TerraFusion: Joint Generation of Terrain Geometry and Texture Using Latent Diffusion Models

Kazuki Higo Toshiki Kanai Yuki Endo Yoshihiro Kanamori
University of Tsukuba

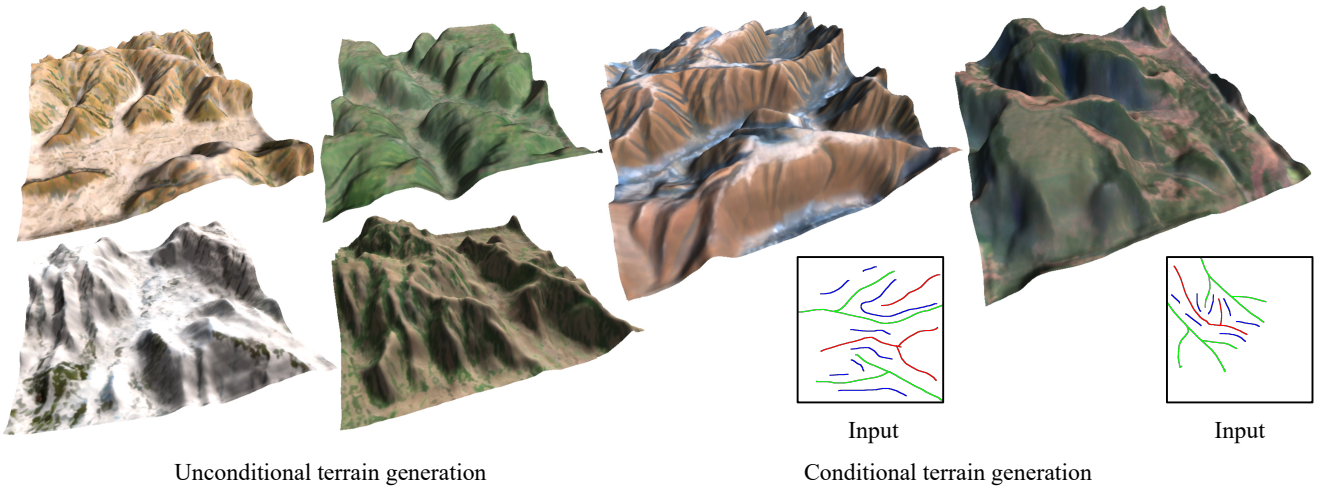

Figure 1. We introduce *TerraFusion*, a novel diffusion-based framework for jointly generating terrain geometry and textures. Our method not only enables the random synthesis of plausible terrain models (left), but also supports intuitive user control through rough sketches (right), where valleys, ridgelines, and cliffs are indicated by red, green, and blue lines, respectively.

Abstract

3D terrain models are essential in fields such as video game development and film production. Since surface color often correlates with terrain geometry, capturing this relationship is crucial to achieving realism. However, most existing methods generate either a heightmap or a texture, without sufficiently accounting for the inherent correlation. In this paper, we propose a method that jointly generates terrain heightmaps and textures using a latent diffusion model. First, we train the model in an unsupervised manner to randomly generate paired heightmaps and textures. Then, we perform supervised learning of an external adapter to enable user control via hand-drawn sketches. Experiments show that our approach allows intuitive terrain generation while preserving the correlation between heightmaps and textures.

1. Introduction

3D terrain models play a crucial role in creating realistic landscapes and enhancing user experiences in various applications, including video games and film production. However, manually designing realistic terrain is challenging due to the inherent correlation between terrain geometry and texture. For example, steep regions often exhibit rocky or forested textures, with color and pattern variations aligned with the terrain's contours.

Numerous methods have been proposed for automatic terrain generation [5]. Recent advances in deep learning have particularly enabled efficient and high-quality terrain generation [7, 11, 16, 18, 26]. These approaches typically represent terrain geometry using heightmaps that store elevation values per pixel, framing terrain generation as an image generation task. Examples include generating heightmaps using Generative Adversarial Networks (GANs) [7, 26] and diffusion models [16]. There is also a method that gen-

erates textures with seasonal variation conditioned on heightmaps [11]. These existing methods can synthesize either heightmaps or textures. However, consistent synthesis and editing of both heightmaps and textures are not straightforward by combining these methods.

One possible solution is a two-stage approach that combines a heightmap generator with a texture generator. In this setup, the first stage generates a heightmap, and the second stage generates a texture conditioned on it, and vice versa. However, such two-stage approaches are prone to error accumulation across stages, which can degrade output quality. Another challenge is to preserve the correlation between the heightmap and texture during user editing. For instance, if a user modifies the second-stage output via a sketch, the correlation with the first-stage output may be lost.

To address these issues, we propose *TerraFusion*, a direct approach that models the joint distribution of heightmaps and textures (see Figure 1). Specifically, we employ a Latent Diffusion Model (LDM) [20] to generate both components simultaneously. We extend an LDM to handle paired heightmaps and textures, and train the model in an unsupervised manner using real-world terrain data. To reduce computational cost, the diffusion model operates in a low-dimensional latent space learned by a Variational Autoencoder (VAE). We train a VAE dedicated to heightmaps to ensure accurate latent representation. To improve output quality, we fine-tune the model using prior knowledge from Stable Diffusion [20], which was pre-trained on large-scale image datasets. This joint approach avoids error accumulation inherent in sequential models and preserves the correlation between geometry and color during user editing. Prior work has shown that diffusion models are applicable to various image editing tasks [9]; as one such application, we incorporate a conditioning module into the LDM to enable user control via sketch input.

Our key contributions are summarized as follows:

1. **Joint generation framework**: We propose TerraFusion, a novel framework based on an LDM that simultaneously generates terrain heightmaps and textures in a fused latent space.
2. **Domain-specific VAE**: We train a VAE dedicated to heightmaps, achieving more accurate reconstructions than generic image-based VAEs.
3. **User-guided control**: We incorporate a conditioning module into our architecture, enabling intuitive and flexible terrain design from input sketches.

Compared to baselines, our approach produces better qualitative and quantitative results and more accurately captures the correlation between heightmaps and textures. Project page: https://millennium-nova.github.io/terra-fusion-page/.

2. Related Work

Geometry generation. Classical terrain generation methods include procedural and physics-based approaches [6]. Procedural methods generate terrain geometry automatically using random noise and fractals, while physics-based methods simulate natural processes such as erosion and sedimentation. However, both approaches typically require expert knowledge and iterative parameter tuning.

To make terrain generation more accessible, deep learning-based methods have been explored. Guérin et al. [7] proposed a framework based on conditional Generative Adversarial Networks (cGANs) trained on real-world terrain data. Their system enables users to generate terrain models by sketching features such as ridgelines, rivers, and elevation reference points. Zhang et al. [26] extended this approach to support control over both elevation and terrain style. Lochner et al. [16] employed diffusion models to generate higher-quality heightmaps, also allowing users to control terrain geometry and style through sketches. However, these methods focus exclusively on generating heightmaps and do not address texture generation.

Texture generation. Zhu et al. [29] proposed a method for generating satellite imagery from labeled map data. Dachsbacher et al. [4] reconstructed terrain textures from satellite images using heightmaps, temperature, precipitation, and solar radiation data. Kanai et al. [11] introduced a method for generating terrain textures that reflect seasonal variations. However, these approaches focus solely on texture generation and do not address heightmap generation.

Joint generation. Spick et al. [22] proposed a method for jointly generating heightmaps and textures using GANs. They trained a Spatial GAN [28] on a satellite image dataset containing both terrain heightmaps and corresponding textures, representing them as four-channel images to enable joint generation. However, this method is limited to random terrain generation and does not support user control. Moreover, GANs typically produce lower-quality outputs than more recent diffusion-based methods and offer limited flexibility for image editing. In contrast, our method leverages diffusion models to achieve high-quality terrain generation with user-guided control.

Concurrently, MESA [3] explores text-driven terrain generation using an LDM trained on global terrain data. In contrast to our approach, which focuses on localized and sketch-based control, MESA operates at a global scale with textual conditioning. While MESA employs

separate heads for heightmaps and textures, our method jointly generates two modalities within a fused latent space, supported by a VAE tailored for elevation data. Moreover, MESA provides only qualitative results of its method, whereas we conduct extensive evaluations, both quantitative and qualitative, to demonstrate the effectiveness of our method over various baselines.

3. Background

Latent Diffusion Model (LDM). LDM [20] is an image generation model based on diffusion models [8]. To improve computational efficiency, it uses a VAE [14] to compress input images into a low-dimensional latent space, where the diffusion process is applied during training and inference. The compression and reconstruction of an image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ (where H, W, and C are the height, width, and number of channels of the image, respectively) using the VAE encoder $\mathcal{E}$ and decoder $\mathcal{D}$ are defined as follows:

$$\mathbf{z} = \mathcal{E}(\mathbf{x}), \quad \hat{\mathbf{x}} = \mathcal{D}(\mathbf{z}), \tag{1}$$

where $\mathbf{z} \in \mathbb{R}^{h\times w\times c}$ denotes the low-dimensional latent representation and h, w, and c are the height, width, and number of channels of the latent, respectively. The diffusion process in this latent space is given by:

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

where t is the timestep, $\bar{\alpha}_t$ is a constant, and $\mathbf{I}$ is the identity matrix. In the reverse diffusion process, a neural network ϵ_θ is trained to predict the added noise by minimizing the following loss:

$$\mathcal{L}_{\text{LDM}} := \mathbb{E}_{\mathbf{z},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t}\left[|\epsilon - \epsilon_\theta(\mathbf{z}_t, t)|_2^2\right]. \tag{3}$$

To generate an image, the process starts with a noise vector $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Noise is then iteratively predicted and removed using ϵ_θ, and the final latent $\mathbf{z}_0$ is decoded by $\mathcal{D}$ to produce the output image. In this paper, we extend the LDM framework to jointly generate heightmaps and textures, and to support user-guided control.

4. Method

An overview of our method is shown in Figure 2. Our approach builds the terrain generation model in two stages. In the first stage, we train an LDM [20] in an unsupervised manner to model the joint distribution $p(\mathbf{h}, \mathbf{x})$, where $\mathbf{h} \in \mathbb{R}^{H\times W}$ is a heightmap and $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ is the corresponding texture. In the second stage, we incorporate user input such as sketches, represented as a conditioning signal $\mathbf{c} \in \mathbb{R}^{H\times W\times 3}$, and extend the model to learn the conditional distribution $p(\mathbf{h}, \mathbf{x} \mid \mathbf{c})$. We achieve this by adopting ControlNet [27] and training an external adapter in a supervised manner to condition the generation process on $\mathbf{c}$.

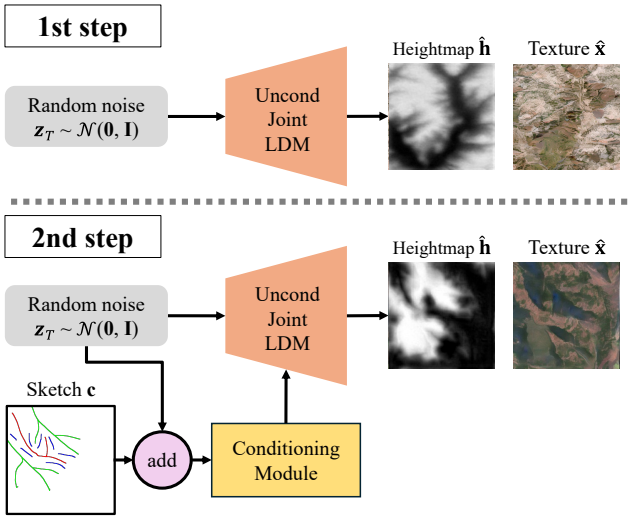

Figure 2. Overview of our method. Our approach consists of two stages: (1) training of an unconditional joint generative model that randomly generates heightmaps and textures, and (2) training of a conditioning module that enables output control based on inputs such as user sketches.

4.1. Dataset

We used NASADEM elevation data[1] (30 m spatial resolution) for heightmaps, and atmospherically corrected Sentinel-2 imagery[2] (Level-2A product, 10 m spatial resolution) for terrain textures. We focused on natural terrain between 60°S and 60°N, extracting $1° \times 1°$ geographic regions from 276 land areas (see Appendix A for the details). Each region was resampled to a spatial resolution of 25 m, producing images of approximately 4096×4096 pixels. These were divided into non-overlapping 256×256 patches, excluding data with missing texture details due to clouds and shadows. To match the input size required by the diffusion model, each patch was then upsampled to 512×512. The elevation values in the heightmaps ranged from 0 to 8000 m. However, as discussed in Section 4.2, VAEs struggle to capture this wide distribution accurately. To mitigate this, we excluded minor samples with elevations above 2000 m. This preprocessing yielded a total of 4,119 paired samples of heightmaps and texture images.

4.2. VAE Training

For texture images, we use the VAE from Stable Diffusion [20]. However, applying this VAE to heightmaps leads to poor reconstruction quality due to a mismatch between the VAE's training distribution and elevation

[1] https://lpdaac.usgs.gov/products/nasadem_hgtv001/
[2] https://sentiwiki.copernicus.eu/web/s2-processing#S2Processing-CopernicusSentinel-2Collection-1AvailabilityStatus

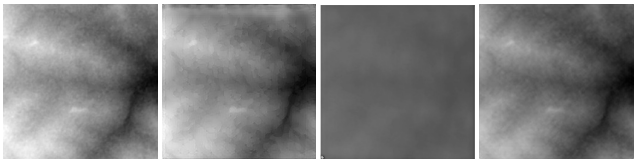

Figure 3. Comparison of heightmap reconstruction results. (a) Input image, (b) Reconstruction using the VAE from Stable Diffusion, (c) Reconstruction using a VAE trained on data with a maximum elevation of 8000 m, (d) Reconstruction using a VAE trained on data with a maximum elevation of 2000 m. Zooming in is recommended to better observe fine-grained differences.

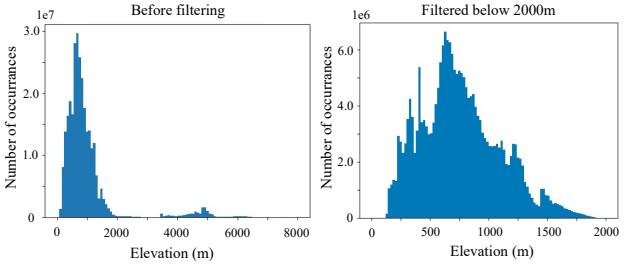

Figure 4. Elevation distribution of the heightmap dataset before (left) and after (right) filtering.

distribution (see Figure 3 (b)). To resolve this, we train a VAE specifically for heightmaps. We adopt a Kullback-Leibler (KL) Autoencoder with the same architectural design as the VAE used in Stable Diffusion.

Normalization. It is a standard practice to normalize the inputs and outputs of a VAE to the range $[-1, 1]$ for compatibility with activation functions. Accordingly, we normalize heightmaps in this study using the following equation:

$$\mathbf{h}_{scaled} = \left(\frac{\mathbf{h}}{H_{\max}} - 0.5 \right) \times 2, \tag{4}$$

where $\mathbf{h}$ is the original heightmap, $\mathbf{h}_{\text{scaled}}$ is the normalized version, and $H_{\max}$ denotes the maximum elevation value in the dataset.

As mentioned in Section 4.1, although elevation values from NASADEM range from 0 to 8000 m, the VAE struggles to capture elevation variation when using $H_{\max} = 8000$ for normalization in Equation (4). This normalization results in poor reconstruction quality (see Figure 3 (c)). As shown in Figure 4, most heightmaps in the dataset have elevations below 2000 m. To improve reconstruction quality, we excluded heightmaps with elevations above 2000 m and normalized the remaining data using $H_{\max} = 2000$. This normalization yields better reconstruction results (see Figure 3 (d)). Handling higher-elevation terrain is left as future work.

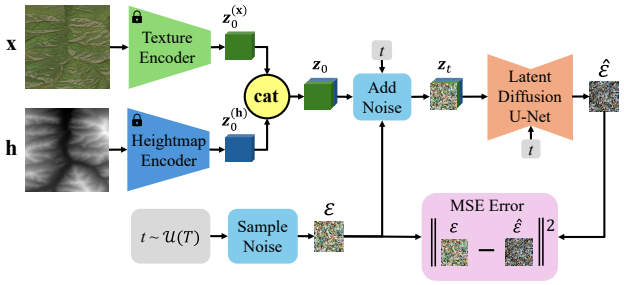

Figure 5. Training flow of the unconditional joint generative model.

4.3. Unconditional Joint Generative Model

Training. We train an LDM to jointly generate heightmaps and texture images in the latent space of the VAE introduced in Section 4.2. Figure 5 illustrates the training architecture. Given a heightmap $\mathbf{h}$ and a texture image $\mathbf{x}$, they are encoded into their respective latent representations $\mathbf{z}^{(\mathbf{h})}$ and $\mathbf{z}^{(\mathbf{x})}$ as follows:

$$\mathbf{z}^{(\mathbf{h})} = \mathcal{E}_h(\mathbf{h}), \quad \mathbf{z}^{(\mathbf{x})} = \mathcal{E}_x(\mathbf{x}), \tag{5}$$

where $\mathcal{E}_h$ is the encoder for heightmaps, and $\mathcal{E}_x$ is the encoder from Stable Diffusion. Noise is added to these latent representations using the diffusion process defined in Equation (2), and a U-Net ϵ_θ is trained to predict the added noise from the noisy inputs. The loss function $\mathcal{L}$ is defined as follows:

$$\mathcal{L} := \mathbb{E}_{\mathbf{z}_0^{(\mathbf{h})}, \mathbf{z}_0^{(\mathbf{x})}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \left[\left| \epsilon - \epsilon_\theta(\mathbf{z}_t^{(\mathbf{h})}, \mathbf{z}_t^{(\mathbf{x})}, t) \right|^2 \right]. \tag{6}$$

While the original U-Net predicts noise for a single latent variable $\mathbf{z}_t$, our method predicts noise for both the heightmap and texture latents, $\mathbf{z}_t^{(\mathbf{h})}$ and $\mathbf{z}_t^{(\mathbf{x})}$, respectively.

Inference. The inference process is illustrated in Figure 6. During inference, randomly sampled latent representations $\mathbf{z}_T^{(\mathbf{h})}$ and $\mathbf{z}_T^{(\mathbf{x})}$ are used as inputs, and noise is iteratively predicted and removed using the U-Net. The resulting latents, $\hat{\mathbf{z}}_0^{(\mathbf{h})}$ and $\hat{\mathbf{z}}_0^{(\mathbf{x})}$, are then decoded into the final heightmap $\hat{\mathbf{h}}$ and texture image $\hat{\mathbf{x}}$ as follows:

$$\hat{\mathbf{h}} = \mathcal{D}_h(\hat{\mathbf{z}}_0^{(\mathbf{h})}), \quad \hat{\mathbf{x}} = \mathcal{D}_x(\hat{\mathbf{z}}_0^{(\mathbf{x})}), \tag{7}$$

where $\mathcal{D}_h$ is the decoder for heightmaps, and $\mathcal{D}_x$ is the decoder from Stable Diffusion.

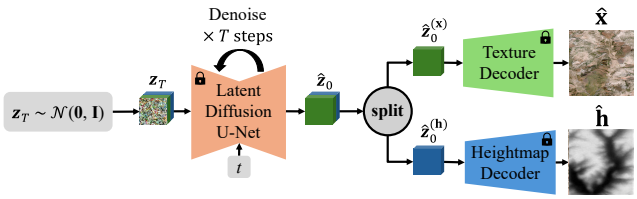

Figure 6. Inference flow of the unconditional joint generative model.

Latent code fusion. We extend the U-Net architecture used in LDM [20] to jointly process textures and heightmaps in a unified latent space. Specifically, latent representations $\mathbf{z}_t^{(\mathbf{h})}$ and $\mathbf{z}_t^{(\mathbf{x})}$ are concatenated along the channel dimension to form a single input $\mathbf{z}_t = \text{cat}(\mathbf{z}_t^{(\mathbf{h})}, \mathbf{z}_t^{(\mathbf{x})})$, where $\text{cat}(\cdot, \cdot)$ denotes concatenation along the dimension of channels. The output is similarly defined as $\hat{\mathbf{z}}_0 = \text{cat}(\hat{\mathbf{z}}_0^{(\mathbf{h})}, \hat{\mathbf{z}}_0^{(\mathbf{x})})$. To support this structure, we double the number of input and output channels in the U-Net relative to the original LDM implementation.

Leveraging Stable Diffusion prior. To enhance the quality of the generated terrain data, we leverage the prior knowledge of Stable Diffusion [20], which was pre-trained on large-scale image datasets. We first extend the input and output layers of Stable Diffusion's U-Net to double the number of channels, aligning with the requirements of our model. The additional channels are initialized with random weights. We then fine-tune the extended U-Net using the loss function defined in Equation (6). As Stable Diffusion is a text-conditioned model, we use a fixed prompt, "*A satellite terrain image.*", during both training and inference.

4.4. Conditional Joint Generative Model

To enable user control over the unconditional joint generation model described in Section 4.3, we extend it into a conditional model using ControlNet [27]. Among various possible conditioning inputs, we follow the approach of Lochner et al. [16] and use user-provided sketches representing terrain geometry (see Appendix B for the details).

Figure 7 provides an overview of the training process. Let $\mathbf{c}$ be the input sketch image, and let ϵ_{θ_c} denote the U-Net with ControlNet, parameterized by θ_c. The training loss $\mathcal{L}_c$ is defined as follows:

$$\mathcal{L}_c := \mathbb{E}_{\mathbf{z}_0^{(\mathbf{h})}, \mathbf{z}_0^{(\mathbf{x})}, \mathbf{c}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \left[\left| \epsilon - \epsilon_{\theta_c}(\mathbf{z}_t^{(\mathbf{h})}, \mathbf{z}_t^{(\mathbf{x})}, \mathbf{c}, t) \right|^2 \right]. \tag{8}$$

The inference process follows the same procedure as the unconditional model, except that the sketch input is additionally provided to the U-Net via the trained adapter.

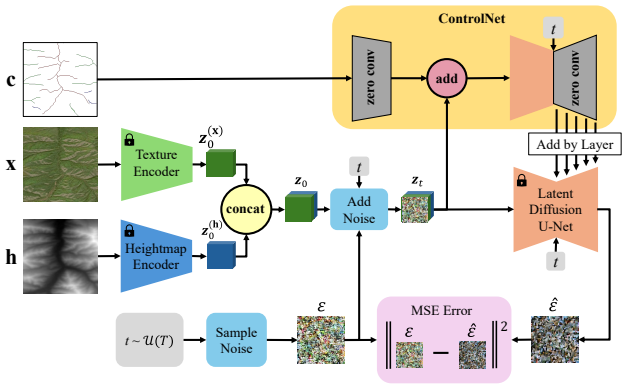

Figure 7. Training flow of the conditional joint generative model.

5. Experiments

Experimental settings. Our method was implemented using Python, PyTorch, and the Diffusers library [24]. For the unconditional joint generative model, we used the AdamW optimizer [17] with a learning rate of 1×10^{-4}. The noise scheduler was set to DDPM [8] with 1000 timesteps. Training was performed for 200 epochs on an NVIDIA RTX A6000 GPU, taking approximately two days. During inference, the reverse diffusion process also used 1000 steps and required about 75 seconds to generate a single 512×512 heightmap and texture pair. For the conditional joint generative model, we used the Adam optimizer [13] with a learning rate of 1×10^{-5}, along with the same DDPM scheduler. Training was conducted for 500 epochs on an NVIDIA RTX 6000 Ada GPU and took approximately seven days. At inference time, the number of reverse diffusion steps was reduced to 20, enabling generation of a single 512×512 sample in approximately 2 seconds.

Compared methods. As baselines for comparison, we trained two-stage generation models in which heightmaps and textures are generated separately using individual LDMs. Specifically, we prepared two model pairs: one that first generates heightmaps independently and then generates textures conditioned on the heightmaps (i.e., $\mathbf{h} \rightarrow \mathbf{x}$), and another that first generates textures independently and then generates heightmaps conditioned on the textures (i.e., $\mathbf{x} \rightarrow \mathbf{h}$). We also consider comparison with a GAN-based terrain generation method by Spick et al. [22]. Unfortunately, their code and pre-trained model are not publicly avail-

Table 1. Quantitative comparison of textures between the two-stage baselines and our method. Lower values indicate better performance. The best score is shown in **bold**, and the second-best is underlined.

Method	$\text{FID}_{\text{CLIP}}\downarrow$
Baseline ($\mathbf{h} \rightarrow \mathbf{x}$)	24.0
Baseline ($\mathbf{x} \rightarrow \mathbf{h}$)	19.4
Ours (from scratch)	<u>16.7</u>
Ours (fine-tuning)	**9.8**

Table 2. Quantitative comparison of textures between PSGAN [2] and our method.

Method	$\text{FID}_{\text{CLIP}}\downarrow$
PSGAN [2]	22.1
Ours (from scratch)	<u>16.7</u>
Ours (fine-tuning)	**9.8**

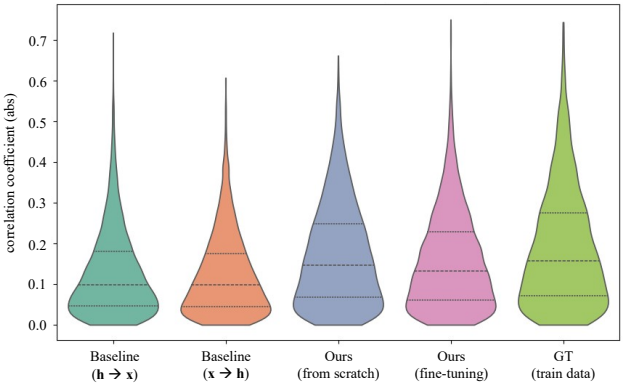

Figure 8. Comparison of correlation coefficients between textures and heightmaps generated by the two-stage baselines and our method. Dashed lines within each distribution indicate the first quartile, median, and third quartile.

able and the official implementation of SGAN used in their method is too old to use. We thus instead trained PSGAN [2], an improved version of SGAN, using our dataset.

5.1. Quantitative Evaluation

In this section, we quantitatively evaluate the quality of the generated textures, as well as the extent to which correlation between textures and heightmaps is preserved. For this evaluation, we use 4,119 pairs of heightmaps and textures generated by each model, corresponding to the same number of data used during training.

Texture quality. To evaluate the quality of generated terrain textures, we use CLIP Fréchet Inception

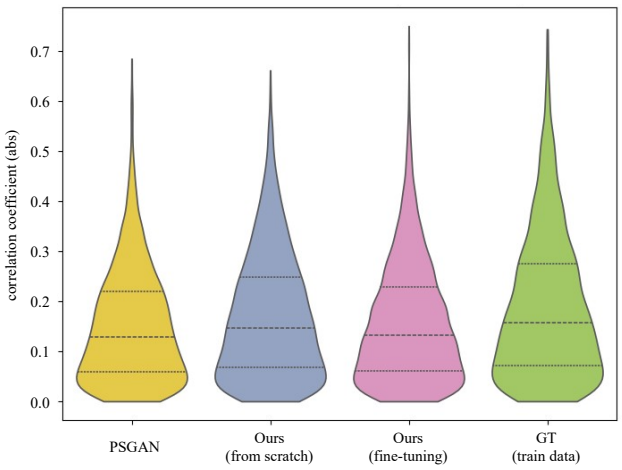

Figure 9. Comparison of correlation coefficients between textures and heightmaps generated by PSGAN [2] and our method.

Distance (FID_{CLIP}) [15], computed between the generated images and the training images. While standard FID uses features from Inception-V3 trained on natural images, it may not reliably capture semantic similarity in terrain textures. In contrast, FID_{CLIP} employs the CLIP visual encoder, which provides more semantically meaningful features and is thus better suited for texture quality evaluation.

Table 1 shows a comparison between our method and the two-stage baselines. Compared to the two-stage baselines, our joint generation model achieves lower scores, indicating higher similarity to the real textures. Within our methods, the fine-tuned model outperforms the model trained from scratch. Table 2 also demonstrates that our method outperforms the GAN-based method [2].

Correlation between textures and heightmaps. We quantitatively evaluate how well each model captures the correlation between heightmaps and textures. Specifically, we compute the Pearson correlation coefficient between the pixel values of the heightmap and each channel of the corresponding texture image, and then average the results across channels. The distribution of correlation coefficients across all generated pairs is visualized using violin plots for comparison. The closer a model's distribution is to the training data, the better it reflects real-world terrain relationships.

The results are presented in Figure 8. The two-stage generation models produce overall lower correlation values compared to the training data. In contrast, our models yield distributions that more closely match the training distribution. These results suggest that the

Table 3. Statistics of correlation coefficients (↑) between textures and heightmaps generated by the two-stage baselines and our method. IQR denotes the interquartile range. Values in parentheses indicate absolute differences from the ground-truth (training data). The best scores are shown in bold, and the second-best are underlined.

Method	mean	std	25%	50%	75%	IQR
Baseline ($\mathbf{h} \rightarrow \mathbf{x}$)	0.13 (0.06)	0.11 (0.03)	0.05 (0.02)	0.10 (0.06)	0.18 (0.10)	0.13 (0.07)
Baseline ($\mathbf{x} \rightarrow \mathbf{h}$)	0.12 (0.07)	0.10 (0.04)	0.04 (0.03)	0.10 (0.06)	0.18 (0.10)	0.13 (0.07)
Ours (from scratch)	**0.17 (0.02)**	**0.12 (0.02)**	**0.07 (0.00)**	**0.15 (0.01)**	**0.25 (0.03)**	**0.18 (0.02)**
Ours (fine-tuning)	0.16 (0.03)	**0.12 (0.02)**	0.06 (0.01)	0.13 (0.03)	0.23 (0.05)	0.17 (0.03)
GT (train data)	0.19	0.14	0.07	0.16	0.28	0.20

Table 4. Statistics of correlation coefficients (↑) between textures and heightmaps generated by PSGAN [8] and our method. IQR denotes the interquartile range. Values in parentheses indicate the absolute differences from the GT (training data)

Method	mean	std	25%	50%	75%	IQR
PSGAN	0.15 (0.04)	0.11 (0.03)	0.06 (0.01)	0.13 (0.03)	0.22 (0.06)	0.16 (0.04)
Ours (from scratch)	**0.17 (0.02)**	**0.12 (0.02)**	**0.07 (0.00)**	**0.15 (0.01)**	**0.25 (0.03)**	**0.18 (0.02)**
Ours (fine-tuning)	0.16 (0.03)	**0.12 (0.02)**	0.06 (0.01)	0.13 (0.03)	0.23 (0.05)	0.17 (0.03)
GT (train data)	0.19	0.14	0.07	0.16	0.28	0.20

joint generative models are more effective at capturing the correlation between heightmaps and textures. In addition, Figure 9 shows a comparison between our method and PSGAN [2]. Although the correlation values of our method and PSGAN are comparable, PSGAN often yields artifacts in hightmaps, as discussed in Section 5.2.

We also evaluate how well each model captures the correlation between heightmaps and textures. For each sample, we compute the Pearson correlation coefficient between the heightmap and each texture channel, and then average across channels. The distributions of these correlation coefficients are visualized using violin plots. A distribution closer to that of the training data indicates better modeling of real-world terrain relationships. Figure 8 presents the results. Overall, the two-stage models produce lower correlation values compared to the training data, while our models yield distributions that more closely match the training distribution. Table 3 summarizes the differences in correlation values between each method and the ground-truth data, further demonstrating the effectiveness of the joint generative models. In addition, Figure 9 and Table 4 show comparisons between our method and PSGAN [2]. Although the correlation values of the two methods are comparable, PSGAN often produces artifacts in heightmaps, as discussed in Section 5.2.

5.2. Qualitative Evaluation

Unconditional terrain generation. Figure 10 shows heightmaps and textures generated by each model, and Figure 11 presents their corresponding 3D visualizations. In Figures 10 and 11 (a), the two-stage model that generates textures conditioned on heightmaps produces textures with limited variation, often dominated by flat green tones, indicating low diversity. In contrast, (b) shows that the model generating heightmaps conditioned on textures yields higher texture diversity, but frequently results in unrealistic color schemes. The heightmaps also exhibit issues: many outputs contain unnatural structures such as elevated edges (e.g., the third and fourth rows in (a), and the fourth row in (b)). Additionally, heightmap diversity is low. For example, in the first and third rows of Figure 10 (b), the outputs are often sharply divided into black and white regions.

Next, Figures 10 and 11 (c) show results from our model trained from scratch. While some textures exhibit unnatural colors, both the heightmaps and textures demonstrate high diversity. In (d), where our model is fine-tuned from Stable Diffusion [20], we observe similarly high diversity, along with more natural texture color schemes. Compared to the outputs from (a), (b), and (c), the model in (d) produces fewer heightmaps with unnatural structures. These results suggest that the model effectively leverages the prior knowledge of Stable Diffusion. Notably, in the second row of (d), the texture appearance varies with elevation, indicating that the correlation between them has been successfully captured.

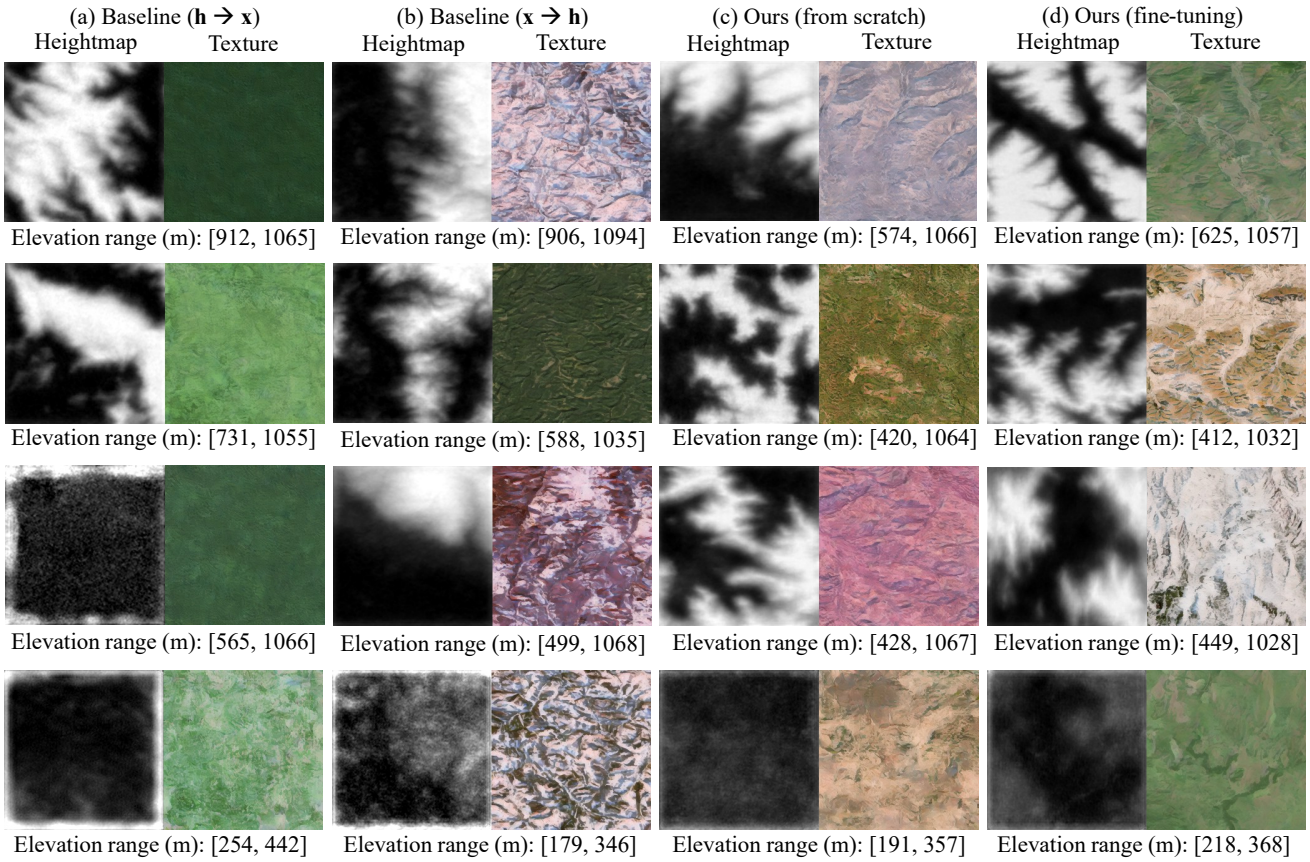

Figure 10. Comparison of terrain data generated by our method and the two-stage baselines. Heightmaps are normalized within the range of [0, 1] for better display, where white and black represent higher and lower elevations, respectively. The original elevation range is shown below each image.

Comparison with PSGAN. Figure 12 shows a comparison of terrain data generated by our method and PSGAN [2]. The heightmaps generated by PSGAN exhibit noticeable noise in their elevation values despite we applied a median filter as a postprocessing, following Spick et al. [22]. Also, the heightmaps generated by PSGAN show repeated small-scale undulations, and the corresponding textures exhibit periodic patterns aligned with the elevation features. This suggests that PSGAN learns to repeatively replicate local patterns from the training data. As a result, PSGAN struggles to generate globally coherent terrain structures, such as continuous ridgelines or expansive plains.

In contrast, the heightmaps generated by our method exhibit broad ridgelines near the centers of mountainous regions, with finer ridgelines radiating outward. The plains extending from the base of the mountains also appear smooth and natural, without any unnatural undulations. These observations indicate that, compared to PSGAN, our method is more effective in generating terrain that reflects natural structure and realism.

Conditional terrain generation. The results of conditional generation using user sketches are shown in Figure 13. The lines drawn in the input sketches align closely with elevation changes in the generated heightmaps. As shown in Figure 14, even sketches representing extreme terrain geometry not found in the real world are faithfully reflected in the outputs. Figure 15 presents results from sketches with similar overall shapes but differing levels of detail. Between two sketches sharing the same ridgeline, the one with more detailed surrounding features leads to outputs where the ridge is more clearly emphasized, resulting in a closer match to the input sketch.

Texture Conditioning. To enable more flexible terrain generation, we explored texture control. As shown on the left side of Figure 16, we first created a two-color-quantized texture dataset by approximating each

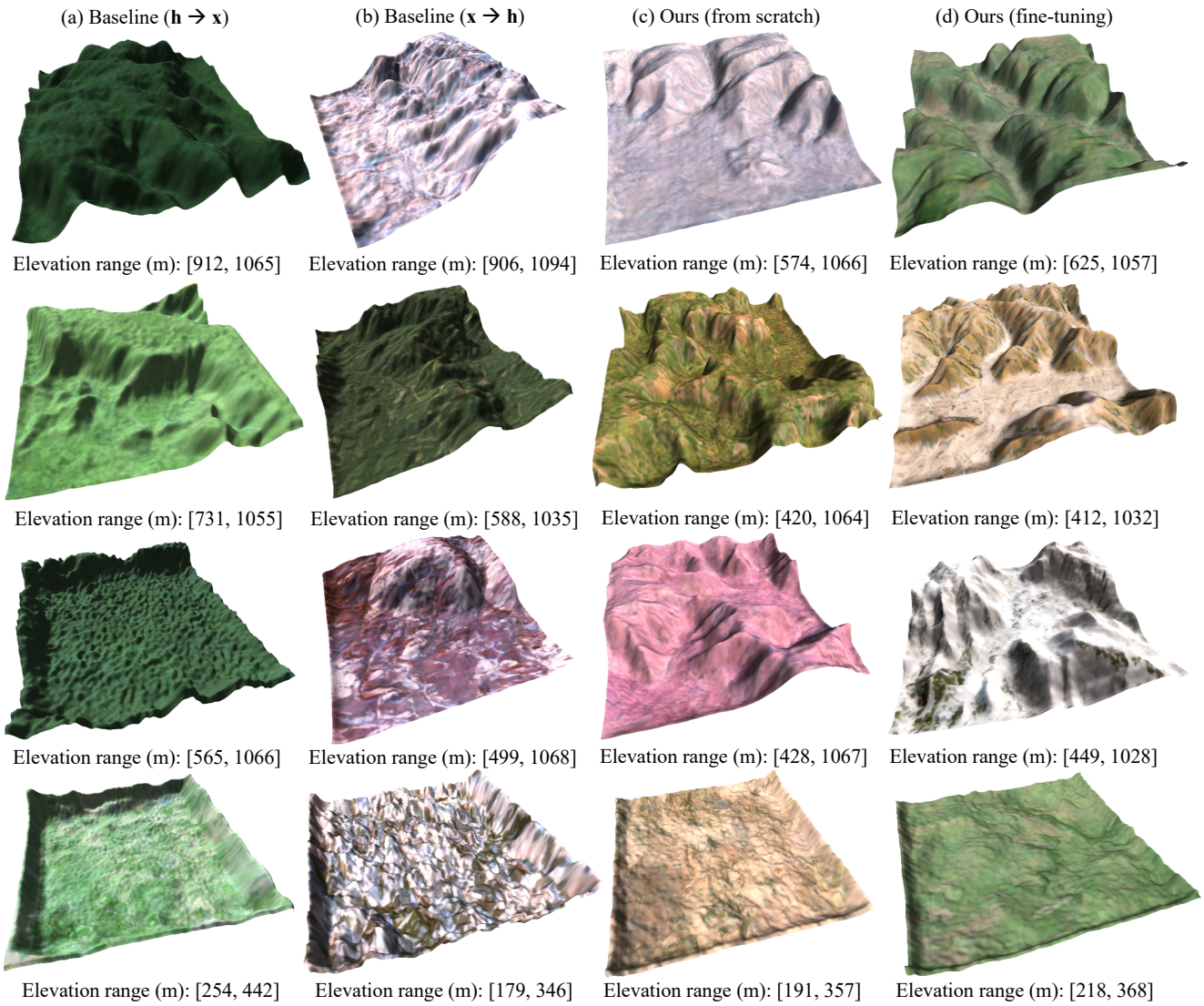

Figure 11. 3D visualization of terrain data generated by our method and the two-stage baselines.

texture image in the satellite image dataset with two representative colors. ControlNet was then trained using this dataset, following the same procedure described in Figure 7. Given a user-drawn two-color image, the model synthesizes a texture conditioned on that image. Representative results are shown on the right side of Figure 16. These textures reflect the user-drawn inputs; however, the current model struggles to capture global terrain features, indicating room for improvement. Producing terrain-like outputs also requires complex input cues, and enabling simpler inputs remains an open challenge.

6. Discussion

We demonstrate that our method outperforms GAN-based approaches [2, 22], which represent the state of the art in unconditional terrain generation. A recent concurrent study, MESA [3], employs a diffusion-based model for terrain synthesis, as discussed in Section 2. However, MESA focuses on text-driven terrain generation, which differs from the unconditional and sketch-based tasks addressed in our work, making direct comparison beyond the scope of this study. We regard our method as complementary, and extending it to support textual input remains an interesting direction for future research.

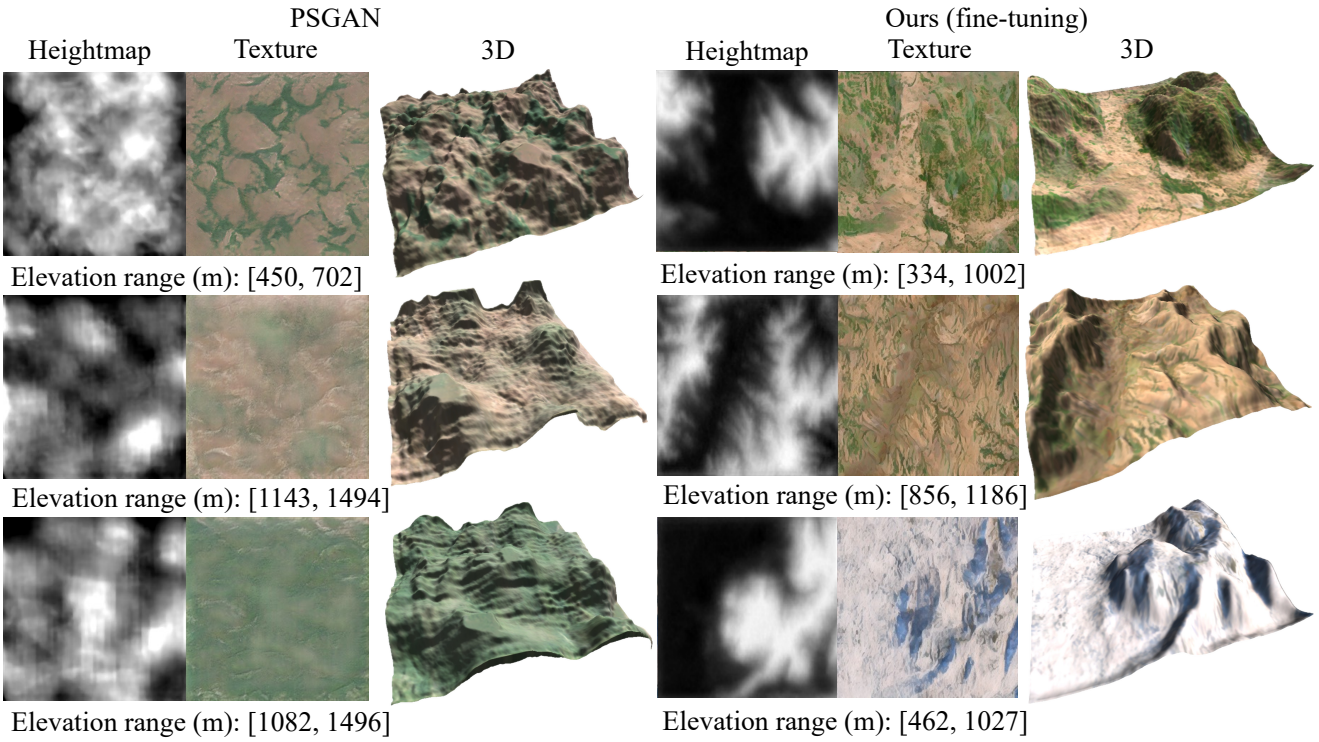

Figure 12. Comparison of terrain data generated by our method and PSGAN [2].

Evaluation by professional artists. We asked professional artists at the game studio Polyphony Digital Inc. to review our generated terrains and provide feedback. They noted that the textures vary plausibly with local slope and elevation, suggesting that our framework successfully captures essential geometry–appearance correlations. However, they also identified several limitations. Some textures contain baked-in shading, indicating the need for shading-free (albedo) textures via de-lighting. In addition, low-frequency noise sometimes causes blurring in the textures. In terms of geometry, the relatively low resolution of both heightmaps and textures limited the evaluation of fine-scale realism relative to real-world terrains. The generated results also lacked key geomorphological features such as rivers, lakes, tree lines, and snow cover, which are important for producing convincing landscapes. The artists further emphasized the importance of explicitly defining spatial scale (e.g., 500 $m \times 500$ m versus 10 $km \times 10$ km), because recognizable landforms depend on scale. For more in-depth evaluation, they suggested focusing on specific geographic regions to simplify the problem, improve output quality, and assess results in the context of region-specific landforms.

Resolution Resolution. Although our current experiments are conducted at a resolution of 512×512, scaling diffusion-based terrain generation to higher resolutions remains a significant challenge. While GANs are often considered efficient for high-resolution synthesis, recent advances in latent diffusion models (LDMs), such as SDXL [19], demonstrate comparable capabilities. We believe our framework can be extended to higher resolutions by leveraging such improved LDMs, as well as approaches like latent-space super-resolution [10], which support coherent upsampling in the latent domain and help preserve structural fidelity and fine detail. Furthermore, methods such as consistency models [21], which reduce the number of denoising steps, may lower computational costs during high-resolution inference. Exploring these approaches is an important direction for future work.

7. Conclusion

In this paper, we have proposed a method for jointly generating terrain heightmaps and textures. By extending the latent space of an LDM, we model the joint distribution of heightmaps and textures, enabling unconditional terrain generation. Furthermore, by integrating this model with ControlNet, we enable user

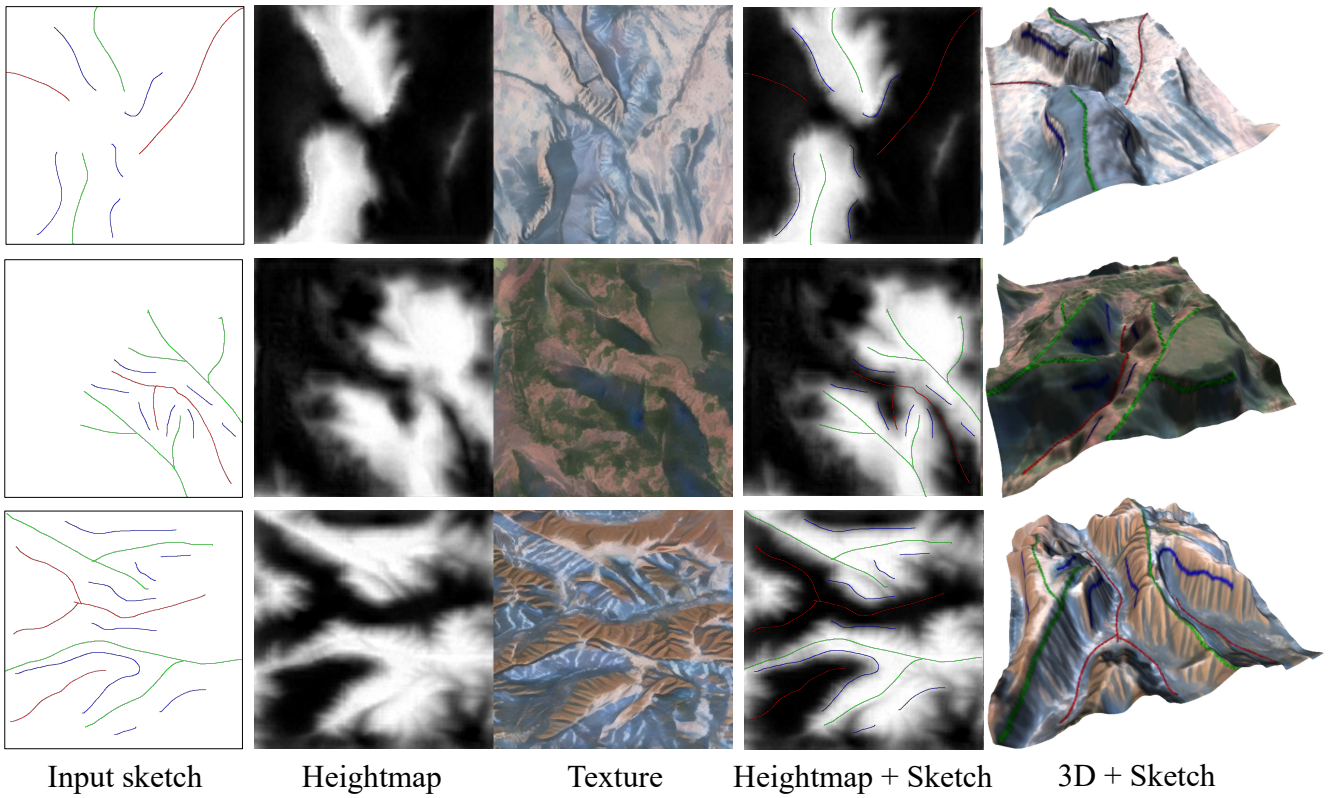

Figure 13. Results of terrain generation from sketch inputs. Red lines represent valleys, green lines indicate ridgelines, and blue lines denote cliffs.

control of terrain geometry through sketch input. Qualitative results show that our method outperforms two-stage generation approaches and a GAN-based method. Quantitatively, we confirmed that it captures a strong correlation between heightmaps and textures. In the sketch-based control setting, our method successfully generates not only realistic terrain geometry but also extreme, user-defined geometries.

Limitations and future work. Although fine-tuning Stable Diffusion improved output quality in our method, artifacts in the generated textures remain. Additionally, some heightmaps exhibit unnatural structures, suggesting that the overall quality is not yet sufficient for practical use in real-world 3D terrain modeling. To address these limitations, we plan to expand the dataset and explore alternative data representations to further enhance output quality. Beyond sketch-based input, incorporating other forms of conditioning, such as text prompts or semantic masks, could enable more precise control over texture colors and patterns. These enhancements would significantly improve the practical utility of our approach as a terrain generation tool.

References

[1] Richard Barnes. *RichDEM: Terrain Analysis Software*, 2016. 15

[2] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 469–477. JMLR.org, 2017. 6, 7, 8, 9, 10

[3] Paul Borne-Pons, Mikolaj Czerkawski, Rosalie Martin, and Romain Rouffet. MESA: Text-driven terrain generation using latent diffusion and global copernicus data. *arXiv preprint arXiv:2504.07210*, 2025. 2, 9

[4] Carsten Dachsbacher, Tobias Bolch, and Marc Stamminger. Procedural reproduction of terrain textures with geographic data. In *Vision, Modeling and Visualization (VMV 2006)*, pages 105–112. IOS Press, 2006. 2

[5] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. A review of digital terrain modeling. *Comput. Graph. Forum*, 38(2):553–577, 2019. 1

[6] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. A review of digital terrain modeling. *Computer Graphics Forum*, 38(2):553–577, 2019. 2

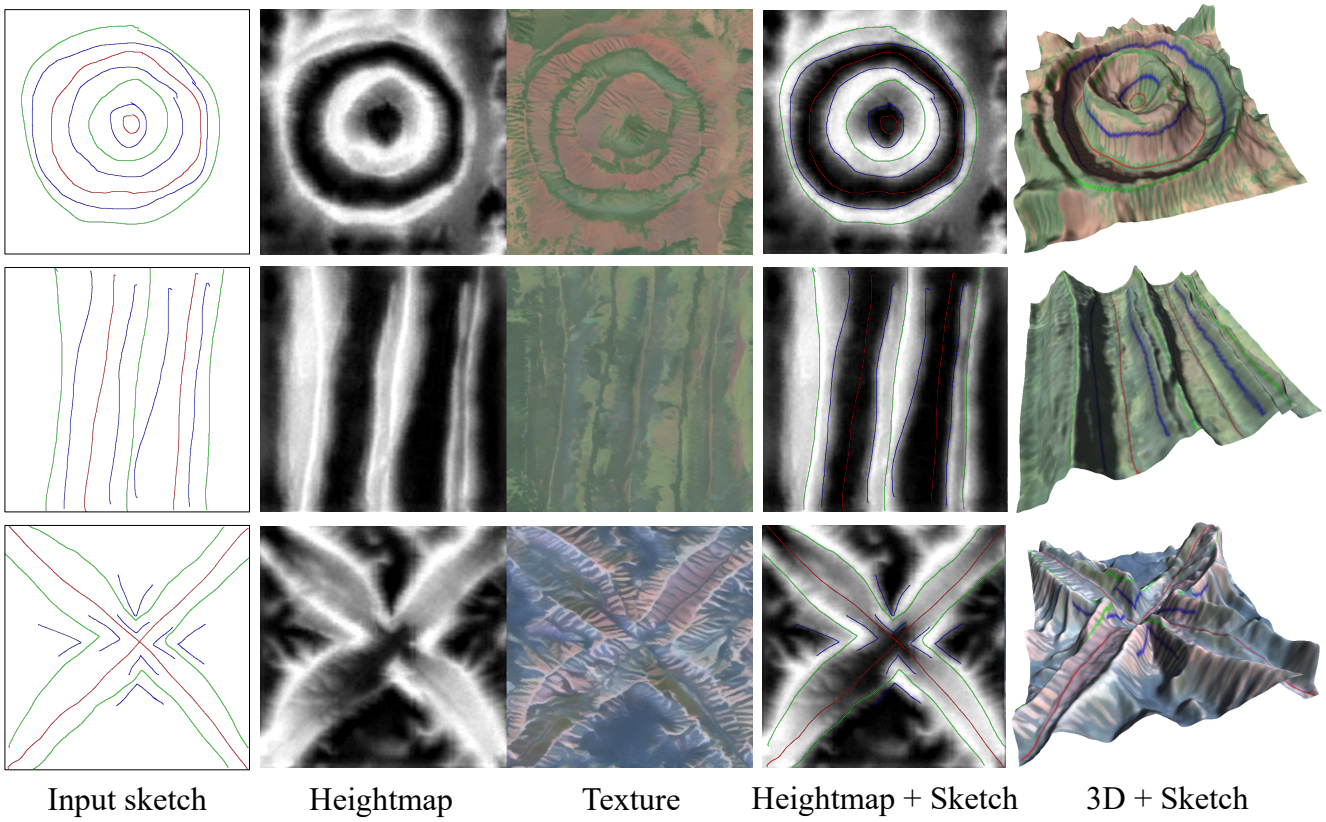

Figure 14. Results of terrain generation from sketch inputs representing extreme terrain geometry not found in the real world.

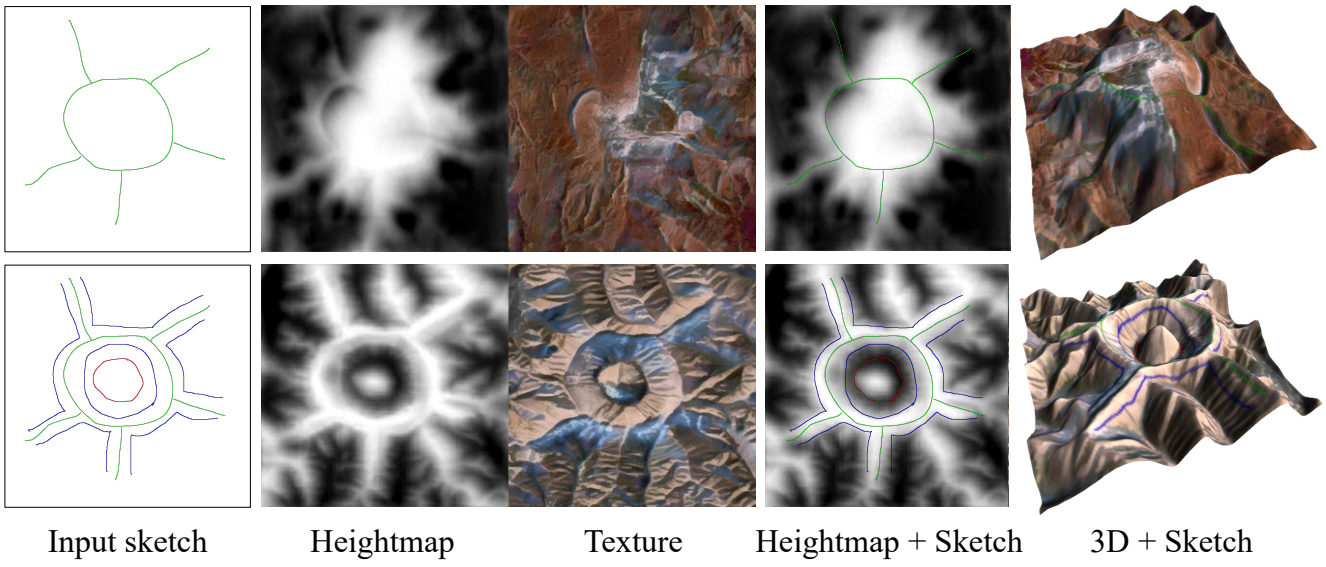

Figure 15. Results of conditional generation using sketches with similar overall shapes but varying levels of detail.

[7] Éric Guérin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoît Martinez. Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Trans.*

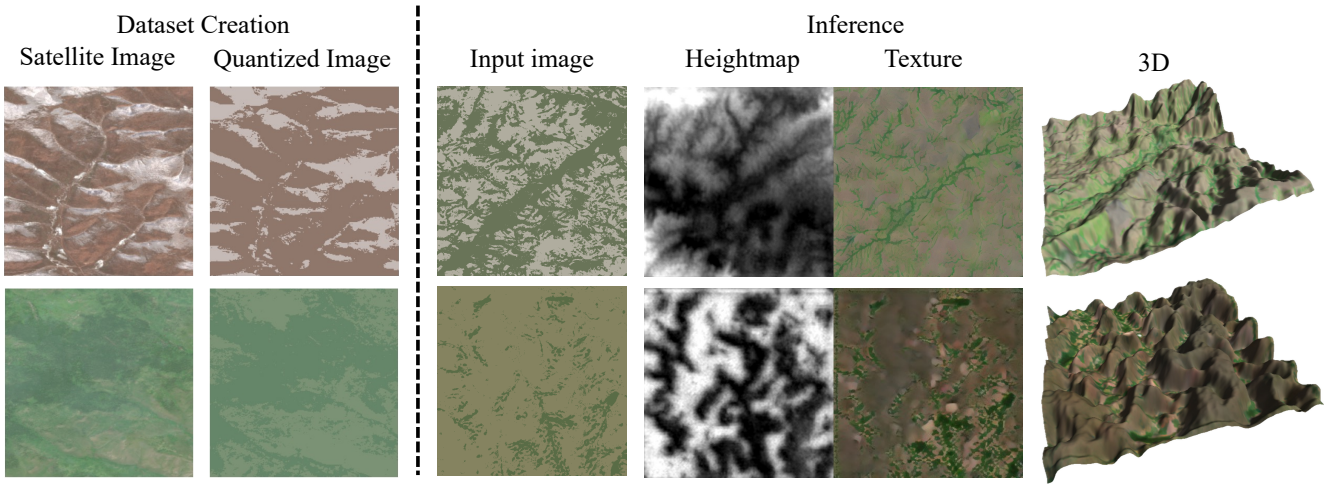

Figure 16. Details on our experiment on texture conditioning using ControlNet.

Graph., 36(6):228–1, 2017. 1, 2

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3, 5

[9] Yi Huang, Jiancheng Huang, Yifan Liu, Mingfu Yan, Jiaxi Lv, Jianzhuang Liu, Wei Xiong, He Zhang, Shifeng Chen, and Liangliang Cao. Diffusion model-based image editing: A survey. *CoRR*, abs/2402.17525, 2024. 2

[10] Jinho Jeong, Sangmin Han, Jinwoo Kim, and Seon Joo Kim. Latent space super-resolution for higher-resolution image generation with diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2355–2365, 2025. 10

[11] Toshiki Kanai, Yuki Endo, and Yoshihiro Kanamori. Seasonal terrain texture synthesis via köppen periodic conditioning. *The Visual Computer*, pages 1–12, 2024. 1, 2

[12] Christina M. Kennedy, James R. Oakleaf, David M. Theobald, Sharon Baruch-Mordo, and Joseph Kiesecker. Managing the middle: A shift in conservation priorities based on the global human modification gradient. *Global Change Biology*, 25(3):811–826, 2019. 15

[13] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Conference proceedings: papers accepted to the International Conference on Learning Representations (ICLR) 2014*, 2014. Published on ArXiv as arXiv:1312.6114. 3

[15] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance. In *Proc. ICLR 2023*, 2023. 6

[16] J. Lochner, J. Gain, S. Perche, A. Peytavie, E. Galin, and E. Guérin. Interactive authoring of terrain using diffusion models. *Computer Graphics Forum*, 42(7): e14941, 2023. 1, 2, 5, 15

[17] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[18] Simon Perche, Adrien Peytavie, Bedrich Benes, Eric Galin, and Eric Guérin. Authoring terrains with spatialised style. *Comput. Graph. Forum*, 42(7), 2023. 1

[19] 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. In *ICLR*. OpenReview.net, 2024. 10

[20] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2, 3, 5, 7

[21] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023. 10

[22] Ryan Spick and James Alfred Walker. Realistic and textured terrain generation using gans. In *Proceedings of the 16th ACM SIGGRAPH European Conference on Visual Media Production*, pages 1–10, 2019. 2, 5, 8, 9

[23] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014. 15

[24] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022. 5

[25] D. Zanaga, R. Van De Kerchove, W. De Keersmaecker, N. Souverijns, C. Brockmann, R. Quast, J. Wevers, A. Grosu, A. Paccini, S. Vergnaud, O. Cartus, M. Santoro, S. Fritz, I. Georgieva, M. Lesiv, S. Carter, M. Herold, Linlin Li, N.E. Tsendbazar, F. Ramoino, and O. Arino. Esa worldcover 10 m 2020 v100. https://worldcover2020.esa.int, 2021. Accessed: 2025-01-20. 15

[26] Jian Zhang, Chen Li, Peichi Zhou, Changbo Wang, Gaoqi He, and Hong Qin. Authoring multi-style terrain with global-to-local control. *Graphical Models*, 119: 101122, 2022. 1, 2

[27] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 3, 5

[28] Lei Zhao, Sihuan Lin, Ailin Li, Huaizhong Lin, Wei Xing, and Dongming Lu. Spatialgan: Progressive image generation based on spatial recursive adversarial expansion. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2336–2344, 2020. 2

[29] J Zhu and T Kelly. Seamless satellite-image synthesis. In *Computer Graphics Forum*. Wiley, 2021. 2

Appendix

A. Detail of Terrain Dataset

For texture acquisition, we targeted land areas between 60°S and 60°N observable by Sentinel-2. We randomly sampled 500 locations, each covering a $1^\circ \times 1^\circ$ region in latitude and longitude. To focus on natural terrain, we excluded regions with an average elevation below 100 meters, which are likely to include urban or artificial structures. We further filtered locations using the Global Human Modification dataset [12], which quantifies human impact on the environment (spatial resolution: 1000 m; value range: [0, 1], with higher values indicating greater modification). Regions with a human modification index of 0.3 or higher were excluded. To mitigate climate-related sampling bias, we applied constraints based on the Köppen climate classification. We excluded arid zones and focused on four climate categories: tropical, temperate, subarctic, and polar. Sampling was limited to a maximum of 125 locations per category to maintain climate balance. We did not enforce uniform sampling (e.g., 100 per category), as some climate zones contained fewer than 100 suitable regions. After sampling, we applied additional filtering using the ESA WorldCover dataset [25] (spatial resolution: 10 m). We removed images including areas labeled as cropland, built-up regions, or water bodies, as were regions obscured by clouds or cloud shadows. As a result, we obtained usable data from 276 locations.

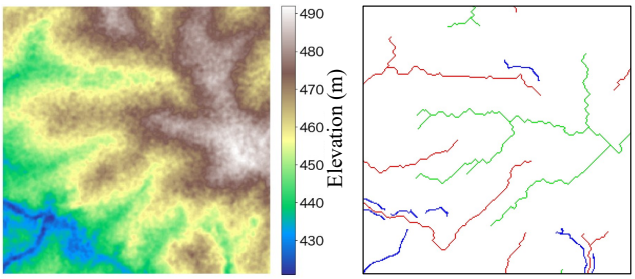

Figure 17. Sketch extraction (right) from a heightmap (left). The original heightmap is a grayscale image, but is visualized here with color mapping to emphasize elevation differences. Valleys are shown in red, ridgelines in green, and cliffs in blue.

B. Detail of Sketch Dataset

As illustrated in Figure 17, the sketches used in our method follow the method proposed by Lochner et al. [16], where valleys, ridgelines, and cliffs are represented by red, green, and blue lines, respectively. To generate these sketches, we use the `FillDepressions` and `FlowAccumulation` functions with the `D8` method from the Python library RichDEM [1], along with the Canny edge detection algorithm from the scikit-image library [23]. The `FillDepressions` function preprocesses the heightmap by filling sink areas, which is essential for simulating hydrological flow. The `FlowAccumulation` function with the `D8` method assumes that water flows from each pixel to its steepest downward neighbor and computes the accumulated flow per pixel. Pixels with flow values exceeding a predefined threshold are extracted as valleys. To detect ridgelines, we invert the elevation values of the heightmap and apply the same flow accumulation process. Finally, cliffs are extracted by applying the Canny edge detection algorithm to the heightmap.