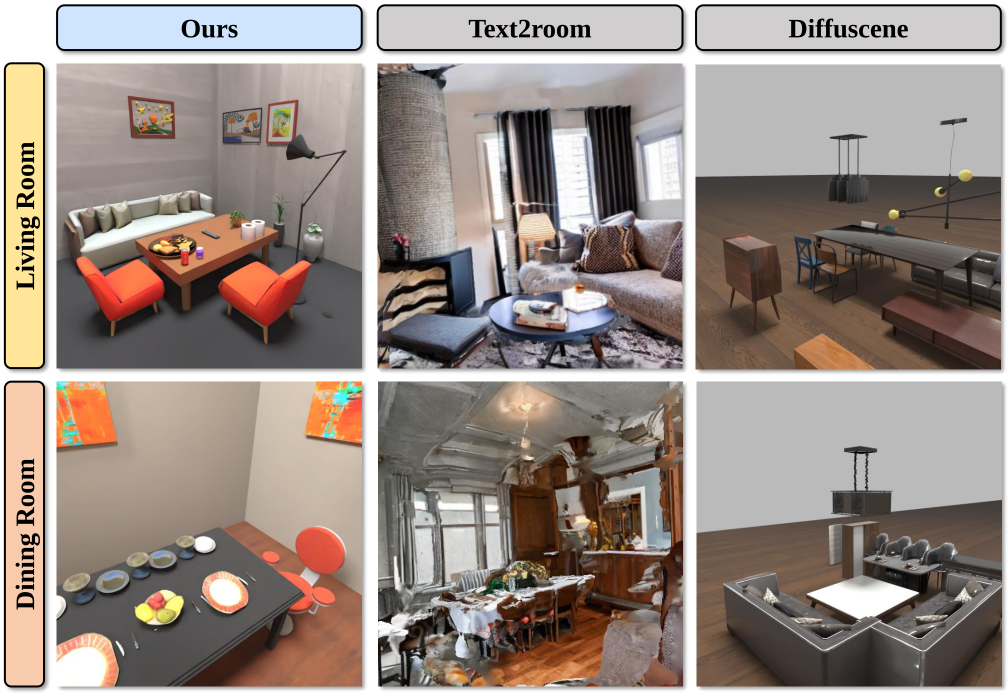


Figure 7: Comparison of living room and dining room scene generated by ARCHITECT, Text2room, Diffscene.

B.2 Comparison with Baselines

The comparison with PhyScene is shown Table 4, where we present the comparison results for generated living rooms (only the weight of living room generation is released).

In our experiments, we aim to provide a general comparison with three types of related works: works that generate only the static mesh, works that are trained on existing datasets and works that generate open-vocabulary scenes using foundation models.

It's challenging to make a completely fair comparison between our methods and the Text2Room method since they serve different purposes. While Text2Room generates the entire mesh without retrieving objects, none of its assets are interactive and it may achieve higher photorealism by directly generating meshes from 2D images.

B.3 Controllability and Editing

In short, our method combines a diffusion-based pipeline with an LLM-based method, which still possess the ability of controlling and editing. The inpainting-to-layout pipeline functions

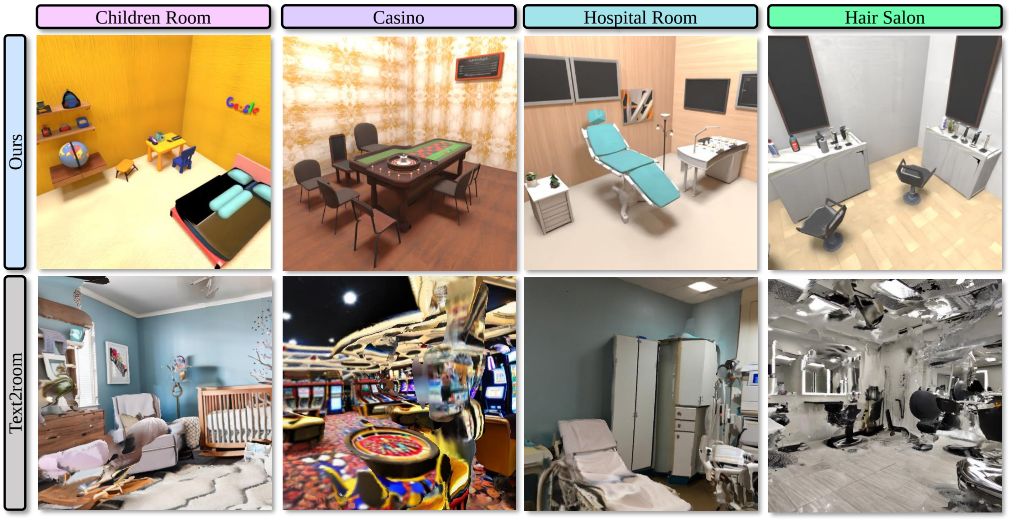


Figure 8: Comparison of four scenes generated by ARCHITECT, Text2room.

can be considered as an API function callable by the LLM. Our approach aims to generate scene configurations seeded from diffusion models, with scene editing as an othogonal feature enabled by LLMs. Specifically, the scene configuration generated by our pipeline can be represented by each object's name, position, scale, bounding box, orientation, and asset UID, which can be easily converted to text representations. This allows us to feed this information directly into LLMs to further control or edit the scene layout. Corresponding experiments could be found in Appendix C.

B.4 Comparison of Architect and LLMs in Small Object Placement

It's challenging for LLMs to directly solve arrangement problems. First, for small object placement on shelves, LLMs lack information about supporting surfaces, making it impossible for them to solve this issue. Second, for placement on tables, while we might know the supporting surfaces given the bounding boxes, LLMs struggle with object orientations, often resulting in less complex scenes or scenes with severe collisions. We show a comparison of small objects generated by our methods and LLMs in the middle part of Figure 9 and in Table 3.

B.5 Similarity of Generated Scenes and Inpainted Images

We've also evaluated the image similarity between inpainted images and images of generated scenes against empty scenes, as shown in Table 3. The results indicate that, although not exactly the same, the generated scenes are to some degree faithful to the image generation results.

B.6 Consistency of Inpainting

The appearance of the masked area is consistent with other areas both stylistically and geometrically. We also apply a commonly used technique, softening the boundary of inpainting masks, to improve consistency. A comparison of the results before and after using softened inpainting masks is shown in the left part of Figure 9.

	Large Furniture	Small Objects
	Similarity (%) ↑	Similarity (%) ↑
Empty vs. Inpaint	45.33	77.85
Inpaint vs. Placed	**85.04**	**83.83**
LLM Placement VQScore ↑	74.93	
Our Placement VQScore ↑	**80.81**	
RoboGen Self-BLEU ↓	0.284	
Ours Self-BLEU ↓	**0.198**	

Table 3: Quantitative experimental Results.

Method	CLIP ↑	BLIP ↑	VQScore ↑
PhyScene	71.42	46.51	88.72
Holodeck	69.37	53.23	84.06
Text2Room	64.91	12.22	90.73
Diffuscene	65.32	49.95	87.28
ARCHITECT (Ours)	**72.96**	**63.62**	**94.58**

Table 4: Experimental result of comparison with PhyScene.

C Scene Editing

To demonstrate that our pipeline is compatible with scene editing and complex text control, we implemented additional APIs to add, remove, and rescale objects, enabling LLMs to edit the scene.

Initial results for scene editing are shown in the right part of Figure 9. We issued commands to LLMs such as replace the books on the shelf with vases, replace the bookshelf with a cabinet, and make the bookshelf smaller. The LLMs achieved the correct results by calling the provided APIs.

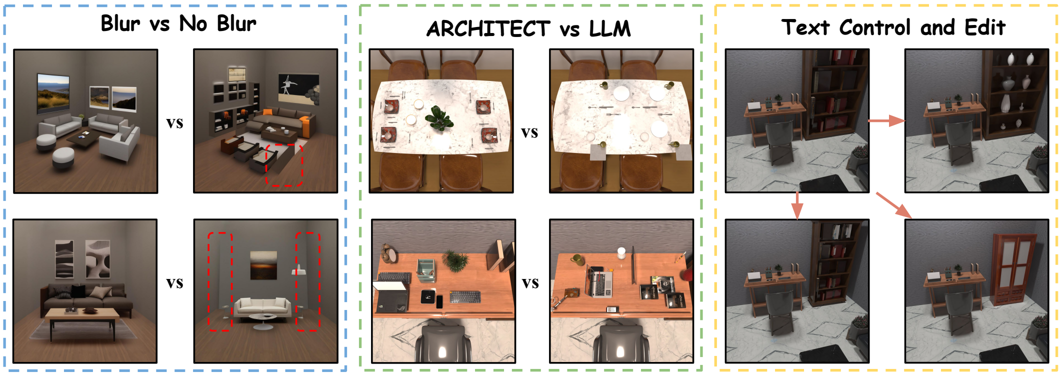


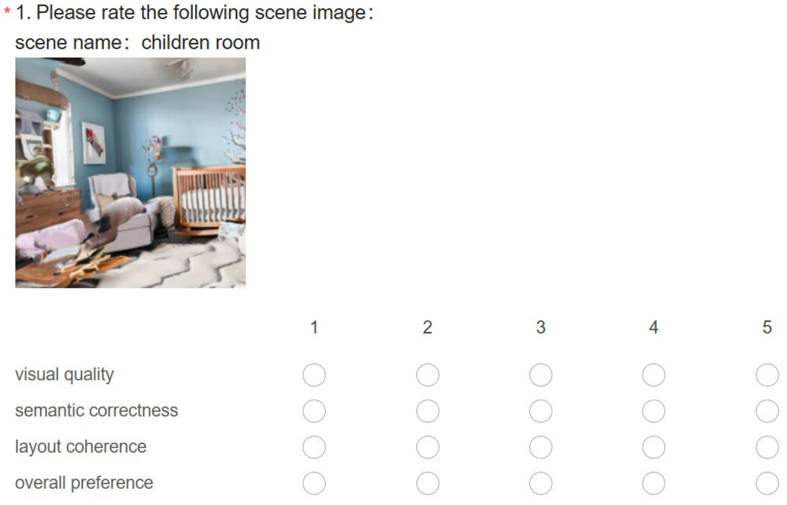
Figure 9: Demonstration of comparison between different mask, different small objects placement and effect of text control. Red dashed box indicates inconsistency when using non-blur mask.

D User Study Details

We conducted comprehensive human evaluations to assess the quality of ARCHITECT scenes, with a total of 115 undergraduate students and graduate students participating in the user studies. All participants were volunteers without compensation.

We first provided participants with two minutes to read the instructions, the specific content of which was as follows:

· Thank you for your participation! This questionnaire is used for the experimental part of scientific research articles, which requires human evaluation. We will keep the information of the participants confidential.
· The estimated total time is about 10 minutes.
· There are some pictures in the questionnaire from different utils. Just observe and score all of these pictures. The higher the score, the better the quality.
· An example is shown in the following image:



* 1. Please rate the following scene image：

scene name： children room

	1	2	3	4	5
visual quality	○	○	○	○	○
semantic correctness	○	○	○	○	○
layout coherence	○	○	○	○	○
overall preference	○	○	○	○	○

Figure 10: The example questionnaire for participants.

Then, we randomly assigned each volunteer 23 scenes and asked them to score the scenes from one to five, considering visual quality, semantic correctness, layout coherence, and overall preference. Each volunteer received only 23 scenes to ensure they could complete their responses in approximately 10 minutes.

We collected ratings from all participants and calculated the average scores for these four metrics. These average scores were used to evaluate our model, Holodeck, Text2Room and DiffuScene allowing us to compare the scene generation performance of our model with the two baseline models.

The average response time was 525 seconds, with the longest response time being 1113 seconds and the shortest being 150 seconds. Responses with a duration of less than 230 seconds were filtered out.

E Prompts

E.1 Object Recognition Prompt

```
Detect all objects in the picture, generate a description for each object, and decide whether it
is floor-object or wall-object.

Here are the definitions of object types:
floor object: object that is placed on floor or in direct contact with the floor.
wall object: object that is placed on wall and not in contact with the floor.

Here is a sample answer:
table: A big yellow table | floor-object
chair: A gray armchair | floor-object
tv: A black wall-mounted television | wall-object

Requirements:
1. Description should not be too long.
2. You should only give the result and no unnecessary words.
3. Don’t describe the positional relationship between objects.
4. Classification can only be **floor-object**, **wall-object**.
5. Please pay attention to only large furniture like sofa, table, lamp, shelf, and ignore small
objects like bottles or books.
```

The prompt above is fed into GPT-4V along with an image generated by a 2D inpainting model. The prompt asks GPT-4V to recognize all objects in the inpainting image, briefly describe them, and then classify them as either objects on the floor or objects on the wall. The generated object names will be used to prompt Grounded-SAM. The generated descriptions will be used for retrieving objects or for text-to-3D generation.

E.2 Inpainting Prompt and Generation Prompt

```
Given the objects in the current scene, please list which objects have already reached their
potential limits, and the objects are still lacking.

Your answer should be in the following format:
reached limit: object A, object B, ...
lacking: object C, object D, ...

The objects in the current scene are: /* a list of objects with quantities, eg: 2 sofa, 1 coffee
table, 1 TV*/

Remember, do not answer anything not asked. The lacking objects should ideally contain
objects that are not in the scene. The lacking objects you list should be precise, do not give
things like "other furniture".
```

The prompt above asks GPT-4V to provide negative prompts and positive prompts in addition to the room caption for the inpainting model. ROOM-CAPTION will be substituted with the actual room caption. The lacking objects will be added to the positive prompt, and the objects that have reached their limit will be added to the negative prompt.

F Societal Impacts

F.1 Positive Impacts

- **Advancements in Robotics and AI**: ARCHITECT enhances the development of versatile robots capable of assisting in various tasks.

- **Educational Tools**: Generated 3D environments can be used for immersive learning experiences, aiding in the understanding of spatial relationships and complex systems.
- **Accessibility**: Improved AI environments can lead to the development of assistive technologies for individuals with disabilities, enhancing their quality of life.

F.2 Negative Impacts

- **Job Displacement**: Advanced AI and robotics could potentially displace jobs in certain sectors, necessitating consideration of economic and societal impacts.
- **Bias and Fairness**: Ensuring training data and algorithms are representative and fair is crucial to avoid perpetuating existing biases.
- **Misuse of Technology**: Inferring internal geometric structures of 3D objects could be misused for unauthorized reproduction or surveillance, leading to ethical and legal concerns.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions, a zero-shot generative pipeline that creates diverse, complex, and realistic 3D interactive scenes and the paper's scope are accurately reflected by main claims in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discuss the limitations in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not provide theoretical results, it provides a practical 3D scene generation pipelines instead. Thus it does not provide full set of assumptions and proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides clear and specific description of the pipeline that places large furniture and small objects. All the information needed to reproduce the main experimental results are provides, thus, the paper is easy to be reproduced.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code will be made publicly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The ARCHITECT method that the paper proposed dose not need for training data of indoor layouts, which is specified in main text.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide experiments compute resources in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the positive and negative social impacts thoroughly in Appendix F.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks, we are using all existing pre-trained models and datasets rather than releasing new ones.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets, used in the paper are properly credited and the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new ARCHITECT method is well documented and the documentation is provided alongside the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: The paper include the full text of instructions given to participants and screenshots in appendix. The participants are volunteers with no compensation.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: There are no potential risks for participants and the IRB approvals were obtained.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.