Leveraging Stable Diffusion for 3D Environment Generation

Aditya Angajala, Harshal Bharatia, Krish Agarwal

Abstract— **Because traditional methods of generating 3D models require extensive manual effort, we propose a novel methodology of generating 3D environments using Stable Diffusion. We propose several approaches to encode and decode 3D environments, including using run-length encodings, Hilbert curves, multi-layer slices, and topological height maps. We further propose utilizing various generative technologies, including individual stable diffusion models, an AnimateDiff pipeline, and a recurrent variant of regular stable diffusion. Visual analysis suggests that the diffusion model can reliably replicate the structure of the environment, although it can be less successful at absolutely following the given prompt. This makes it useful for automatically generating novel 3D environments.**

I. Introduction

Traditional methods for creating 3D worlds and environments that adhere to certain specifications require extensive manual effort. This process is time-consuming and limits the scale and variety of environments that can be produced. Additionally, there exists insufficient labeled 3D data to train a foundational generative model from scratch to perform this task. Since 3D environments are prevalent in many games in simulations, quickly creating robust 3D environments is critical. Recent advancements in AI tools like diffusion models have shown great promise for the generation of a wide variety of unique 2D images that correctly follow prompting [1]. There has also been work in converting these generated 2D images into 3D object representations [2]. However, progress in 3D environment generation has been limited in comparison. This is because representing a 3D environment can be difficult, as it not only requires simultaneously representing multiple perspectives but also internal structures and materials that cannot be easily seen by the user. Additionally, representing a dense 3D structure on a 2D plane, especially in a way replicable by diffusion, can be challenging. An alternative is to allow a recursive method of generation, which iteratively generates one 2D plane of images at a time. However, generation across planes must be smooth, thus requiring a new generative approach.

In this work, we propose a novel approach to procedurally generate 3D environments utilizing the capabilities of models like Stable Diffusion to allow for the use of natural language prompts. The key idea is to encode 3D information from scenes into 2D image representations that can be effectively modeled by the diffusion model. The model is then conditioned on textual prompts to generate specific encoded 2D representations which can then be decoded back into full 3D environments.

This approach could enable customizable procedural generation of 3D worlds with much lower human effort which would prove useful in areas such as gaming, VR, and simulations.

II. Related Work

Diffusion is a generative technology that uses iterative denoising steps to generate new images. At each step, a U-Net model is fed a blank prompt embedding as well as the embedding for the provided prompt to generate two images denoised from the output of the previous step. One of these images is meant to be generative while the other is meant to be prompt-driven. These images are combined to produce the final output of a denoising iteration [1]. While diffusion works in pixel-space of images, stable diffusion first downscales images into a latent-space representation to perform denoising steps on, which it can upscale at the end for the final output. Upscaling and downscaling between pixel and latent-space representations is done through a variational autoencoder (VAE). This approach makes both training and inference much faster in comparison to regular diffusion [3].

DreamBooth allows fine-tuning a diffusion model by representing a unique subject or style with a unique textual token. These tokens can be associated with new images or styles, and with a small number of sample images, it allows representing objects in new environments while maintaining the fidelity of the original object. The representation of new styles with a specific token can be used to represent different training styles and encodings. [4]

AnimateDiff is a method for using any stable diffusion model to generate videos by embedding an attention-based motion module between U-Net iterations that incorporates the cross-attention between different image frames for continuous motion. This method is independent of the model used, and any user can train or fine-tune a stable diffusion model to use with AnimatedDiff. While intended for video, the ability for AnimatedDiff to smoothly transition between video frames indicates its potential applications for effectively transitioning between images that represent cross-sectional layers in a 3D environment [5].

A related task is to have diffusion models generate 2D visualizations of 3D models, which is a widely researched topic. DreamFusion uses a loss based on probability density distillation to optimize a diffusion model for generating consistent 2D views of NeRF models that represent 3D objects [6]. Magic3D furthers this research by optimizing the generation speed and model resolution using a two-stage framework consisting of a 3D neural representation with a textured representation model [2]. Score Jacobian Chaining renders 3D models by chaining the gradients of

a diffusion model and backpropagating through a Jacobian matrix to derive a 3D score [7]. Although these methods effectively convert text input to 3D model visualizations, they are strictly limited to these visualizations and cannot be used for extracting information about an actual 3D environment.

Cano et al. introduce a method for the procedural generation of underground tunnel environments for robot simulation [8]. However, this is limited by requiring a predefined specification of the environments, and it is not a generative method.

There has also been recent work in using diffusion models to generate novel outputs in non-image domains. Riffusion transcodes an audio clip as a 2D image representing frequency and time. Given a prompt of what music to generate, it uses Stable Diffusion to generate new image transcodings and convert them into music. Riffusion's methodology for training a Stable Diffusion model to generate image encodings of non-image tasks demonstrates that Stable Diffusion can be extended to new domains by encoding a variety of tasks in an intermediate image format [9].

Sándor et al. demonstrate a method of encoding 3D models of houses into single 2D images and decoding them back. They also attempt to use their decoded images to train an AI model to create new houses. However, the relatively small dataset, detailed design, and use of StyleGAN instead of Stable Diffusion led to poor decoded models [10].

Space filling curves allow representing a multi-dimensional space by passing through all points of that space. Nested iteration curves iteratively traverse one axis at a time, skipping from the last element on one row to the first element on the next row once. This leads to instances of large space between consecutive elements in the encoding. Z curves follow a z pattern for more locality but still have jumps in the encoding. Hilbert Curves are space filling curves which traverse every point in higher dimensional spaces while preserving spatial locality. This has led to their use in voxel compression [11].

Perlin noise is a method for procedurally generating gradient noise, both in 2D and 3D space [12]. It is commonly used for generating terrain patterns and can be a useful starting point for creating artifical 3D model data.

III. Method

The general method for this project is shown in figure 1. A 3D environment is encoded into one or multiple 2D images, which along with a text prompt is used to train a diffusion model. This could either be a one-shot generation model or a recurrent generation model. Given a prompt, the model generates an embedding that is decoded into another 3D environment that matches that prompt.

A. Generation of Abstract Features

To prove Stable Diffusion's capabilities in generating distinct value classes and generating features given natural text, we explore the generation of topological height maps with discrete height classes that contain distinct features such as "hills" or "basins" with randomized parameters. Success here is indicated by correctly generating encodings which contain features that adhere to the prompts given, specifically since that would indicate that the discrete height classes are properly recognized.

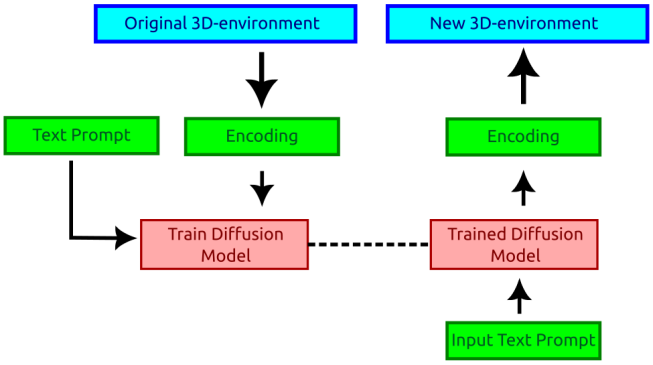


Fig. 1: Environment generation overview

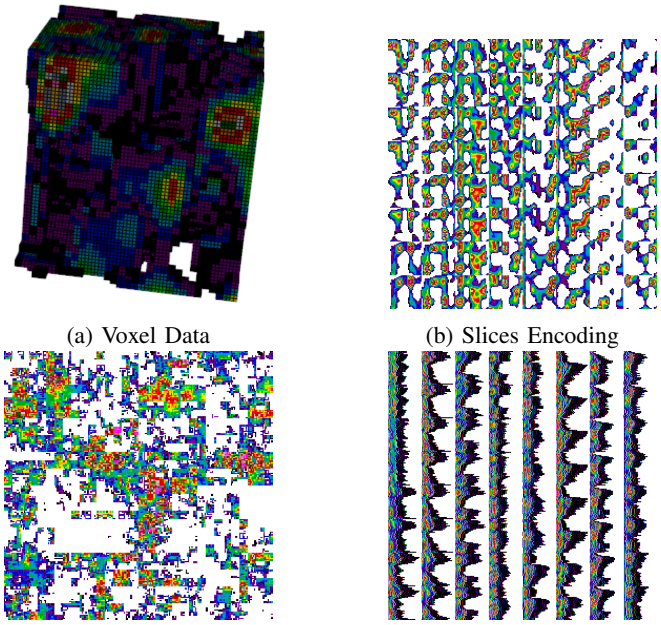
B. Encoding and Decoding 3D Environments

As height maps are inherently unable to capture more complex 3D environments where materials can vary independent of topology, we propose three additional encoding techniques that enable the encoding of true 3D data: Run-Length Encoding, 3D Hilbert Curves compressed to a 2D plane, and layer-by-layer separation. These are shown in figure 2

Run-Length Encoding: The voxel data of a 3D model is compressed column-wise using run-length encoding. Each z-axis column becomes a row in the output image, with material changes represented by pixel pairs. The first pixel encodes the material type, and the second encodes the run length of that material in the original voxel column. Since run lengths exceed the number of distinguishable colors, lengths are binned into ranges with each bin being assigned a color. As the majority of run length values are relatively low, earlier bins, which represent lower values, are given smaller ranges to increase precision. The two rows (material and length) can be plotted separately as two rows or stacked onto one row. We find the stacked variation to result in higher quality outputs.

Hilbert Curve: This encoding is created by plotting the path of the 3D Hilbert curve as a 2D Hilbert curve. The voxel data is initially traversed along the path of a 3D Hilbert curve to get a one dimensional list of the data. This list is then iterated through and plotted along the path of a 2D Hilbert curve. The benefit of this approach is that voxels that appear spatially local in the 3D data remain so in the resulting encoding. However, the resulting encodings are not as visually coherent as other encoding strategies, which may make this encoding format more difficult to train into a stable diffusion model.

Layer by Layer Slices: A cross section of a 3D model is taken along each (x, y) plane and then plotted adjacent to each other in a grid format on a 2D image. Each pixel in a



(a) Voxel Data (b) Slices Encoding

(c) Hilbert Encoding (d) Run-Length Encoding

Fig. 2: Appearance of encoding schemes on the same voxel data

slice directly represents the corresponding material at each (x, y) location for its given height.

C. Artifact Reduction

To reduce artifacting in the decoded models, we upscale the voxel data or resulting encoding. This allows the averaging of the generated pixels during the decoding process to diminish the impact of any outlier pixels and increase visual coherence of the encodings to make it more likely that Stable Diffusion produces viable results.

Additionally, to allow for tolerance in color, each pixel in a generated encoding is rounded to the nearest color from the predetermined set of colors by using the minimum Euclidean distance, where each colors is represented as a 3D vector in the RGB color space.

D. Model Pipeline

We propose three methods for using stable diffusion with the described 3D model encodings: a standalone stable diffusion model, a recurrent stable diffusion model, and a model fed into an AnimatedDiff pipeline.

Single Diffusion Model: We use a single stable diffusion model to generate either of the encodings mentioned earlier. To accomplish this, we primarily use DreamBooth, which allows greater personalization of the fine-tuning process and also uses less VRAM. During training, we optimize the model to take a text caption describing a 3D model and generate an output that closely matches our ground-truth encoding. This approach works for encodings represented in single images but is inflexible to encodings that require multiple images to represent a 3D model.

AnimateDiff: We train a stable diffusion model to generate single layer-by-layer cross-sectional images of a 3D environment, with a separate prompt style per set of layers to allow the model to differentiate between them. We then use this model within an AnimateDiff pipeline: the model is given several prompt checkpoints for the generation, and the pipeline as a result generates continuous ”frames”, where we interpret these frames as continuous cross-sectional images in a 3D model. The model we feed into this AnimatedDiff pipeline is first trained on different prompt styles for predetermined range of layers. Here, a prompt style simply refers to a special token associated with each range of layers that are used to help stable diffusion learn multiple concepts at once. When generating output through AnimatedDiff, the input text must be formatted using prompt travel to ensure that each set of layers follows the required text prompt. This ensures smooth changes between multiple layers and allows for flexible granularity of generating continuous cross-sectional layers. In addition to training a single model, it becomes feasible to train a sequence of encodings, which are more lightweight. However, they may not be as accurate as training with a single model.

Recursive Model Architecture: We implement a novel, yet relatively simple, recurrent architecture for using stable diffusion to sequentially generate layer-by-layer cross-sectional images of a 3D environment. The existing stable diffusion architecture is primarily composed of a set number of U-Net iterations. At each iteration, the U-Net is given a text embedding of the original prompt to perform prompt-driven denoising. Our architecture places an LSTM at the start of the pipeline to capture layer-by-layer dependencies between cross-sectional images. For each U-Net iteration, we first linearly transform the latent-space representation from the previous denoising step into a more compact vector to feed into the LSTM. The LSTM will produce an output vector and a hidden state representation. The output vector is linearly transformed with the original text embedding to produce a new embedding that is aware of dependencies between layers in a terrain generation. This new embedding is fed into the U-Net to denoise a sample in place of the regular text embedding in the case of regular stable diffusion. Figure 3 provides visualization of this model architecture.

At inference time, there are two sequential processes occurring: iterative denoising as with regular stable diffusion and recurrent layer-by-layer generation. We maintain a separate hidden state representation per denoising step. For generating the first layer in the output terrain, none of the denoising steps has any information about previous layer generations, so the embeddings produced by the LSTM will contain no previous context. However, the LSTM will produce a hidden state (per denoising step) that becomes relevant for subsequent layer generations. The last denoising step per layer generation produces the final cross-sectional image per layer in the terrain. Figure 4 provides a simplified view of the terrain generation process with this model architecture.

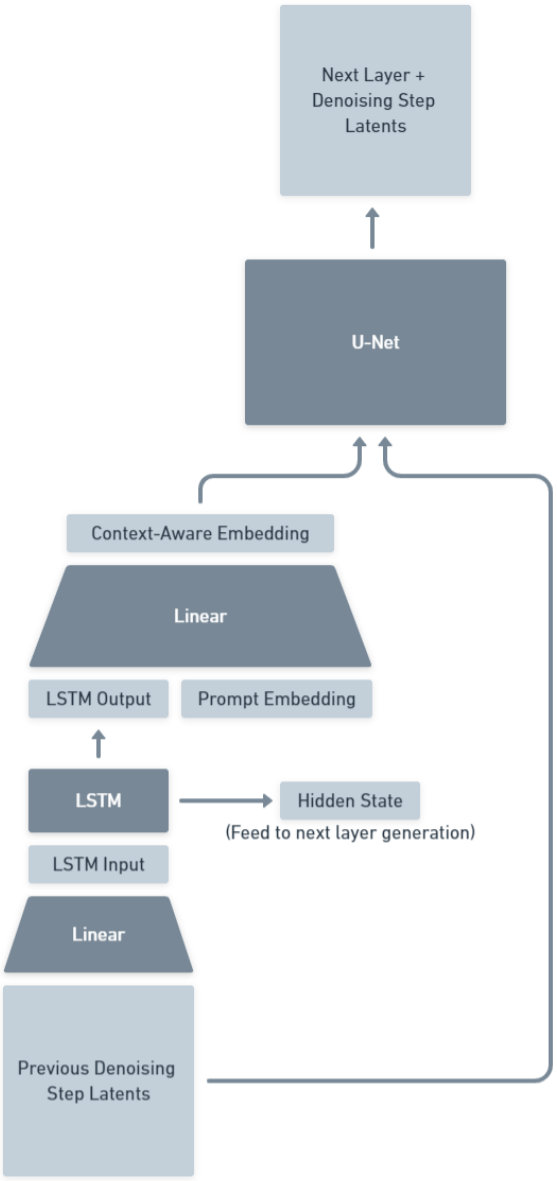


Fig. 3: Recurrent Model Architecture

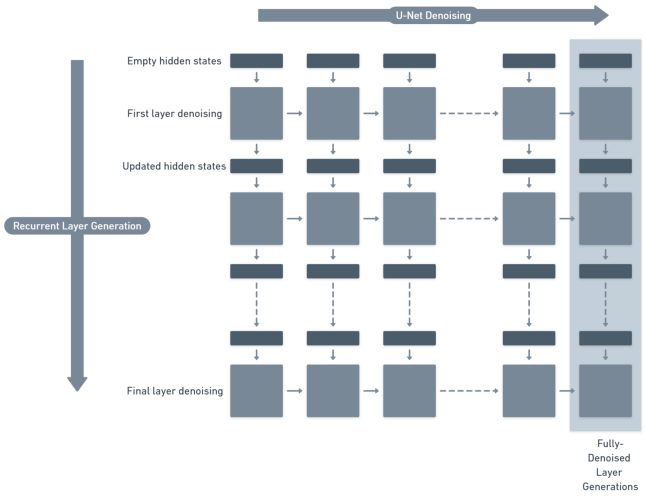


Fig. 4: Recurrent Environment Generation

IV. Evaluation

For our different encoding schemes, we generate artificial 3D models for training our stable diffusion models.

A. 2D Topological map encodings

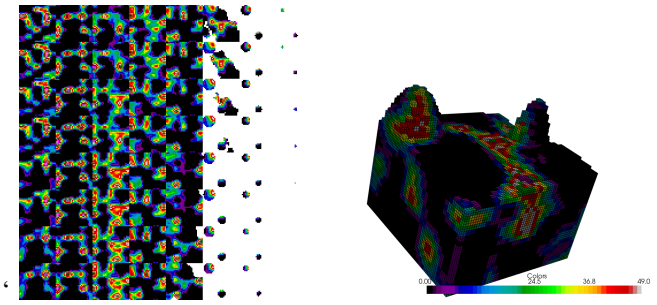
Every unique (x, y) coordinate is assigned a height value that is represented by the color of the pixel at that location in the 2D image encoding. The colors are chosen upfront to discretely represent each height class. For assigning height values to (x, y) coordinates, we sample 2D Perlin noise for a base terrain and then randomly select a set of hills and basins to place in specified locations within the environment (there is still some randomness in terms of the exact width, height/depth, and placement of these features). We then caption the resulting encoding with information about its generation to create pairs of natural language prompts and ground-truth encodings to train stable diffusion. Using Dreambooth, we trained a model given the topological encodings and corresponding captions. Then, we tested the efficacy of the diffusion model for various prompts that were similar, but not the same, as the training data. The diffusion model output was decoded into the corresponding 3D visualization. We evaluated the degree of similarity between the resulting image and the original prompt.

B. 3D encodings

We tested the system for both randomly-generated Perlin noise and a 3D terrain map representation. This was done due to the difficulty in automatically generating artificial 3D models of real objects and a wide range of options regarding the prompting of the model.

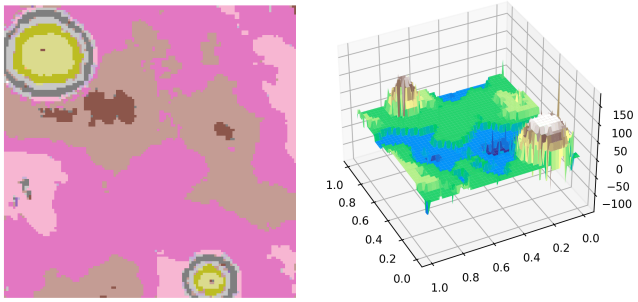
For the Perlin noise representation shown in figure 2, we generate our 3D training data by sampling 3D Perlin noise. The noise provides a color for each voxel, which we round to the nearest color from a predetermined set of colors. Each color from this set is meant to represent a different material. Although the generated models are a form of noise, it is important to note that this noise is coherent, meaning that the voxels are not entirely randomized and rather appear in a gradient fashion, as is the nature of Perlin noise. For a stable diffusion model to generate an encoding that represents similar coherent 3D noise would mean that the model is learning appropriate material distributions and is accurately generating locally coherent noise gradients.

We also generate a set of 3D terrain data as shown in figure 5, which has similar hills and basins as the 2D topological map data with prompts specifying feature attributes. More specifically, we had three for both width and height. The width descriptors were ”wide”, ”normal”, and ”narrow”. The height descriptors include ”tall”, ”normal”, and ”short” and ”deep”, ”normal”, and ”shallow” for hills and basins respectively. However, in comparison to our 2D topological maps, this 3D terrain data contains richer material distri-



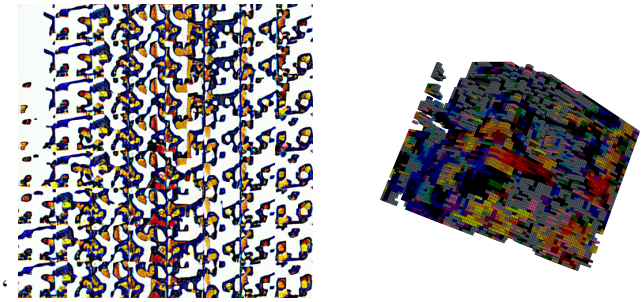
(a) 3D terrain encoding (b) 3d terrain decoding

Fig. 5: 3D terrain prompted slices



(a) Generated topological map (b) 3D visualization

Fig. 6: Stable Diffusion Topological Map output



(a) Generated layer encoding (b) 3D voxel decoding

Fig. 7: Stable Diffusion Layer Separation output

butions rather than a set material per column. Materials were generated with random Perlin noise, similar to that of the Perlin noise representation. This allows the user to request specific prompting and evaluate the efficacy of the model to recreate the user's specific prompts. We encode this 3D terrain data using either a Hilbert curve, a run-length encoding, or a layer-by-layer slices encoding for training into and generating environments through a stable diffusion model.

Lastly, we encode the 3D terrain data as multiple cross-sectional images that we use within our two recurrent pipelines. A stable diffusion model was trained for each range of layers getting a unique token. This model was fed into an AnimateDiff pipeline, in which we specified prompt checkpoints using these unique tokens to identify each range of layers, with which AnimateDiff could perform prompt travel for smooth transitions. We also fed this data into our custom recurrent architecture, where we train the pipeline end-to-end to sequentially generate cross-sectional images.

C. Evaluation

Each model was trained using a sequence of images for different environments, as well as their corresponding prompts. The resulting encodings were decoded into a 3D environment and evaluated against the original models. Each generated environment was compared to the ground-truth environment for overall structure, feature similarity, similarity of material distribution, and prompt adherence. The overall structure included the general appearance of the encoding and matching the style of the original encodings. Feature similarity included the shape of the different features (hills and basins) as well as the similarity of edges, lack of miscellaneous blocks, and unevenness of the figures. Material distribution included the similarity of materials between the original and generated encoding. Prompt similarity was the degree to which the model matched the prompt in number, size, type, and location of features.

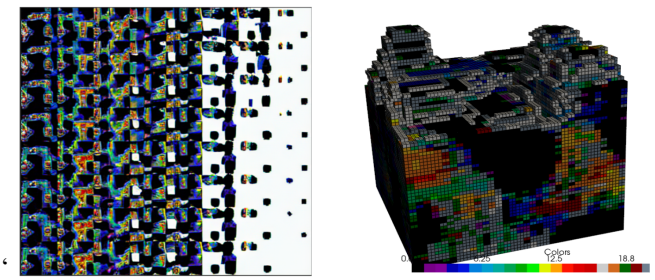
V. Results

A. Generation of Abstract Features

As shown in Figure 6, the model trained on 2D topological maps is able to accurately recognize discrete height classes and generate coherent features such as hills and basins from non-gradient topology coloring. The resulting encoding had a similar structure as the original topological map and was able to smoothly represent different hills and basins. This was the case for both discrete height classes and continuous height levels. Although it was successful at responding to general features, our model does not necessarily follow our prompting regarding the absolute location of the features. This could be because diffusion is unable to represent the absolute specifications appropriately and, as a result, produces a greater or lesser number of features than expected.

B. Encoding and Decoding 3D Environments

As shown in Figure 7, the layer-by-layer encodings generated by our model appear visually consistent between layers. This was the case for both 3D Perlin noise and 3D terrain. For Perlin noise environments, the resulting 3D models



(a) Prompted slices encoding (b) 3d topological decoding

Fig. 8: 3D topology prompted slices

demonstrate that our stable diffusion model is able to generate encodings that accurately capture local gradients since materials appear in clusters rather than as completely random noise. This suggests that a layer-by-layer encoding is able to effectively capture the material distribution. Additionally, the 3D terrain generations effectively replicate the overall shape of the hills and basins, along with the corresponding material distribution.

The layer-by-layer encoding also seemed to effectively respond to prompts, both in type and size of feature. For example, figure 8 was generated using the prompt, *"Tall wide hill, shallow-narrow basin"* and visually represents the original prompt. This suggests that the model is able to accurately associate complex figures in the encoding with the specific text prompt.

As shown in Figure 10, the Hilbert encoding was somewhat noisy and unable to effectively capture the material distribution. This could be in part due to the lack of visual coherence in the encodings. Stable diffusion generally performs better on recreating visually coherent data such as shapes, objects, and structures as opposed to less structured pixels like the Hilbert encoding. However, the Hilbert encoding was still visually coherent on the presence of voxel groups, including for both small and large clusters or gaps in the environment. This visual coherence is apparent from the fact that the presence/non-presence of voxels is not scattered.

Similar to the Hilbert encoding, the run length encodings were also noisy and unable to capture material distributions. This could be due to a loss of information due to the compression of the encoding into latent space, as well as the need to use two separate columns to represent a sequence of blocks. However, there was still visual coherence in the presence of blocks, just like in the Hilbert encoding. This seems to indicate that these distributions could be effective for environments with a low number of material states.

The cross-sectional images generated through the AnimateDiff pipeline were smooth and cohesive on the environment structure and materials. However, they were unable to replicate critical shapes and material distributions, such as the tapered shape of a hill. This is because the model seemed unable to differentiate between different sections of the environment, despite the presence of unique tokens for each section of layers. As a result, it was also unable to respond to the prompt of a specific section. Representing different sections as individual encodings, as opposed to a single model, resulted in worse results as it was unable to reliably replicate the overall structure of each encoding. However, the smoothness of the model is promising to try with alternative models.

The cross-sectional images generated through our recurrent model were visually consistent with the cross-sectional images in the training data. Specifically, it was apparent that the cross-sectional images encode specific features of the intended 3D environment. For example, the cross-sectional image in Figure 9 shows both gradient noise similar to that which was part of our 3D terrain data as well as circular features representing either a hill or a basin. However, our

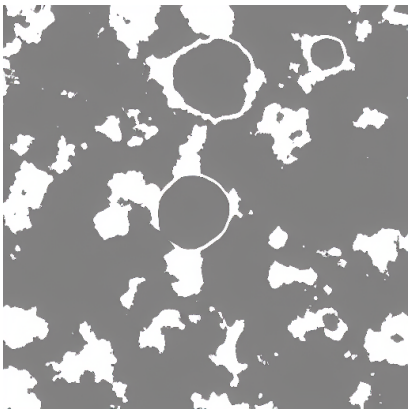
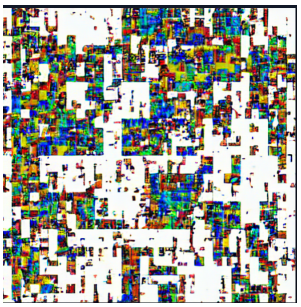
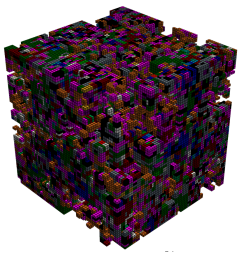


Fig. 9: Cross-Section Generated with Recurrent Model



(a) Generated Hilbert encoding



(b) 3D voxel decoding

Fig. 10: Stable Diffusion Hilbert encoding output

recurrent model was unsuccessful at maintaining coherence between sequential cross-sectional images. This is likely due to our resource limitations that prevented us from training the model over long-range dependencies, and this suggests the need for optimizing the recurrent training process to be less compute-intensive.

VI. Discussion

Overall, the encodings generated by the system resemble the encodings we trained our models on. For the 2D topological maps, the fact that we can accurately generate features from distinct height classes suggests that the model is able to produce discrete encodings and largely follow a prompt. However, a limitation of the system is its inability to accurately generate images with absolute positioning of features.

The ability for the system to represent 3D encodings through the one-shot slices method by responding to particular prompts is also promising, as it shows that an environment can be generated on-demand based on the user's preferences. Although other one-shot methods such as Hilbert and run-length encodings were not as effective due to lower visual coherence, they could still be promising for environments with a low number of material states. The AnimateDiff and recursive pipelines are also promising in their cross-sectional generations, and further work should be done to remedy their issues with layer separation.

In general, we also believe that output quality, in regards to artifacting, can be increased in the future through more effective fine-tuning of the model and/or a more elaborate decoding process. Results from models such as Realistic Vision and Dreamshaper indicate models can be trained to output much more detailed images than the relatively blurry results generated. Additionally, artifacting like the gray blocks surrounding features are the result of feathering between empty space and the terrain in the encoding and could be removed through image post-processing.

Overall, these findings suggest that 3D encodings are feasible to generate using stable diffusion, even for more complicated environments.

VII. Conclusion

In order to make generating 3D models easier, we developed a novel technique using Stable Diffusion. We trained stable diffusion pipelines to generate 2D encodings of 3D environments, whose output at inference-time could then be decoded into an actual 3D environment. Using topological height maps, we showed that stable diffusion is capable of generating distinct features. We proposed various encoding schemes, including run-length encodings, Hilbert curves, and layer-by-layer slices. Additionally, we employed different diffusion pipelines including single stable diffusion models, a custom recurrent model architecture, and an AnimateDiff pipeline. After testing the model for 2D topological maps, material-dense 3D gradient noise, and material-dense 3D terrain, we determined that stable diffusion is capable of successfully generating 2D encodings of various environments through various encoding schemes. It is able to perform best with a slice encoding and a single stable diffusion pipeline, especially in terms of responding to user prompts.

Such a system can be used in the rapidly growing areas of gaming and VR, as it could be used to more efficiently and interactively create a variety of diverse environments to navigate. This could be done easier and faster than manual and heuristic-based environment generation methods popular today. This system could be expanded to allow the user to provide more abstract goals, and have a reasoning engine or large language model that can determine how to best create the environment to create those goals. Additionally, further work can be done to allow more absolute positioning in the system to allow the user to further customize the environment. We are motivated in future work to experiment further with different models and diffusion pipelines to further improve our results.

VIII. Acknowledgement

We would like to thank Dr. Justin Hart for guiding us through the research process and for giving us the necessary resources to do this project. We would like to also thank Yingying Xu for guiding us throughout this process.

References

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

[2] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 300–309.

[3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[4] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," 2023.

[5] Y. Guo, C. Yang, A. Rao, Y. Wang, Y. Qiao, D. Lin, and B. Dai, "Animatediff: Animate your personalized text-to-image diffusion models without specific tuning," 2023.

[6] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," 2022.

[7] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich, "Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation," 2022.

[8] L. Cano, D. Tardioli, and A. R. Mosteo, "Procedural generation of underground environments for gazebo," in *ROBOT2022: Fifth Iberian Robotics Conference*, D. Tardioli, V. Matellán, G. Heredia, M. F. Silva, and L. Marques, Eds. Cham: Springer International Publishing, 2023, pp. 314–324.

[9] S. Forsgren and H. Martiros, "Riffusion - Stable diffusion for real-time music generation," 2022. [Online]. Available: https://riffusion.com/about

[10] V. Sándor, M. Bank, K. Schinegger, and S. Rutzinger, "Collapsing complexities: Encoding multidimensional architecture models into images," in *Hybrid Intelligence*, P. F. Yuan, H. Chai, C. Yan, K. Li, and T. Sun, Eds. Singapore: Springer Nature Singapore, 2023, pp. 371–381.

[11] "Voxel compression." [Online]. Available: https://eisenwave.github.io/voxel-compression-docs/rle/space_filling_curves.html

[12] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, 1985, p. 287–296. [Online]. Available: https://doi.org/10.1145/325334.325247