Adaptive & Multi-Resolution Procedural Infinite Terrain Generation with Diffusion Models and Perlin Noise

Aryamaan Jain
IIIT
Hyderabad, India
aryamaan.jain@research.iiit.ac.in

Avinash Sharma
IIIT
Hyderabad, India
asharma@iiit.ac.in

K S Rajan
IIIT
Hyderabad, India
rajan@iiit.ac.in

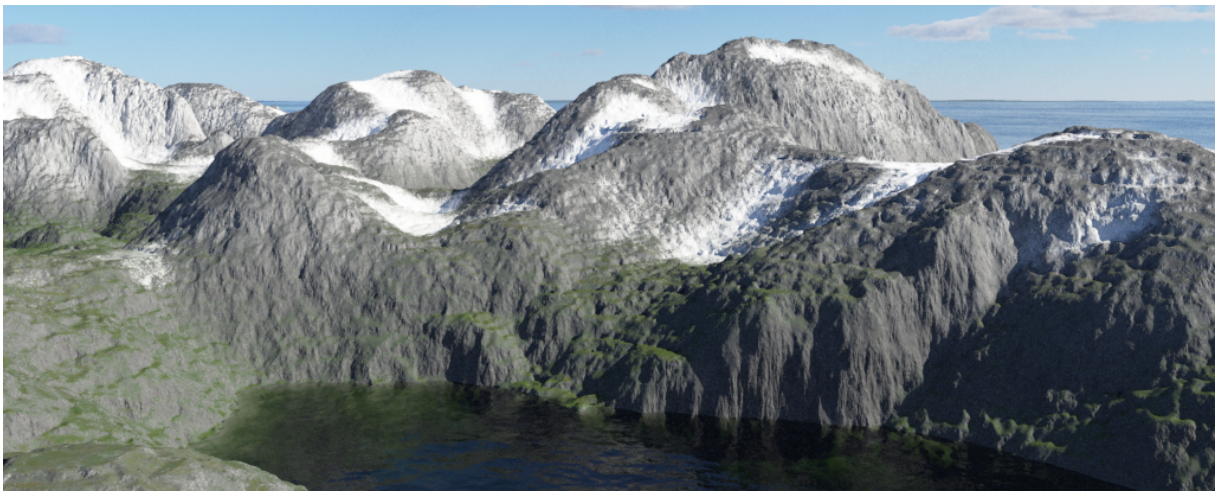

Figure 1: Rendition of a terrain generated using the proposed method.

ABSTRACT

This paper proposes a novel adaptive multi-resolution framework for generating terrains. Our framework combines diffusion-based generative network and novel frequency separated terrain features for terrain patch generation. Additionally, we propose to leverage learnable terrain super-resolution for enhancing generated terrain patch followed by novel kernel-based blending of these patches using Perlin noise to generate infinite terrain with realistic terrain features. We provide a comprehensive quantitative and qualitative evaluation of the proposed framework.

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**; **Machine learning**; **Simulation types and techniques**.

KEYWORDS

Terrain Generation, Terrain Super-Resolution, Diffusion Model, Perlin Noise

ICVGIP'22, December 8–10, 2022, Gandhinagar, India

ACM ISBN 978-1-4503-9822-0/22/12…$15.00
https://doi.org/10.1145/3571600.3571657

ACM Reference Format:
Aryamaan Jain, Avinash Sharma, and K S Rajan. 2022. Adaptive & Multi-Resolution Procedural Infinite Terrain Generation with Diffusion Models and Perlin Noise. In *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'22), December 8–10, 2022, Gandhinagar, India.* ACM, New York, NY, USA, Article 57, 9 pages. https://doi.org/10.1145/3571600.3571657

1 INTRODUCTION

Terrain generation is a classical use-case in the computer graphics community (dates back to four decades [7]) largely driven by gaming and simulation applications. Procedural and artist-driven terrain generation are two popular lines of thought explored in the existing literature [33]. In procedural terrain generation, we aim to algorithmically generate height maps of landmasses such as mountains or deserts. Many domains such as digital gaming, animated movies and architectural models adopt this approach, e.g., popular "open-world" games such as *Minecraft* and *No Man's Sky* need to render infinitely large terrains and hence employ procedural generation techniques. Furthermore, due to the large variation in hardware capability at the gamer's end, it is desired to have a solution with the ability to adaptively generated terrains of varying quality (i.e., multi-resolution terrains).

Traditionally, terrain generation has been attempted using functions like Perlin [31] or Simplex[11] noise. This is to date a popular technique in digital gaming. However, these techniques provide minimal control largely restricted to choosing noise parameters and hence lack real-world terrain features. The real-world terrains are

captured traditionally using microwave imaging (typically sensor over satellites) or recently using LiDAR-based sensing [25]. These digitized terrains are thus represented and stored as Digitial Elevation Models (DEMs) where a raster grid stores per pixel elevation of the discretized terrain.

With the advent of deep learning technology, many generative modelling approaches such as Generative Adversarial Networks (GAN) [9] and Variational Auto Encoders (VAE) [18] have been employed for the task of terrain generation [10, 36] by learning over the real world DEM data. Their key advantage over traditional noise modelling techniques is that they enable learning from a large set of publicly available realistic terrain data [8, 15]. This brings realism to generated terrains. Nevertheless, their usage has been largely limited to producing limited size *tiles/patch* of terrain at a single resolution [27], which greatly limits their applications to open-world games and simulations where large-scale continuous infinite terrain generation is desired. Another related use-case of terrain enhancement/super-resolution is also well attempted with deep learning framework [20]. In regard to deep generative models, recently diffusion-based technique is getting popular where an iterative Markov modelling is proposed for learning data distribution and employed for the task of generating realistic images. The diffusion-based formalism is claimed to outperform other existing generative techniques [5].

In this paper, we propose a framework to generate infinite terrains at multiple resolutions adaptively. Our framework combines diffusion-based generative network [5, 28] and novel frequency separated terrain features along with learnable terrain super-resolution equipped enhancement followed by novel Kernel-based Blending that uses Perlin noise [31] to generate infinite terrain with realistic terrain features. More specifically, our framework consists of training and inference phases. As part of the training phase, we propose to separate multiple spatial frequencies of terrain features extracted from real-world data and independently employ diffusion model based learning of respective data distributions (at associated spatial frequency). Additionally, we also learn a terrain super-resolution model over the same dataset in this phase. In the inference phase, we propose to adaptively sample learnt data distribution from diffusion models (at respective spatial frequencies) and perform fusion to obtain a new terrain patch. Subsequently, we enhance this patch using the trained super-resolution model to enhance realistic terrain features into the patch. Finally, we combine multiple patches using novel kernel blending where Perlin noise help achieve seamless blending near patch boundaries enabling the generation of large and continuous realistic terrains. Figure 1 shows a realistic terrain generated with the proposed framework. We provide a comprehensive quantitative and qualitative evaluation. The code/learnt models are available at our website [1].

The key contributions of our work are: 1) A novel adaptive and multi-resolution framework for generating realistic terrains. 2) Diffusion-based generative modelling of terrains. 3) Generating infinitely large terrain in a learning-based framework. 4) State-of-the-art terrain super-resolution technique.

[1]https://3dcomputervision.github.io/publications/inf_terrain_generation.html

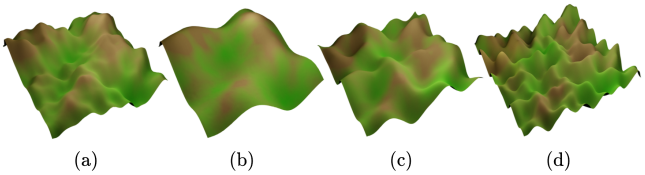

Figure 2: Illustration of fractal Perlin Noise along with its octaves. For this rendition, $f_p = 2.0$, $a_p = 0.5$ and $N_o = 3$.

2 BACKGROUND AND RELATED WORK

Terrain generation can broadly be categorised into traditional and learning based approaches. We can further subdivide traditional approaches into noise-based, example-based and simulation-based techniques. On the other hand, we divide learning based methods based on the applications and techniques they use.

Noise based approaches are among the most commonly used methods due to their simplicity. Noise based functions are mainly of two types, value noise in which we interpolate between random values at fixed grid points and gradient noise in which we interpolate between random gradients in fixed grid points. Examples of noise based approaches are Perlin Noise [31, 32], Simplex Noise [11] or diamond square algorithm [6]. Perlin noise is a type of gradient noise and has been used extensively to generate terrains [21, 29] due to its efficiency and simplicity. Perlin noise uses random gradient values at grid points. For any point within the square grid, we calculate the dot product of the directional vector from the grid corner to the point with the random gradient vector on the grid. We then smoothly interpolate between the dot products to determine the elevation value at the given point. Let us denote Perlin noise by $\text{noise}(x, y)$, such that we get an elevation value at sampled $x, y \in \mathbb{R}^2$. We use fractional Brownian motion (fBm) [23, 30] to fuse multiple frequencies of Perlin noise. Let us denote the frequency of fBm for Perlin noise by $f_p \in [1, \infty)$, persistence $a_p \in [0, 1]$ and the number of octaves as $N_o \in [1, \infty)$. Then fractal Perlin noise is given by,

$$\sum_{i=1}^{N_o} a_p^i \times \text{noise}(f_p^i \times x, f_p^i \times y) \tag{1}$$

Figure 2 renders a terrain corresponding to Equation 1. Figure 2 (b), (c) and (d) correspond to the sum terms for i=1,2 and 3 called octaves o_1, o_2 and o_3 of Perlin noise respectively whereas Figure 2 (a) is the fBm summation of those terms as given by Equation 1. This equation relies on the fractal property of terrains, that is terrains exist as self-repeating structures at different scales and frequencies. This property will be exploited in our framework.

Simulation based methods consist of erosion tools [3, 4, 26], such that given a terrain, an erosion process is simulated over it to produce natural terrain patterns. Whereas example based methods use examples or sketches of terrains to generate terrains [13, 37].

Learning based methods for terrain generation were primarily dominated by GAN [9] and VAE [18] based methods. GANs were introduced as a generative method with two networks, a generator and a discriminator trained together to optimize for a saddle point solving a minimax optimization problem to learn a distribution $p(x)$. In general, GANs are hard to train due to their formulation.

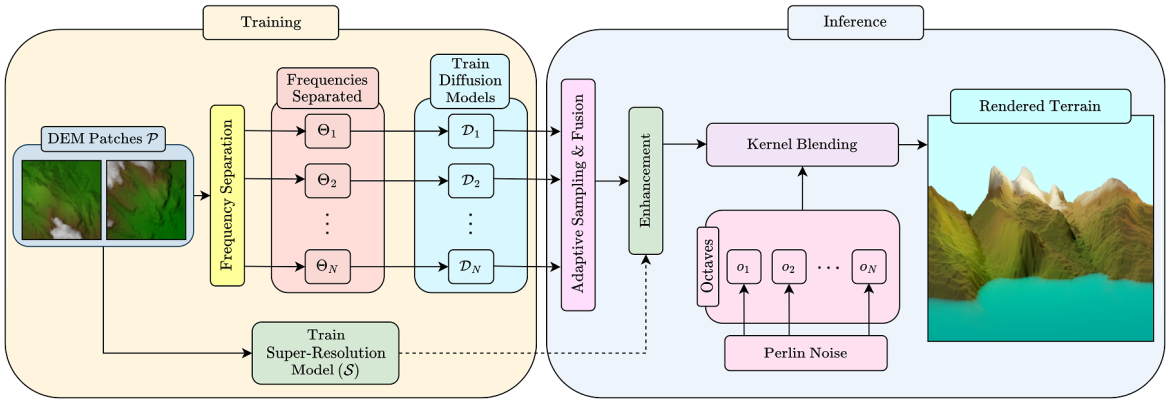

Figure 3: Overview of the proposed framework consisting of the training and inference phases.

Conditional GANs [24] are an extension of GAN which learn a conditional distribution $p(x|c)$. Pip2Pix [16] is a conditional GAN image to image network which have been employed by [10] to convert user inputs such as ridges and valley into terrain. Similarly, [36] used an adversarial loss to amplify terrain with style components. VAE on the other hand optimize for maximizing an evidence lower bound and in general have lower quality but higher diversity than GAN. VAEs learn a latent space which can be manipulated meaningfully and can be used for editing. [27] employed this to propose a framework consisting of a VAE and a GAN for terrain editing.

The diffusion models are another class of generative models introduced in the seminal work by [34]. These models were further improved by [5, 14, 28] until recently when they eventually beat GANs on image synthesis task. Diffusion models work by first adding noise to a data sample and then learn to denoise it. If we keep adding noise following some constraints, we end up with an isotropic Gaussian distribution after T steps of adding the noise. Subsequently, when we want to sample a data point from a desired distribution, we can sample a Gaussian and denoise it to produce the sample. Diffusion models have not yet been used for generating terrains.

Terrain super-resolution is the process of converting a low resolution terrain to a high resolution terrain. This problem was first tackled using a deep learning based solution by [1]. They employed an ortho-photo and a depth-map together to super-resolve the depth map using a multi-scale approach. Their baseline was outperformed by [19] using attention feedback networks. [20] attempted the same task with a depth-map alone using feedback neural networks and achieved on-par performance on some regions compared to [1].

3 METHOD

3.1 Overview & Notations

Our dataset consists of terrain represented as a Digital Elevation Model (DEM), which is a grid raster form of storing the terrain data. Each DEM is a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ representing the height/elevation at any given point in the raster-grid. The DEM is usually large and hence divided into a set of smaller patches for easy processing.

Figure 3 gives an overview of the proposed framework divided into two phases. In the training phase, we assume the availability of DEM dataset where we divide large DEMs into multiple smaller uniform size patches ρ_i and the set of all patches obtained from the dataset is represented as $\mathcal{P}$. Subsequently, we separate each patch $\rho_i \in \mathcal{P}$ into N of its constituent spatial frequencies $\mu_{1_i} \dots \mu_{N_i}$ using Fourier transform such that $\mu_{j_i} \in \Theta_j \forall i$. Hence, Θ_j is the set containing all patches of frequency j and N is a hyper-parameter of the framework. Furthermore, we resize the patches in $\Theta_1 \dots \Theta_N$ such that the size of the patches increases from 1 to N. Finally, we train N separate diffusion models $\mathcal{D}_1 \dots \mathcal{D}_N$ with $\Theta_1 \dots \Theta_N$, respectively. We also train a separate super-resolution model $\mathcal{S}$ on $\mathcal{P}$ to enhance generated patches.

As part of the inference phase, we adaptively sample patches from the first k diffusion models, where $0 \leq k \leq N$. We can increase k adaptively upon the need for more details in the terrain, thereby providing a multi-resolution adaptive output. The sampled set of patches $\mu'_{1_i} \dots \mu'_{k_i}$ are further employed with bicubic up-sampling to be brought to the same size and fused together with fractional Brownian motion (fBm) [23] to generate patch ρ'_i. The generated patch is further enhanced by employing the trained super-resolution model ($\mathcal{S}$) to yield patch $\rho'_{i_{SR}}$. Finally, for a smooth transition between the generated patches $\rho'_{1_{SR}}, \rho'_{2_{SR}}, \dots$, we combine fractal Perlin noise with the generated patches using a derived kernel G. The detailed description of relevant modules in our framework is presented below.

3.2 Frequency Separation

We separate each patch $\rho_i \in \mathcal{P}$ into its N constituent frequencies $\mu_{1_i} \dots \mu_{N_i}$ using Discrete Fourier Transform (DFT) as show in Figure 4. DFT is an image processing method used to convert an image from spatial to frequency domain. We specifically use the Fast Fourier Transform (FFT) algorithm [2] $\mathcal{F}$ for our tasks. Given an input patch $\rho_i \in \mathcal{P}$ of size $M \times M$ in the spatial domain, we convert it into the frequency domain F of the same size as $F = \mathcal{F}(\rho_i)$ given by,

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \rho_i(x,y) e^{-i\,2\pi\left(\frac{ux}{M} + \frac{vy}{M}\right)} \tag{2}$$

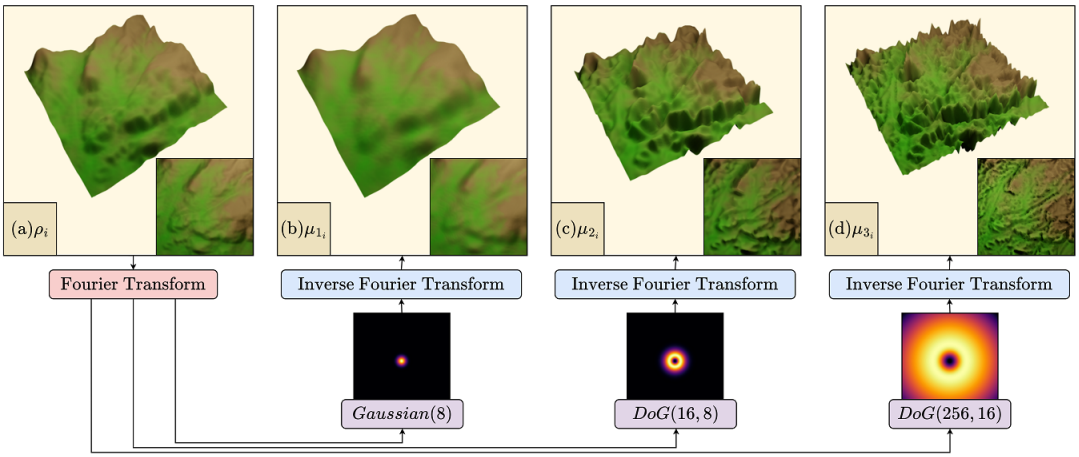

Figure 4: Separation of a patch into its constituent frequencies. The patch dimension is 256×256 and $N = 3$. The Gaussian kernel is of variance 8 and the Gaussians in the DoG kernel are of the variance are given in their parameters.

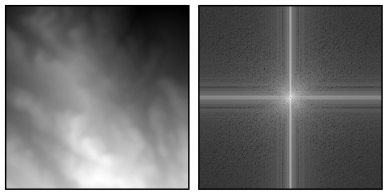

Figure 5: (a) Patch (b) Corresponding log amplitude of the DFT of the patch.

Figure 5 illustrates the patch ρ_i and the corresponding log amplitude of its DFT. After we obtain the frequency domain representation F of the patch ρ_i, we apply a set of kernels to separate its frequencies at varying levels. We use a Gaussian kernel for the lowest frequency and Difference of Gaussian (DoG) kernels for higher frequencies as illustrated in Figure 4. These kernels mask F as Gaussian $\circ F$ or DoG $\circ F$ where $\circ$ represents the Hadamard product which aids in separation of the frequencies. After applying the kernels, we convert the resultant patches back to the spatial domain using Inverse FFT $\mathcal{F}^{-1}$ giving us the constituent patches at successively increasing spatial frequencies $\mu_{1_i} \dots \mu_{N_i}$. Separating a patch into its constituent spatial frequencies will aid in providing a multi-resolution and adaptive framework for producing terrains. One can observe that the extracted spatial frequency patches (shown in Figure 4) are conceptually very similar to the octaves depicted in Figure 2.

3.3 Diffusion Models

Diffusion models generate samples from a distribution by learning successive denoising starting from from a noisy sample at timestep T. Particularly, each sample in the order $\mu_j^T, \mu_j^{T-1}, \dots, \mu_j^1, \mu_j^0$ is closer to the desired distribution, where the superscript gives the timestep of the diffusion process. We adapt our model from [5] which consists of a forward process $q(\mu_j^t|\mu_j^{t-1})$ which adds noise to the given sample and a reverse process $p_\theta(\mu_j^{t-1}|\mu_j^t)$ which learn to denoise the sample. The reverse process is parameterized by θ which is learnt. For a given terrain frequency distribution sample $\mu_j^0 \sim \Theta_j$, we define the forward process as,

$$q(\mu_j^k|\mu_j^0) = \prod_{t=1}^{k} q(\mu_j^t|\mu_j^{t-1}) = \prod_{t=1}^{k} \mathcal{N}(\sqrt{1-\beta_t}\mu_j^{t-1}, \beta_t I)) \quad (3)$$

Here β is increased as per a linear or cosine schedule in our experiments. Since we iteratively scale and sample from a Normal distribution as illustrated in Equation 3, we get an isotropic Gaussian sample at timestep T. We define the reverse process as:,

$$p_\theta(\mu_j^{t-1}|\mu_j^t) = \mathcal{N}(\mathrm{M}_\theta(\mu_j^t, t), \Sigma_\theta(\mu_j^t, t)). \quad (4)$$

At each time step, we predict the mean and covariance of the previous timestep using the parameterized functions M_θ and Σ_θ parameterized by θ. We sample from a Normal distribution with the predicted mean and covariance iteratively until timestep 0, which gives us a sample from the desired terrain frequency distribution. We specifically predict the noise added, optimizing for the L_2 norm of the true and predicted noise $\mathbb{E}[||\epsilon - \epsilon_\theta(\mu_j^t, t)||_2^2]$, where ϵ is the true added noise and ϵ_θ is a parameterized function predicting the noise added. We use a UNet architecture.

We train N diffusion models $\mathcal{D}_i$, $i \in [1, N]$ with Θ_i. Each successive model is trained on a larger patch size, that is, the size of the patch produced by the model $\mathcal{D}_i \propto i$. In particular, we keep $N = 3$ in our experiments with the patch sizes varying as 64×64, 128×128 and 256×256 for $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ respectively. Decreasing the patch size with decreasing frequency aids in adaptive sampling discussed in the sections to follow. We use diffusion models for learning the distribution of terrains because of their superior quality in terrain generation compared to other methods, as discussed in the section 4. An illustration of the process of terrain patch generation with diffusion model is provided in Figure 6.

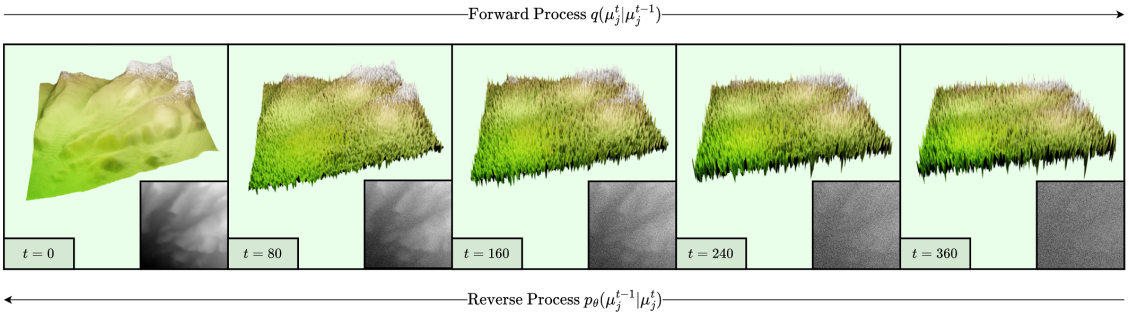

Figure 6: The working of a diffusion model for terrains. t represents the timestep of the diffusion process. Here we adopt linear schedule with β varying between 0.001 and 0.02.

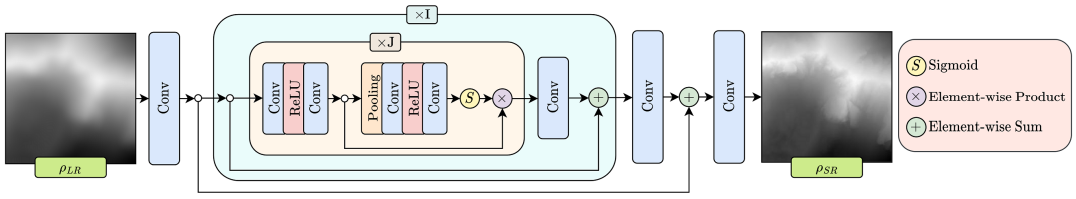

Figure 7: Architectural diagram of the proposed terrain super-resolution model TRCAN.

3.4 Terrain Super-Resolution

Modelling: We propose to adapt RCAN [35] model initially proposed for image super-resolution for the task of terrain enhancement. Henceforth, we will refer to this network as Terrain Residual Channel Attention Network (TRCAN). Figure 7 provides an overview of the architecture of TRCAN. This network uses a residual in residual structure which helps in making very deep networks avoiding vanishing gradients and increasing the receptive field. The input to TRCAN model while training is a low-resolution patch ρ_{LR} and the output is the super-resolved patch ρ_{SR} corresponding to the ground truth high-resolution patch ρ_{HR}.

The key difference between our method compared with RCAN [35] is the head and tail convolutional blocks in the network. Whereas RCAN proposes to pass the image in its low-resolution dimension through the network and upsample towards the end, we cannot employ the same for the task of terrain super-resolution. This difference primarily arises due to the difference in the ways of the processing of the datasets where terrains need the input and output to be of the same dimensions to find low-resolution and high-resolution patch correspondences. We subsequently modify the head and the tail convolutional blocks to account for the same.

We use Adam optimizer [17] along with L_1 loss to optimize the parameters of the network given by,

$$L_1(\theta) = ||\rho_{HR} - \rho_{SR}||_1^1 \tag{5}$$

Post-Processing: As compared to regular RGB images, DEMs are often large in resolutions, for example, Cimavertana region in our dataset is a 5250×3900 raster. Thus, the large DEMs are split into smaller patches for easy processing with typical sizes lying in the range 256×256. In the simplest form, all the patches are split without any overlap, enhanced independently and put together side by side without overlap to obtain the final super-resolved DEM. [20] observed that this was not the optimal way to obtain the final DEM and proposed to split the patches with some overlap along the edges (25% in their case), super-resolve and then average them along the edges. We observed that the scope of improving the results is not limited to just the patch boundaries but throughout the patch and therefore propose a new stride-based post-processing technique. We extract patches from the larger DEM at strides of s, which is a hyperparameter. We super-resolve all the patches extracted at stride s and pool their values at the overlapping regions to obtain the final result. This acts as an implicit ensemble thereby improving the RMSE of the super-resolved DEM. This can be seen as a generalization of the method proposed by [20], with a 25% overlap corresponding to a stride of 192 for a 256×256 tile. We call this model TRCAN+.

3.5 Adaptive Sampling, Fusion & Enhancement

We aim to produce a multi-resolution and adaptive framework to generate terrain. A naïve approach for generating terrains would be to learn the DEM patch $\mathcal{P}$ distribution rather than the distributions of its constituent frequencies Θ_j. In this approach, we would need to learn the patches at a constant resolution, and generate those patches at the same resolution regardless of any constraints. This might be wasteful of resources because high levels of detail for far away terrains may be unnecessary in a real-time setting. For example, in a first person view, we would like terrains close by to be of higher quality than areas of terrains far away. We would also like the details on the terrain to increase as we move closer begging

a multi-resolution solution. Similarly, we would want an algorithm that could function on computational constraints, that is work on PCs with lower specifications too. Therefore, we desire a solution which is adaptive, multi-resolution and realistic.

Firstly, we achieve adaptive sampling by varying the number of models we sample for generating a patch. Particularly, we sample $k \leq N$ diffusion models to produce $\mu'_1 \dots \mu'_k$. For example, in a first person view, we would set k high for a nearby point and keep it low for a far away point interpreting it as a technique for Continuous Levels of Detail (CLOD) in terrains. Another use case would be to set an upper-limit for k to respect computational constraints, with k set to 0 for the slowest PCs corresponding to just Perlin Noise which has efficient implementations available. Since for lower frequencies, the models are learnt on lower resolutions, sampling them would be faster. Once we obtain $\mu'_1 \dots \mu'_k$, we perform bi-cubic interpolation to bring all of the generated frequency separated patches to the same resolution and then fuse them using fBm, a method which is inspired by Perlin noise. Let the persistence of generated patches be denoted by $a_\mu \in [0, 1]$ which scales down higher frequency details, the fBm fusion equation would be given by,

$$\rho' = \sum_{i=1, \mu'_i \sim \mathcal{D}_i}^{k} a_\mu^i \mu'_i \tag{6}$$

This is possible only because we have samples at different frequencies and would not be possible if we used techniques incorporating mipmapping. We can compare this equation with Equation 1. Note that Equation 6 can be computed in an online manner, that is, as we desire more details, we only have to sample models $\mathcal{D}_i$ such that $i > k$ and update the previously produced patch. Equation 6 would thereby enable us to produce multi-resolution terrain patches ρ'_i as discussed in subsection 4.2

We further deal with step-size resizing in our trained diffusion models to improve our framework adaptivity. Given that we have diffusion models trained on T steps, while sampling we can reduce the number of steps to $K \leq T$ [28]. For this, K linearly spaced integers between 1 and T can be used as the input to the diffusion model. This improves the sampling speed of our model by decreasing the time to sample linearly with K. Alas, this comes at the cost of quality. This adds as another use-case for adaptive terrain sampling, which can adjust to computation constraints. An illustration and discussion on this is provided in subsection 4.2.

Finally, we use our enhancement module to add details to the generated terrain patch ρ'. Specifically, the super-resolution model $\mathcal{S}$ takes the unrefined patch ρ' and adds details to produce $\rho'_{SR} = \mathcal{S}(\rho')$.

3.6 Kernel Blending

One of the last challenges in producing infinitely large terrains is a smooth transition among the patches ρ'_{SR_i} we generated in the previous steps. Simply placing the tiles produced in a grid gives rise to discontinuities along the patch boundary, as illustrated in Figure 8 (a) highlighted in red. This is because the terrain tiles are produced independently and their edges do not line up with each other. On the other hand, Perlin noise function is inherently

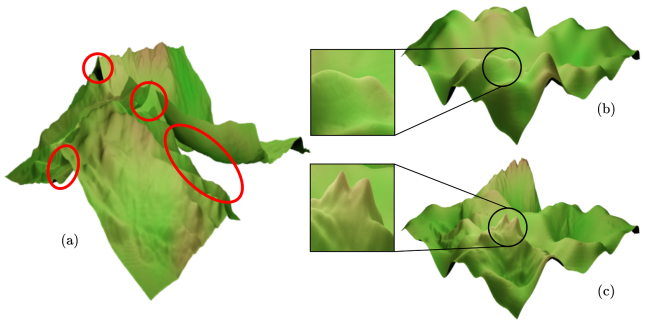

Figure 8: (a) Tiled terrain patches (b) Fractal Perlin noise (c) Kernel blended terrain.

continuous in $\mathbb{R}^2$ because it smoothly interpolates between pseudo-random gradients at fixed intervals at tile vertices. Whereas, as we can observe in Figure 8 (b), Perlin noise does not possess real terrain features and hence its divergence from the distribution of terrain is high. We propose to take the best of both, a learnt distribution from diffusion models (Figure 8 (a)) and the infinite continuity of Perlin noise (Figure 8 (b)) by blending them using a kernel to produce infinite terrains with learnt details as illustrated in Figure 8 (c). The difference in details are highlighted in comparison of (b) and (c), with (b) showing a much smoother surface lacking real world detail. We do not observe boundary artifacts in (c) which were present in (a).

Let g be a 1-D kernel and $t \in [0, 1]$ represent the domain of the kernel such that $g(t) \in [0, 1]$ is the range. We want the kernel to satisfy conditions (1) $g'(t = 0) = 0$ and (2) $g'(t = 1) = 0$ for the continuity along the end-points given that we will tile this kernel, (3) $g(t = 0) = 0$ and (4) $g(t = 1) = 0$ to make sure that fractal Perlin noise is dominant at the tile edges to provide continuity and (5) $g(t = \frac{1}{2}) = 1$ such that the sampled terrain is dominant where continuity is not a concern. We require an order four polynomial to satisfy the five conditions. Let $g(t) = \sum_{i=0}^{4} a_i t^i$ be the template polynomial. Upon simplifying with constraints (1) to (5), we get the linear system of equations,

$$\begin{bmatrix} 1 & 1 & 1 \\ 4 & 3 & 2 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} a_4 \\ a_3 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 16 \end{bmatrix} \tag{7}$$

The solution of this gives us the kernel $g(t) = 16t^4 - 32t^2 + 16t^2$, the corresponding graph is plotted in Figure 9. This kernel resembles a Gaussian but upon experiments with a Gaussian kernel, we observed edge artifacts due to the tails of Gaussian being non-zero. We take the outer product of the kernel with itself which gives us the 2-D kernel $G = g \otimes g$. We blend ρ'_{SR} and fractal Perlin noise fpn for the desired domain as,

$$G \circ \rho'_{SR} + \lambda(1 - G) \circ \text{fpn} \tag{8}$$

where $\circ$ denotes the Hadamard product and λ is a scaling hyper-parameter that we best found to work in the range $[0.75, 1]$. An example of the resulting terrain is illustrated in Figure 8 (c), where Figure 8 (a) corresponds to ρ'_{SR} and Figure 8 (b) corresponds to fpn.

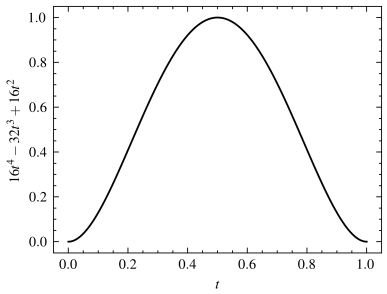

Figure 9: The 1-dimensional derived kernel g.

4 EXPERIMENTS AND RESULTS

4.1 Dataset

We use the same dataset as [1, 20, 27] for fair comparison with state-of-the-art terrain generation and terrain super-resolution. This dataset is publicly available and consists of high resolution DEMs of 2m spatial resolution from the regions of Pyrenees [15] and Tyrol [8], which cover an area of 643 km^2 and 304 km^2 respectively. We divide the DEMs into tiles of 256×256 for processing in the pipeline.

4.2 Diffusion Model, Adaptive Sampling & Fusion

Implementation Details: We train our diffusion models with number of steps $K = 1000$. We use a batch size of 16 along with a cosine β scheduler. We learn the covariance and keep the learning rate fixed at 10^{-4} optimizing with the Adam optimizer [17]. We train for 2^{16} iterations on Nvidia GeForce RTX 2080 Ti. We keep these parameters fixed for all the frequencies. We compare our proposed diffusion based model to a set of baselines. The first baseline is the improved Perlin Noise [32] which we implemented ourselves. We set the number of octaves to 3. The second baseline is a GAN model [9] for which we use the implementation of deep convolutional GAN provided by [22]. Our batch size was set to 64, learning rate 2×10^{-4}, latent dimension 100, trained for 200 epochs on Nvidia GeForce RTX 2080 Ti and optimized using the Adam optimizer [17]. Our final baseline is a model which is composed of a VAE and a GAN [27]. This is a conditional model meant for terrain generation and manipulation. We use the pre-trained weights provided by the authors to generate samples given the test set. This model was trained on the same dataset as ours. For fair comparison, we set $N = 1$ for diffusion models.

Results: We generate samples from various baseline models (explained before) and compare it to the ground truth dataset using the FID metric [12]. The results given in Table 1 show that diffusion models outperform other methods in terrain generation.

In terms of qualitative understanding of adaptive terrain generation, Figure 11 displays terrains rendered at varying values of diffusion sampling steps (K) in which we can observe a slight drop in the finer details as we decrease K. Nevertheless, this decrease in quality can be acceptable as it yields faster generation, with $K = 1000$ taking 35s and $K = 10$ taking 5s on an Intel Xeon CPU as sampling time reduces linearly with K in diffusion models. Since terrain data is much more unstructured compared to natural images or facial data, slight deviation from the actual distribution are not caught visually, hence reducing K to improve sampling speed would not be as detrimental as it would be with other domains.

Table 1: Comparison of Fractal Perlin [32], GAN [22], VAE+GAN [27] and the proposed diffusion based modeling using the FID ↓ metric for terrain generation.

	Fractal Perlin	GAN	VAE+GAN	Diffusion
FID ↓	335.851	204.365	119.124	**54.444**

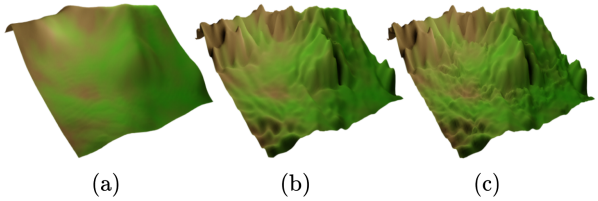

Figure 10: An illustration our fusion strategy (following Equation 6) with $a_\mu = 0.5$. (a), (b) and (c) correspond to $k = 1, 2$ and 3.

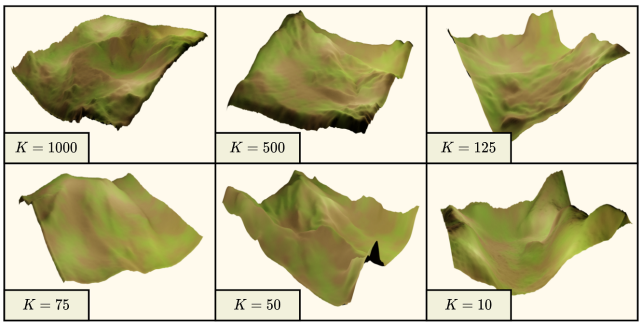

Figure 11: Terrains sampled with diffusion model by varying the number of sampling steps K for $N = 1$.

In regard to qualitative evaluation of fusion multiple patches generated by respective learnt diffusion models for different spatial frequencies is shown in Figure 10, where we can observe the finer details increase with an increase in k (following Equation 6).

4.3 Super-Resolution model

Implementation Details: The architectural diagram for our proposed method is given in Figure 7, corresponding to which we found the values of $I = 8$ and $J = 16$ to work best for us. We used average pooling in our pooling layer in the process to get the attention weights. We used Adam optimizer [17] with a learning rate (LR) of 10^{-4} and a step LR scheduler reducing the LR by 5% every 2 epochs. We train for 256 epochs on Nvidia GeForce RTX 2080 Ti.

Table 2: Comparison of Bicubic Upsampling, FCND [1], DSRFB/DSRFO [20] and our proposed TRCAN/TRCAN+ using RMSE (in meter, ↓) / PSNR (in dB, ↑) for 8x terrain super-resolution.

Region	Bicubic	FCND	DSRFB	DSRFO	TRCAN	TRCAN+
Bassiero	1.406/60.5	1.146/62.261	1.091/62.687	1.083/62.752	**1.086/62.728**	**1.077/62.807**
Forcanada	1.632/58.6	1.326/60.383	1.270/60.761	1.259/60.837	**1.260/60.828**	**1.248/60.909**
Durrenstein	1.445/59.5	0.957/63.076	0.884/63.766	0.868/63.924	**0.869/63.915**	**0.847/64.138**
Monte Magro	0.917/67.2	0.632/70.461	0.589/71.081	0.581/71.196	**0.584/71.144**	**0.574/71.293**

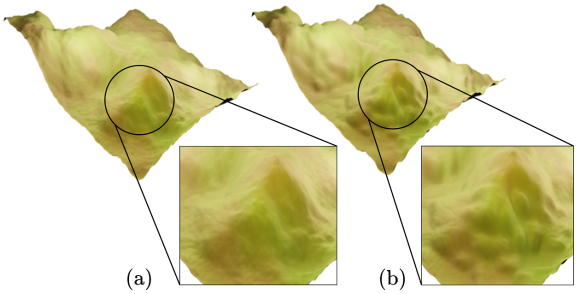

Figure 12: Terrain enhancement where (a) is enhanced to (b).

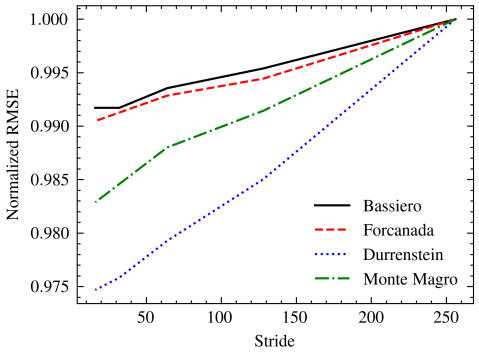

Figure 13: Effect of stride on the RMSE.

Results: Figure 12 illustrate qualitative result of terrain enhancement, where we can observe that (a) is the generated terrain (from fusion of multi spatial frequency patches) is smoothed out whereas fine details like sharper edges are present in enhanced terrain obtained by employing terrain super-resolution as shown in (b).

In terms of quantitative evaluation, we compare the RMSE and PSNR of our model with bicubic upsampling and two baselines models, all of which were tested on the same dataset as ours. The first baseline model FCND [1] proposed a multi-scale architecture with the possibility of combination with aerial-imagery. We use their method with just depth for fair comparison. The second baseline model [20] proposed a feedback neural network based architecture DSRFB along with extention with post-processing DSRFO. Table 2 compares our method with the baselines using RMSE and PSNR and establishes a new state-of-the-art in 8x terrain super-resolution. The results are reported for the test set. DSRFB should be compared with TRCAN because they are tested without post-processing and DSRFO with TRCAN+ with post-processing. TRCAN+ used a stride of 16.

We further experiment with the effect of stride on the RMSE. Figure 13 illustrates that there is an approximately linear relationship between the two. This can be seen as an ensemble where the variance in the output reduces with increasing number of samples, which is inversely proportional to the stride. The RMSE has been normalized in this plot such that the RMSE of the maximum stride is 1.0 for each region for easy visualization.

4.4 User Study

We conduct a user study with 30 domain expert participants. We show each participant 5 renders each of terrain generated via fractal Perlin Noise, our method and the ground truth dataset and ask them to rate the terrains based on realism and aesthetics out of 5. We report the mean and standard deviation (STD) of the scores in Table 3. We observe the expected progression in mean and STD scores with the highest scoring ground truth followed by our method and then fractal Perlin noise. We can observe that Mean rating of the generated terrains obtained with Perlin noise is significantly lower as compare ratings of terrains generated with our method, which is very close to ratings given to real terrains.

Table 3: Comparison of Fractal Perlin Noise [32], ground truth data and our proposed method based on user rating for terrain generation.

	Fractal Perlin	Ground Truth	Our
Mean	1.948	3.931	3.465
STD	1.085	1.298	1.074

5 CONCLUSION

We propose a novel framework for terrain generation introducing concepts such as frequency separation using Fourier transform, kernel blending and fBm fusion for generating patches which gives a new perspective to terrain generation in a learning based framework. Our framework is adaptive and multi-resolution for generating learning based infinite terrains procedurally which might contribute greatly to the gaming and simulation community. We test and report superior qualitative and quantitative results for diffusion based models and show their applicability for terrain generation. Along with that, we propose a state-of-the-art terrain super-resolution model with a novel post-processing technique which we employ for terrain enhancement.

REFERENCES

[1] Oscar Argudo, Antoni Chica, and Carlos Andujar. 2018. Terrain super-resolution through aerial imagery and fully convolutional networks. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 101–110.

[2] E Oran Brigham and RE Morrow. 1967. The fast Fourier transform. *IEEE spectrum* 4, 12 (1967), 63–70.

[3] Norishige Chiba, Kazunobu Muraoka, and Kunihiko Fujita. 1998. An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation* 9, 4 (1998), 185–194.

[4] Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. 2017. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

[5] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 8780–8794.

[6] Deng Fang. 2009. The study of terrain simulation based on fractal. *WSEAS Transactions on Computers* 8, 1 (2009), 133–142.

[7] Alain Fournier, Don Fussell, and Loren Carpenter. 1982. Computer rendering of stochastic models. *Commun. ACM* 25, 6 (1982), 371–384.

[8] Südtiroler Bürgernetz GeoKatalog. [n.d.]. Tyrol. http://geokatalog.buergernetz.bz.it/geokatalog/

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[10] Éric Guérin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoît Martinez. 2017. Interactive example-based terrain authoring with conditional generative adversarial networks. *Acm Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

[11] Stefan Gustavson. 2005. Simplex noise demystified. *Linköping University, Linköping, Sweden, Research Report* (2005).

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

[13] Houssam Hnaidi, Eric Guérin, Samir Akkouche, Adrien Peytavie, and Eric Galin. 2010. Feature based terrain generation using diffusion equation. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 2179–2186.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

[15] Institut Cartogràfic i Geològic de Catalunya. [n.d.]. ICGC – VISSIR3. http://www.icc.cat/vissir3/

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[19] Ashish Kubade, Diptiben Patel, Avinash Sharma, and KS Rajan. 2020. AFN: Attentional Feedback Network Based 3D Terrain Super-Resolution. In *Proceedings of the Asian Conference on Computer Vision*.

[20] Ashish A Kubade, Avinash Sharma, and KS Rajan. 2020. Feedback Neural Network based Super-resolution of DEM for generating high fidelity features. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 1671–1674.

[21] Huailiang Li, Xianguo Tuo, Yao Liu, and Xin Jiang. 2015. A parallel algorithm using Perlin noise superposition method for terrain generation based on CUDA architecture. In *International Conference on Materials Engineering and Information Technology Applications (MEITA 2015)*. Atlantis Press, 967–974.

[22] Erik Linder-Norén. 2021. GitHub - eriklindernoren/PyTorch-GAN: PyTorch implementations of Generative Adversarial Networks. — github.com. https://github.com/eriklindernoren/PyTorch-GAN. [Accessed 02-Sep-2022].

[23] Benoit B Mandelbrot and John W Van Ness. 1968. Fractional Brownian motions, fractional noises and applications. *SIAM review* 10, 4 (1968), 422–437.

[24] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[25] VL Mulder, S De Bruin, Michael E Schaepman, and TR Mayr. 2011. The use of remote sensing in soil and terrain mapping—A review. *Geoderma* 162, 1-2 (2011), 1–19.

[26] F Kenton Musgrave, Craig E Kolb, and Robert S Mace. 1989. The synthesis and rendering of eroded fractal terrains. *ACM Siggraph Computer Graphics* 23, 3 (1989), 41–50.

[27] Shanthika Naik, Aryamaan Jain, Avinash Sharma, and K S Rajan. 2022. Deep Generative Framework for Interactive 3D Terrain Authoring and Manipulation. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. 6410–6413. https://doi.org/10.1109/IGARSS46834.2022.9884954

[28] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*. PMLR, 8162–8171.

[29] Ian Parberry. 2014. Designer worlds: Procedural generation of infinite terrain from real-world elevation data. *Journal of Computer Graphics Techniques* 3, 1 (2014).

[30] Pbr-book.org. 2022. https://www.pbr-book.org/3ed-2018/Texture/Noise

[31] Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.

[32] Ken Perlin. 2002. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 681–682.

[33] Thomas J Rose and Anastasios G Bakaoukas. 2016. Algorithms and approaches for procedural terrain generation-a brief review of current techniques. In *2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*. IEEE, 1–2.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. PMLR, 2256–2265.

[35] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. 2018. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*. 286–301.

[36] Yiwei Zhao, Han Liu, Igor Borovikov, Ahmad Beirami, Maziar Sanjabi, and Kazi Zaman. 2019. Multi-theme generative adversarial terrain amplification. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

[37] Howard Zhou, Jie Sun, Greg Turk, and James M Rehg. 2007. Terrain synthesis from digital elevation models. *IEEE transactions on visualization and computer graphics* 13, 4 (2007), 834–848.