AirBlender: Enhancing Interactive 3D Environment Generation with Text-driven Modifications

Agam Bhatia
Computer Science Department
Stanford University
agam2026@stanford.edu

Abstract

3D simulated environments play a critical role in developing embodied artificial intelligence agents, but their creation requires expertise and extensive manual effort, restricting their diversity and scope. To mitigate this limitation, we present AirBlender, a system that modifies 3D environments to match a user-supplied prompt fully automatedly in an end-to-end manner. AirBlender can modify diverse indoor scenes, adjust scene styles and themes, and can capture the nuances of complex queries such as "place the table horizontally in parallel to the bed" and "make the sofa look rustic". AirBlender leverages a fine-tuned vision model, CLIP, for asset retrieval in data generated by perturbing existing scene datasets and uses a large collection of 3D assets from Objaverse to populate the scene with diverse objects. We show that the performance of a depth estimator, Marigold, which is derived from Stable-Diffusion-2, finetuned on AirBlender data results in better spatial understanding than off-the-shelf diffusion models. The improved performance of a depth estimator like Marigold on a small dataset acts as a proof of concept for AirBlender as it extends the utility of modified 3D scenes in various domains such as virtual reality, game development, and embodied AI training environments by providing automatic data augmentation for training.

1. Introduction

Realistic, diverse, and interactive 3D simulated environments serve as vital platforms where AI systems can learn and refine complex behaviors in a controlled yet flexible setting. Generating realistic, diverse, and interactive 3D environments plays a crucial role in the success of this process and facilitates better training of embodied AI agents.

Conventional methods for creating 3D training environments— such as manual design [3], 3D scanning [2], and procedural generation [5] — lack the scalability and adaptability needed for the dynamic requirements of AI research, which demands diverse scenarios for robust agent training. Methods like manual design involve meticulous modeling by experts, which is not only time-consuming but also limits the frequency of updates and revisions. Similarly, 3D scanning provides high-fidelity reproductions of real-world spaces, but is constrained by the physical availability and the static nature of scanned environments. Procedural generation offers a more scalable approach, allowing environments to be created according to predefined rules; however, this method often lacks the flexibility needed to accommodate specific or unique training scenarios, and can result in environments that feel unnatural due to their formulaic construction.

There is a critical need for tools that simplify and democratize the creation of 3D environments, allowing for quick modifications without extensive 3D modeling skills. By enabling easier and faster customization of training scenarios, these tools can significantly accelerate AI research and development, broadening the scope of tasks AI agents can learn and enhancing their ability to operate in varied, complex scenarios.

In light of these challenges, we present AirBlender, a language-guided system tested using Marigold[12], a diffusion model derived from Stable-Diffusion-2, [18] to automatically modify diverse, customized, and interactive 3D embodied environments from textual descriptions. Given a user described modification, AirBlender parses the query to get user intent. AirBlender then uses its hierarchical framework of parsing user queries to generate low-level actions to modify existing scenes according to the given prompt using generated Blender code. It then chooses from over 50K diverse and high-quality 3D assets from Objaverse [4] to satisfy a myriad of environment descriptions. Motivated by the emergent abilities of Large Vision-Language Models (VLMs), AirBlender exploits the commonsense priors and spatial knowledge inherently present in VLMs like Stable-Diffusion-2 [21]. Using pairs of modified images along with a supplied user conversation, AirBlender pro-

vides a basis for enhanced controllability and contextual understanding previously absent in diffusion models. To test the effectiveness of AirBlender for training, we focus on AirBlender's contribution in aiding zero-shot monocular depth estimation on Marigold. When finetuned on a small dataset with minimal compute resources, we find Marigold performs comparably in out-of-distribution tasks, serving as a proof-of-concept for the scalability of AirBlender as a method for accurate scene modifications.

2. Related Work

Embodied AI Environments Previous work in this area relies on 3D artists to design these environments. This has several scalability problems, as design modifications need to be extremely well thought out and specific, and require days or sometimes weeks for iteration and improvement requiring human labor. Additionally, scenes generated using this method are less interactive. The procedural generation framework PROCTHOR [5] showcases its potential to generate largescale interactive environments for training embodied agents. Phone2Proc [3] uses a phone scan to create training scenes that are semantically similar to the desired real-world scene. A concurrent work, RoboGen [25], proposes to train robots by generating diversified tasks and scenes. These works parallel our concept, AirBlender, which aims to train generalizable embodied agents and presents an avenue for further exploration in text-driven 3D interactive scene generation.

LLMs for Scene Design Many works on scene design either learn spatial knowledge priors from existing 3D scene databases [17] or leverage user input and refine the 3D scene iteratively. However, having to learn from datasets of limited categories such as 3D-FRONT [7] restricts this method's applicability. Recently, Large Language Models (LLMs) were shown to be useful in generating 3D scene layouts. However, their methods of having LLMs directly output numerical values can yield layouts that defy physical plausibility (e.g., overlapping assets). In contrast, AirBlender builds on HOLODECK [26] which uses LLMs to sample spatial relational constraints and a solver to optimize the layout, ensuring physically plausible and realistic scene arrangements.

Diffusion Models Denoising Diffusion Probabilistic Models (DDPMs) [9] are a powerful class of generative models that learn to reverse a diffusion process that progressively degrades images with Gaussian noise so that they can draw samples from the data distribution by applying the reverse process to random noise. This idea was extended to DDIMs [23], which provide a non-Markovian shortcut for the diffusion process. Conditional diffusion models are an extension of DDPMs [8] that ingest additional information on which the output is then conditioned, similar to cGAN [16] and cVAE [24]. In the realm of text-based image generation, [21] have trained a diffusion model on the large-scale image and text dataset LAION-5B [22] and demonstrated image synthesis with previously unattainable quality. The cornerstone of their approach is a latent diffusion model (LDM), where the denoising process is run in an efficient latent space, drastically reducing the complexity of the learned mapping. Such models distill internet-scale image sets into model weights, thereby developing a rich scene understanding prior, which is used in this work for monocular depth estimation.

Text-driven 3D Generation Early endeavors in 3D generation focus on learning the distribution of 3D shapes and/or textures from category-specific datasets [7]. Subsequently, the advent of large vision-language models like CLIP [18] enables zero-shot generation of 3D textures and objects. These works excel at generating 3D objects but struggle to generate complex 3D scenes. More recently, emerging works generate 3D scenes by combining pre-trained text-to-image models with depth prediction algorithms to produce either textured meshes or NeRFs [15]. However, these approaches yield 3D representations that lack modular composability and interactive affordances, limiting their use in embodied AI. In contrast, AirBlender utilizes a comprehensive 3D asset database to generate semantically precise, spatially efficient, and interactive 3D environments suitable for training embodied agents and showcases that such a method can improve scene generation and understanding in diffusion models.

3. Data

Marigold is a monocular depth estimator developed on a latent diffusion model that requires ground truth depth labels for training.

Training Datasets In this work, we train Marigold exclusively with 2000 synthetic data samples generated from AirBlender. We find that real depth datasets suffer from missing depth values caused by the physical constraints of the capture rig and the physical properties of the sensors. Specifically, the disparity between cameras and reflective surfaces diverting LiDAR laser beams are inevitable sources of ground truth noise and missing pixels. On the other hand, synthetic depth is inherently dense and complete, meaning that every pixel has a valid ground truth depth value, allowing us to feed such data into the variational auto-encoder, which cannot handle data with invalid pixel values. Second, synthetic depth is the cleanest possible form of depth, which is guaranteed by the rendering pipeline of AirBlender. Since we are using a text-to-image Latent Diffusion Model, synthetic depth gives the cleanest set of examples and reduces noise in gradient updates during the fine-tuning protocol. Incomplete samples are filtered out. RGB images and depth maps are resized to 480×640 size during preprocessing. Additionally, we transform the original distances relative to

the focal point into conventional depth values relative to the focal plane. The generation process for AirBlender data is described in the Methods section.

Evaluation Datasets Marigold is evaluated on 3 datasets consisting of real images that are not seen during training by the model. NYUv2 [11] and ScanNet [1] are indoor scene datasets captured with an RGB-D Kinect sensor. For NYUv2, we utilize the designated test split, comprising a total of 654 images. In the case of the ScanNet dataset, we randomly sampled 800 images from the 312 official validation scenes for testing. KITTI [13] is a street-scene dataset with sparse metric depth captured by a LiDAR sensor.

4. Method

Our goal is to transform a text query q into a modified 3D scene s that is not only spatially coherent but also contextually rich before training a model on the generated images and text. This requires (a) identifying the correct spatial and contextual relationships between assets, (b) predicting a high fidelity and nice looking arrangement that aligns with these relationships, and (c) identifying user intent and accurately replicating that in-context of the visual arrangement of the scene. AirBlender performs this task by building on top of a state-of-the-art VLMs and a professional rendering software (Blender). We now describe the key components of our method, which builds upon SceneCraft [10]. Our unique contributions over SceneCraft here are (1) the use of a finetuned CLIP model on indoor scenes for asset retrieval, (2) a chained prompting system for input to modified scene generation, and (3) a hierarchical prompting framework to better capture user intent and distill it to low-level actions that can be coded via an LLM on-the-fly and executed via the Blender API.

4.1. Asset Retrieval and Scene Decomposition

A scene consists of a set of assets, where each asset is a 3D model. Given the input text query q, the agent makes an LLM (in this case, GPT-4) call to generate a list of asset names and description that shall be put in the scene. Based on them, a set of 3D assets are retrieved from a large repository of 3D objects utilizing a CLIP-based retriever. The retrieval process first finds the top assets based on the text description of each asset. Then each retrieved asset is rendered as an image and the one with the highest text-to-image score is selected.

Some scenes might contain up to a hundred assets, making the layout planning very difficult. The SceneCraft agent decomposes the scene into a set of sub-scenes, each representing a part of the entire scene. Breaking the problem into small pieces is a widely adopted strategy in natural language question answering and general reasoning [30].

The agent calls a LLM-empowered decomposer that breaks the input query into a sequence of sub-scenes $\hat{s}_k$, each containing a title, a list of asset names A_k and a sub-scene description q_k.

$$(q_1, A_1), \ldots, (q_K, A_K) \leftarrow \text{LLM-decomposer}(q). \quad (1)$$

4.2. Parsing User Intent into Editing Actions

To accurately understand vague user queries, we created a chained prompting and clarification system that gathers refined user clarification for the particular task and interprets actions based on user intentions for our text-based 3D scene modification interface. Based on scene graphs constructed like in SceneCraft, we develop an Object Action library that encodes different types of simple actions (add, remove) and complex actions (swap, replace, modify) with the aim of expanding this library. We distill compound queries into much simpler, fundamental actions that are easier to execute via a Blender Script. For example, the action of swapping can be interpreted as the sequence of the following simple instructions: remove object 1, remove object 2, add object 2 where object 1 was, add object 1 where object 2 was. Efforts to build a robust input evaluation system that understands user queries is a work in progress and will only get better with the progress in foundation models.

4.3. Hierarchical User Intent Parsing

Additionally, we use a hierarchical question-answering system to better understand user intention. For each query, we ask a few LLM-generated clarification questions to the user based on the query to better capture intent. These questions encourage the user to be more specific and detailed. Once that is done, we take the entire context of the user-module conversation and categorize it as one of several possible actions, along with identifying the target objects to be modified in the scene. Lastly, this is passed into a low-level action generation module that specifies exact location coordinates, rgb color values, degrees of rotation etc. that need to be executed via the Blender API.

AirBlender achieves a level of robustness where it deals with multiple compound queries for multiple objects in one prompt for a limited set of actions. We presently use a blender interface for recursive self-improvement and scene generation but this work extends to any 3D simulation environment, like Unity for virtual reality, where object actions can be codified and different 3D assets can be retrieved.

4.4. Monocular Depth Estimation

We test the effects of AirBlender on Monocular Depth Estimation performance, since we hypothesize that embedded information about depth captured in modified scenes guided by a user-system conversation will lead to a better understanding of scene depth during training and evaluation of vision models. Monocular depth estimation aims to transform a photographic image into a depth map, i.e.,

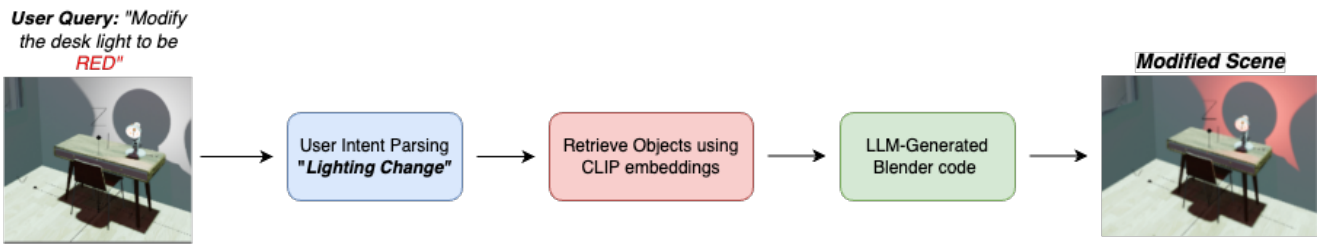

Figure 1: Modified Scene Generation Pipeline

regress a range value for every pixel. The task arises whenever the 3D scene structure is needed, and no direct range or stereo measurements are available. Clearly, undoing the projection from the 3D world to a 2D image is a geometrically ill-posed problem and can only be solved with the help of prior knowledge, such as typical object shapes and sizes, likely scene layouts, occlusion patterns, etc. In other words, monocular depth implicitly requires scene understanding. At the technical level, monocular depth estimation is a dense, structured regression task. Previous methods to get better 'in-the-wild' or out-of-distribution performance were functions of the diversity of the training dataset but Marigold shows transfer learning by leveraging commonsense priors stored in these large vision models, proving to be a good foundational model to test AirBlender's effectiveness on.

Marigold poses monocular depth estimation as a conditional denoising diffusion generation task. In the forward process, which starts at $d_0 := d$ from the conditional distribution, Gaussian noise is gradually added at levels $t \in \{1, \dots, T\}$ to obtain noisy samples d_t as

$$d_t = \sqrt{\alpha_t} d_0 + \sqrt{1 - \alpha_t}\epsilon$$

where $\epsilon \sim N(0, I)$, $\alpha_t := \prod_{s=1}^{t} 1 - \beta_s$, and $\{\beta_1, \dots, \beta_T\}$ is the variance schedule of a process with T steps. In the backward or reverse processs, the conditional denoising model $e_\theta(\cdot)$ parameterized with learned parameters θ gradually removes noise from d_t to obtain d_{t-1}.

At training time, parameters θ are updated by taking a data pair (x, d) from the training set, noising d with sampled noise ϵ at a random timestep t, computing the noise estimate $\hat{\epsilon} = e_\theta(d_t, x, t)$ and minimizing one of the denoising diffusion objective functions. The canonical standard noise objective $\mathcal{L}$ is given as follows:

$$\mathcal{L} = \mathbb{E}_{d_0, \epsilon \sim N(0,I), t \sim \mathcal{U}(T)} \left[\|\epsilon - \hat{\epsilon}\|^2 \right]$$

At inference time, $d := d_0$ is reconstructed starting from a normally-distributed variable d_T, by iteratively applying the learned denoiser $e_\theta(d_t, x, t)$.

Marigold utilizes a frozen Variational Autoencoder (VAE) to encode both the image and its corresponding depth map into a latent space, essential for training the conditional denoiser. Furthermore, the model incorporates a specialized normalization technique to achieve affine-invariance, which is crucial for maintaining accuracy in depth predictions. This method ensures that the depth map can be reconstructed with minimal error from the encoded latent code, demonstrated by the approximation $d \approx D(E(d))$. At inference, the model decodes the depth latent code once at the end of the diffusion process, and the average of the three channels is computed to produce the predicted depth map.

Marigold also uses a combination of multi-resolution noise and an annealed schedule to converge faster and substantially improve performance over the standard DDPM formulation. The multi-resolution noise is composed by superimposing several random Gaussian noise images of different scales, all upsampled to the U-Net input resolution. The proposed annealed schedule interpolates between the multi-resolution noise at $t = T$ and standard Gaussian noise at $t = 0$.

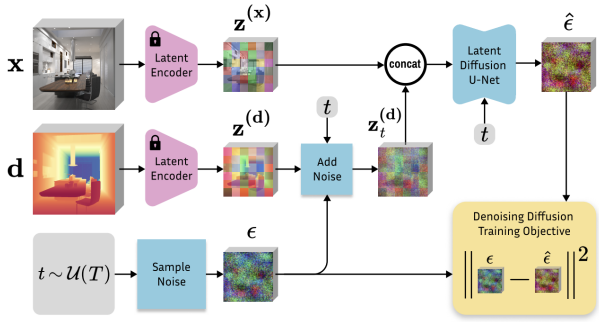

Figure 2: Marigold Model derived from Stable Diffusion-2

5. Experiments

To conduct a thorough evaluation of our system, we test the capabilities of the individual modules of our system which comprises hierarchical user intent parsing and asset retrieval modules before using generated data for monocular depth estimation using the Marigold model. We do this to develop a user-guided framework for data augmentation

and training for our module, such that even arbitrary user queries can be interpreted correctly and scenes can be modified accurately for training.

To evaluate the effectiveness of AirBlender in modifying 3D environments according to user-supplied prompts, we conducted several experiments that assess different components of the system. Here, we detail the metrics used for evaluation and the results achieved.

5.1. User Intent Parsing Accuracy

The ability of AirBlender to correctly parse user intentions from text queries was quantitatively measured. We utilized a set of 200 manually crafted queries reflecting various complexity levels, from simple asset additions to complex spatial arrangements. Each query was processed by AirBlender's parsing module, and the output editing actions were compared against manually generated ground truths. AirBlender achieved an intent parsing accuracy of 83%, indicating robust performance in understanding and converting natural language into actionable editing commands. Since we aren't able to conduct large scale human evaluation of the system present like in SceneCraft, we cannot compare this to the three qualitative metrics that measure the quality of images generated by SceneCraft but qualitatively that AirBlender achieves accuracy on par with SceneCraft and beats the BlenderGPT baseline.

5.2. Asset Retrieval Accuracy

The CLIP model, pre-trained on a wide array of images and textual descriptions, undergoes specialized fine-tuning tailored to the unique demands of 3D scene generation in the context of the AirBlender application. This fine-tuning procedure optimizes the model's weights to enhance its ability to associate specific 3D asset images with their corresponding textual descriptions effectively.

This enhanced model was evaluated for its precision in accurately retrieving 3D assets from the Objaverse in response to user queries. This was done using the CLIP-Frechet Inception Distance metric, which captures the quality of the distribution o f 3D Renders against a ground truth, which we supply. Inception distance scores based on a pre-trained visual representation can in principle capture the overall quality of the scenes, including natural placement and cohesiveness. To conduct this evaluation, we established a benchmark dataset comprising 500 specific queries, each paired with appropriately tagged assets. We compare our finetuned CLIP model against different CLIP retrievers on common indoor object retrieval and the results are shown in Table 1, where lower CLIP-FID scores are better.

The results highlight our system's capability to identify and prioritize the most relevant assets for 3D scene modifications effectively. Particularly, we see that our model is better able to capture large scenes that contain a variety of 3D Assets and is able to effectively generate scenes based on them, showing that finetuning CLIP on indoor scenes boosts its asset retrieval capabilties.

Room Type	Model	CLIP-FID(l)
Bedroom	ATISS	0.33
	CLIP-Layout	0.25
	CLIP-Finetuned	0.22
Dining room	ATISS	0.27
	CLIP-Layout	0.19
	CLIP-Finetuned	0.18
Library	ATISS	1.68
	CLIP-Layout	1.31
	CLIP-Finetuned	1.20
Living room	ATISS	0.19
	CLIP-Layout	0.13
	CLIP-Finetuned	0.10

Table 1: Metrics for one-step partial scene completion (Baselines derived from CLIP-Layout [14])

5.3. Marigold Performance

We implement a finetuning script on top of Marigold gold using PyTorch and utilize Stable Diffusion v2 [38] as our backbone, following the original pre-training setup with a v-object. The original Marigold pipeline has text-conditioning disabled, but we enable this for our purposes. We apply the DDPM noise scheduler with 1000 diffusion steps. At inference time, we apply the DDIM scheduler and only sample 50 steps. For the final prediction, we aggregate results from 10 inference runs with varying starting noise. Due to compute resources, training as part of our method takes 7K iterations (700 steps or epochs) using a batch size of 32 and 16 gradient accumulation steps. We use the Adam optimizer with a learning rate of 3e-5 and a mixed-batch sampler that prevents data overloading on our GPU. Additionally, we apply random horizontal flipping augmentation to the training data. Training using our method takes approximately 4 hours on a single Nvidia A100 GPU with 32 GB VRAM. The smooth loss curve obtained during training is shown in Figure 3 and provides evidence of our model learning the priors that make up scene modifications, which are tested further.

For evaluation with the datasets mentioned in Section 3, we follow the procedure highlighted in Marigold where, obeying the protocol of affine-invariant depth evaluation [20], we first align the estimated merged prediction m to the ground truth d with the least squares fitting. This step gives us the absolute aligned depth map $a = ms + t$ in the same units as the ground truth. Next, we apply two widely recognized metrics [20] [19] for assessing quality of depth estimation. The first is Absolute Mean Relative Error (*Ab-*

Figure 3: Loss Curve for Marigold on AirBlender generated Training Scenes

sRel), calculated as:

$$\text{AbsRel} = \frac{1}{M}\sum_{i=1}^{M}\left|\frac{a_i - d_i}{d_i}\right|,$$

where M is the total number of pixels. The second metric, δ_1 accuracy, measures the proportion of pixels satisfying

$$\max\left(\frac{a_i}{d_i}, \frac{d_i}{a_i}\right) < 1.25.$$

The results for Marigold finetuned on AirBlender data are shown in comparison to baselines in Table 2.

As Table 2 shows, Marigold trained with AirBlender generated scenes exhibits comparable performance to Marigold trained on Hypersim (which is a much larger dataset) and surpasses other baselines without going through as many training steps as other methods. We see that Marigold+AirBlender is consistently worse off than Marigold, both because of compute (Marigold was trained for 2 days) and data availability but close enough to make us believe that this method will only improve with scale. Marigold+AirBlender consistently beats other depth estimation methods, showcasing the potential of AirBlender as a method to better equip understanding in diffusion models for depth estimation. Despite being trained solely on synthetic depth datasets, the model can generalize to a wide range of real scenes, as shown by the performance on the test sets, all of which contain real data. This successful adaptation of diffusion-based image generation models toward depth estimation confirms our initial hypothesis that a comprehensive representation of the visual world is the cornerstone of monocular depth estimation. Moreover, it shows that AirBlender can allow the model to better capture spatial understanding in different 3D Scenes and understand nuances in depth previously uncaptured. It also shows that our fine-tuning protocol was successful in adapting Stable Diffusion for this task without unlearning such visual priors.

Figure 4: Out-of-Distribution Test Image of a person running through a road surrounded by a forest

5.4. Qualitative Evaluation

Beyond quantifying the capabilities of AirBlender, we feel it is important to qualitatively see the difference in scene understanding and generation that AirBlender equips Marigold with.

Figures 4 and 5 distinctly show the capabilities of AirBlender on Marigold. From Figure 5, we can see that the model preserves the overall layout of the scene and recognizes differences between objects that are closer (the dark orangish red road) than others (the light blue and yellow faraway trees) in the pictures, and applies a depth transformation accordingly. It also is not distracted by the lighting present in the scene, which, during our analysis, sometimes spoils the generated depth map. Despite being trained solely on synthetic depth datasets that contain indoor environments, the model can well generalize to a wide range of real scenes. Crucially, the depth map is clear evidence of spatial understanding equipped within Marigold, highlighting that the AirBlender method reinforces existing priors and equips new ones when understanding scenes.

Beyond spatial understanding and generation, the AirBlender user intent parsing and modified scene generation pipeline is not foolproof. While AirBlender is able to generate complex actions that require multi-step reasoning in a systematic manner, it is sometimes unable to generate specific low-level action for modification. At times, AirBlender struggles with what object the user is talking about, especially if there are multiple such objects in the scene and so it fails to differentiate one from the other. Even when it is able to, it misses out on distilling complex actions to simpler ones. For example, Figure 6 below shows a user-system conversation that shows that while the system identifies the

Method	NYUv2		KITTI		ScanNet	
	AbsRel	δ_1	**AbsRel**	δ_1	**AbsRel**	δ_1
DiverseDepth [27]	11.7	87.5	19.0	70.4	10.9	88.2
MiDas [28]	11.1	88.5	23.6	63.0	12.1	84.6
LeReS [20]	9.0	91.6	14.9	78.4	9.1	91.7
Omnidata [6]	7.4	94.5	14.9	83.5	7.5	93.6
HDN [29]	6.9	94.8	11.5	86.7	8.0	93.9
DPT [19]	9.8	90.3	10.0	90.1	8.2	93.4
Marigold (w/ ensemble)	5.5	96.4	9.9	90.6	6.4	95.1
Marigold + Airblender (w/ ensemble)	7.2	93.4	10.0	89.1	7.9	92.1

Table 2: Summary of Depth Estimation Metrics on Selected Datasets

Figure 5: Generated Depth Map by the Modified Marigold Model where darker colors indicate objects perceived by the model to be closer

User Query:

```
Color the door black
```

System Response:

```
Clarifying Question: Which door are
you referring to? Is it the one that is
closed in front of the bed or is it the
one that is ajar facing the window?
```

User Query:

```
The one that is ajar
```

System Response:

```
Object addition: black door
Adding a black door...
Solving for the layout...
Rendering the new scene...
```

Figure 6: Failure Case of User Query and System Response

need of a black door, it fails to realize that it needs to replace the existing door before adding a new one.

We mitigate this problem to some extent by providing a scene description to the LLM and believe that using more powerful vision language models would imbue stronger priors to ground the model on for better user clarification questioning and low-level action generation.

6. Conclusion

In this paper, we introduced AirBlender, an innovative framework designed for generation and modification of interactive 3D environments using text-driven inputs. By integrating the common sense priors of advanced diffusion models and extensive 3D assets, AirBlender provides a versatile tool that reduces the need for specialized 3D modeling expertise and accelerates the creation and customization of diverse 3D training environments for AI applications.

Throughout our research, we demonstrated that AirBlender's capabilities extend beyond traditional scene modification methods by incorporating user-supplied textual prompts to dynamically alter and enhance virtual environments. This is made possible through the fine-tuning of Marigold, our depth estimation model, on the AirBlender-generated data, which has shown remarkable adaptability and precision in handling complex scene modifications, even in challenging out-of-distribution settings.

Our experiments affirm the efficacy of AirBlender in parsing user intent with high accuracy and retrieving and arranging 3D assets to fulfill specified environmental contexts. These functionalities are critical in supporting the nuanced requirements of training scenarios for embodied AI agents, thereby contributing to more robust and adaptable AI systems. AirBlender can be adapted and applied to a range of fields from virtual reality and game development to embodied agent training and acts both as a form of data augmentation but also focused finetuning for end applications.

7. Future Work

As we continue to refine and enhance AirBlender, future work will focus on expanding its asset library, improving the system's ability to handle even more complex modifications, and enhancing its understanding of spatial relationships and user intents. Additionally, the effect of AirBlender on other kinds of models beyond Monocular Depth Estimation is yet to be seen as a test of generalization of this method.

References

[1] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017. 3

[2] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi. Robothor: An open simulation-to-real embodied ai platform, 2020. 1

[3] M. Deitke, R. Hendrix, L. Weihs, A. Farhadi, K. Ehsani, and A. Kembhavi. Phone2proc: Bringing robust robots into our chaotic world, 2022. 1, 2

[4] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects, 2022. 1

[5] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi. Procthor: Large-scale embodied ai using procedural generation, 2022. 1, 2

[6] A. Eftekhar, A. Sax, R. Bachmann, J. Malik, and A. Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans, 2021. 7

[7] H. Fu, B. Cai, L. Gao, L. Zhang, J. W. C. Li, Z. Xun, C. Sun, R. Jia, B. Zhao, and H. Zhang. 3d-front: 3d furnished rooms with layouts and semantics, 2021. 2

[8] H. Fu, Z. Yang, M. Wang, and M. Chen. Unveil conditional diffusion models with classifier-free guidance: A sharp statistical theory, 2024. 2

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020. 2

[10] Z. Hu, A. Iscen, A. Jain, T. Kipf, Y. Yue, D. A. Ross, C. Schmid, and A. Fathi. Scenecraft: An llm agent for synthesizing 3d scene as blender code, 2024. 3

[11] D. Ignatov, A. Ignatov, and R. Timofte. Virtually enriched nyu depth v2 dataset for monocular depth estimation: Do we need artificial augmentation?, 2024. 3

[12] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler. Repurposing diffusion-based image generators for monocular depth estimation, 2024. 1

[13] Y. Liao, J. Xie, and A. Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d, 2022. 3

[14] J. Liu, W. Xiong, I. Jones, Y. Nie, A. Gupta, and B. Oğuz. Clip-layout: Style-consistent indoor scene synthesis with semantic furniture embedding, 2023. 5

[15] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 2

[16] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014. 2

[17] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. Virtualhome: Simulating household activities via programs, 2018. 2

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021. 1, 2

[19] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction, 2021. 5, 7

[20] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020. 5, 7

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2022. 1, 2

[22] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022. 2

[23] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models, 2022. 2

[24] Y. Wang, J. Liao, H. Yu, G. Wang, X. Zhang, and L. Liu. Advanced conditional variational autoencoders (a-cvae): Towards interpreting open-domain conversation generation via disentangling latent feature representation, 2022. 2

[25] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation, 2023. 2

[26] Y. Yang, F.-Y. Sun, L. Weihs, E. VanderBilt, A. Herrasti, W. Han, J. Wu, N. Haber, R. Krishna, L. Liu, C. Callison-Burch, M. Yatskar, A. Kembhavi, and C. Clark. Holodeck: Language guided generation of 3d embodied ai environments, 2024. 2

[27] W. Yin, X. Wang, C. Shen, Y. Liu, Z. Tian, S. Xu, C. Sun, and D. Renyin. Diversedepth: Affine-invariant depth prediction using diverse data, 2020. 7

[28] W. Yin, J. Zhang, O. Wang, S. Niklaus, L. Mai, S. Chen, and C. Shen. Learning to recover 3d scene shape from a single image, 2020. 7

[29] C. Zhang, W. Yin, Z. Wang, G. Yu, B. Fu, and C. Shen. Hierarchical normalization for robust monocular depth estimation, 2022. 7

[30] P. Zhou, J. Pujara, X. Ren, X. Chen, H.-T. Cheng, Q. V. Le, E. H. Chi, D. Zhou, S. Mishra, and H. S. Zheng. Self-discover: Large language models self-compose reasoning structures, 2024. 3