DOI: 10.1111/cgf.14936

Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

Authoring Terrains with Spatialised Style

Simon Perche[1], Adrien Peytavie[1], Bedrich Benes[2], Eric Galin[1] and Eric Guérin[1]

[1]Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS, UMR5205, F-69621 Villeurbanne, France
[2]Purdue University

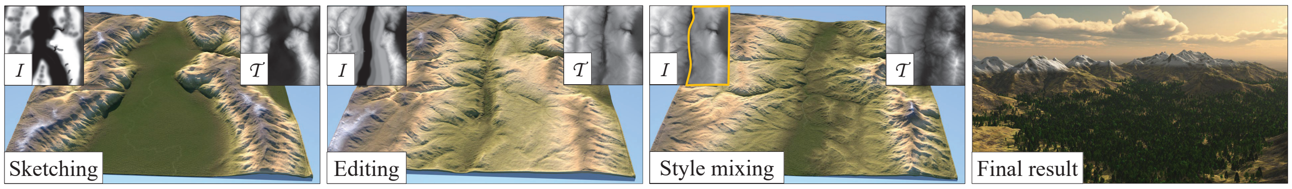

Figure 1: *An editing session showing the result of the inversion of a draft sketch as input, and then some editing is done to turn the canyon into a valley. Finally, the user changed the style of the right part of the scene to emphasise the mountain range.*

Abstract

Various terrain modelling methods have been proposed for the past decades, providing efficient and often interactive authoring tools. However, they seldom include any notion of style, which is critical for designers in the entertainment industry. We introduce a new generative network method that bridges the gap between automatic terrain synthesis and authoring, providing a versatile set of authoring tools allowing spatialised style. We build upon the StyleGAN2 architecture and extend it with authoring tools. Given an input sketch or existing elevation map, our method generates a terrain with features that can be authored, enhanced, and augmented using interactive brushes and style manipulation tools. The strength of our approach lies in the versatility and interoperability of the different tools. We validate our method quantitatively with drainage calculation against other previous techniques and qualitatively by asking users to follow a prompt or freely create a terrain.

CCS Concepts
• ***Computing methodologies*** → *Shape inference;* ***Shape modeling;***

1. Introduction

Realistic and controllable terrain models are necessary for creating virtual worlds. Existing approaches encompass a variety of methods, including procedural generation, physically-based erosion simulations, and example-based synthesis. Some of the proposed methods provide a varying level of control, allowing the user to generate, edit, or modify synthetic terrains, but do not incorporate the concept of style. Conversely, existing methods accounting for style offer limited control to the designer: they cannot apply different styles to the same terrain at selected locations (see Section 2). Both control and style are critical demands from designers in the entertainment industry. In our approach, we spatialise the style and propose a collection of tools to help artists in their creative process.

We define style as related to the user perception of the overall quality and common properties of a terrain or a specific category with similar elevation and landmark characteristics. A key observation is that style has a tremendous impact on the perception of terrains: a highly irregular mountainous landscape from the Karakoram looks radically different from smooth hills in the Appalachians. Style affects all ranges of scales from large-scale geographic structures of hundreds of kilometres, as demonstrated in the orometry-based method proposed in [AGP*19], to landforms of a few meters. This diversity is the consequence of complex natural processes acting over varying temporal and spatial scales, including tectonics, stratification, aeolian, and hydraulic erosion processes. Modelling such complex phenomena is complicated and comes at the price of computationally intensive, involved, and hard-to-control simulations. Most erosion algorithms focus on restricted temporal and spatial scales: they only account for a limited category of phenomena and simple extrapolation of bedrock and sediment materials properties, thus effectively simulating a limited range of landforms.

Another crucial observation is that artists often prefer interactive editing when authoring terrains and therefore favour techniques that allow control at a varying level of spatial scales. While methods exist for interactive modifications of landforms or local style transfer for virtual worlds, little research has been dedicated to combining the concept of style with an interactive edition framework for rapid modelling and enhancement of large-scale complex landscapes. We address these limitations by proposing a novel approach that presents versatile interactive tools for editing terrains, including sketching, copy-and-paste sequence, and details enhancement while providing ways for the designer to define or impose a given style at different levels (Figure 1).

Our method is based on a generative approach trained from real-world examples. From this learning phase, the model is capable of extracting the notion of terrain style out of the examples. An inversion process (called encoder) consisting of finding the latent representation of a given terrain or sketch is later used to generate and edit terrains that match a given style. More precisely, the contributions are: 1) a novel terrain model employing an improved StyleGAN2 architecture [KLA19] that describes and generates high-quality digital elevation models; 2) a variety of tools crucial for terrain authoring and adapted to the model; 3) the introduction of style at different scales and at different locations, which allows for rich context-aware content generation. We demonstrate that our approach can be seamlessly integrated in a production environment with a complete implementation of the model, along with datasets, into a Blender addon that demonstrates interactive editing (see accompanying video). The code and data are available at `https://simonperche.github.io/authoring_terrains_with_spatialised_style`.

2. Related work

Terrain generation methods in Computer Graphics can be classified as procedural generation, physically-based (erosion) simulation, and synthesis from exemplars, which includes deep-learning algorithms (see the review [GGP*19]). Given the identified goal of control, style, and synthesis from real digital elevation models, we briefly review and focus on authoring frameworks that evidence interactivity.

Procedural terrain modelling generally relies on procedural noises, often combined with river network carving, to algorithmically reproduce the self-similarity across scales. The elevation results from summing scaled noise functions [MKM89] or assembling procedurally defined and compactly supported primitives [GGP*15]. They are generally computationally efficient and lend themselves to parallel implementation on graphics hardware. Control typically takes the form of applying noise with brushes [dCB09] or matching curve and point constraints using warping [GMS09] or using diffusion curves [HGA*10]. Another strategy consists in controlling generation by constraining landforms to conform to prescribed low-resolution elevation maps and river networks [GGG*13]. Recently, [GPM*22] introduced gradient-domain editing for terrain modelling, demonstrating its effectiveness for seamless blending of patches, copy-and-paste operations, and generation from control feature points and curves. Unfortunately, these approaches do not consider style and require careful editing and parameter tuning to achieve user intent.

Erosion simulation methods, broadly classified into surface erosion algorithms and geology-based simulations, are in essence difficult to control and provide limited interactive feedback (see [GGP*19] for a complete overview). Recent works attempt to bridge the gap between physically-based simulation and authoring. Event-based surface erosion [CGG*17] simulate material detachment, transport, and deposition, accounting for the presence of vegetation and event scenario and enhancing relief with convincing details such as gorges and ravines as long as the initial input terrain is sufficiently realistic and supplies large-scale landmarks. In contrast, controlled geology-based simulations attempt to reproduce large-scale landforms by prescribing the uplift of the bedrock balanced by different types of erosion, most often the Stream Power erosion [SPF*23] or glacier erosion [CJP*23] forming hanging valleys and quarries. However, those methods often provide limited and indirect control to accord to the underlying erosion equations governing the erosion phenomenon and are computationally intensive, which deters them from interactive authoring.

Example-based methods tackle the realism by combining patches extracted from real-world digital elevation models. Their control is often achieved by structure-sensitive warping to match sketched silhouettes [TEC*14]. Parallel texture-based synthesis [GMM15] modifies the matching process to support style painting, region-based copy-and-paste, and curve and point manipulators. The assembly of patches, even locally geomorphologically correct, is not sufficient for generating globally consistent landscapes. Sparse modelling is another efficient way to generate high-resolution terrains from sketches [GDGP16] guided by exemplars. Recently, Scott *et al.* [SD21] proposed a breaching algorithm interlaced in multi-resolution example-based terrain synthesis to improve hydrological consistency. In contrast, our method encodes hydrological consistency at different scales in the latent space.

Deep-learning algorithms are a specific case of example-based approaches. Guérin *et al.* [GDG*17] first demonstrated the effectiveness of Conditional Generative Adversarial Networks (cGAN) for learning the correspondence between terrains and the sketch maps corollaries containing ridge and river lines and feature points. Zang *et al.* [ZLZ*22] used a modified version of a GAN with low-resolution maps, global style information, and local style maps as input, and discriminators capable of classifying different types of terrains. The generator is based on UNet architecture, and patch-based discriminators allow local style control. Another approach, specialised in style embedding, was proposed in [ZLB*19]. By using a CGAN to insert the embed theme into the generation process, the method can amplify a low-resolution input terrain into a high resolution with style variation. While producing compelling results in terms of style transfer, this approach does not provide any editing mechanisms. Recently, [NJSR] trained a Variational Auto Encoder [KW13] combined with a GAN to generate a heightmap from a low-resolution map coupled to a sketch. Recently, some other works have also shown the possibility to generate textures [ABGT20, KHWM17] and heighmap [PC20] at the same time using GANs.

While providing authoring tools such as sketching and interpolation, existing techniques generate terrain lacking details. We propose a novel method that addresses the blind spot of previous ap-

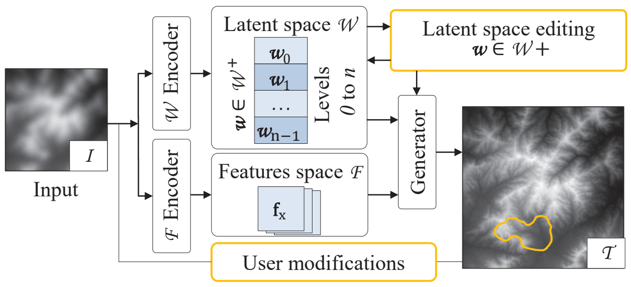

Figure 2: *The input heightfield $\mathcal{I}$ (a high or a low-resolution terrain, or even a sketch) is injected into the encoding process to produce a latent vector representation in $\mathcal{W}^+$, with features maps $\mathbf{f} \in \mathcal{F}$. The generator uses $\mathbf{f}$ as input and one or multiple $\mathbf{w} \in \mathcal{W}^+$ vectors to synthesise a new terrain $\mathcal{T}$. The user may modify the latent space to control the generation process, or directly edit $\mathcal{T}$ and inject it again in the generator.*

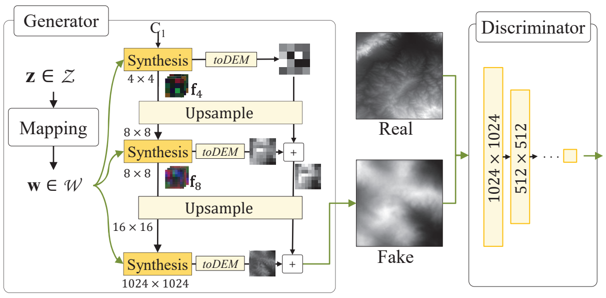

Figure 3: *StyleGAN2 architecture. A vector from the latent space $\mathcal{Z}$ is projected into a second latent space $\mathbf{w} \in \mathcal{W}$ by using a mapping network that disentangles latent directions. $\mathbf{w}$ is then streamed to the generator at various levels of resolution, and the output is sent to a discriminator, which seeks to separate real and fake images. The generator and the discriminator are trained together in a zero-sum game. The results are utilised during the backward propagation phase. The* toDEM *component corresponds to the* toRGB *layer in the original StyleGAN2 which has been modified to handle a single channel image.*

proaches by developing a framework for simultaneous style manipulation and editing.

3. Model

We introduce a deep neural model based on the StyleGAN2 architecture [KLA19] applied to Digital Elevation Models (DEMs), which are terrains represented as discrete heightfields denoted as $\mathcal{T}$. StyleGAN2 takes a large set of input images and encodes their style into latent space. Inspired by this approach, we use latent space as the primary way to represent and manipulate digital terrains.

The framework consists of two main parts depicted in Figure 2. The StyleGAN2 itself is a *generator* trained on a carefully selected and designed data set of DEMs. The generator synthesises a high-resolution terrain $\mathcal{T} = \mathcal{G}(\mathbf{w})$ from its latent representation $\mathbf{w}$. Conversely, an inverter [XZY*] called *encoder* reverts the process by producing a latent vector $\mathbf{w} = \mathcal{E}(\mathcal{I})$ that matches the input $\mathcal{I}$. In some cases, applying the generator again to the inferred latent vector may not reproduce exactly the initial $\mathcal{I}$, which is part of the authoring process, especially when $\mathcal{I}$ was a sketch and not a real terrain, formally: $\mathcal{G}(\mathcal{E}(\mathcal{I})) \neq \mathcal{I}$. Latent space is central in the representation of the style but fails at localising features, which is crucial in our application. Therefore, we extend the encoder capabilities by adding a direct inference of the inner representation of the StyleGAN2 called *feature maps*. The user can interact at multiple stages of the pipeline, from user inputs to direct latent space modification. At every step, the output is directly rendered for preview or, when the result is satisfactory, streamed to the rendering pipeline.

Generator The generation step incrementally builds on the *Generative Adversarial Network* (GAN) architecture introduced in [GPAM*14]. The generator denoted as $\mathcal{G}$ creates a high-resolution terrain. During the training process, we couple $\mathcal{G}$ with another neural network, called the discriminator $\mathcal{D}$, that attempts to detect whether the generated terrain is real or not. Therefore, the generator and the discriminator compete against each other: the generator tries to fool the discriminator by generating images resembling the ones found in the training database, whereas the discriminator tries to distinguish between images generated by the generator from those in the training dataset. This architecture allows the unsupervised training of the network and does not require explicit specification of a loss function.

The introduction of the StyleGAN2 architecture [KLA19] significantly improved the GAN model and demonstrated its performance by being the first to create photorealistic images while maintaining user control by taking advantage of its latent space $\mathcal{W}$. Here we extend the scope of StyleGAN2 to digital terrains. One particularity is the progressive growth of the output image performed by generators at increasing resolutions that take the previous layers as input, driven by the latent representation in $\mathcal{W}$. The process starts at an initial low resolution (generally 4×4). Every step increases the resolution of the features maps by a factor of two, and the generator employs the latent vector $\mathbf{w} \in \mathcal{W}$ to add details. Another layer, denoted as *toDEM*, is specialised to project these features maps into images combined with the previous layers . Figure 3 illustrates our particular architecture.

Unfortunately, as demonstrated in [AQW19], $\mathcal{W}$ cannot represent every real image. Therefore, $\mathcal{W}^+$ vectors were introduced to express the necessary variability across models. The StyleGAN2 generator defines $\mathcal{W}^+$ as a concatenation of 18 different 512-dimensional $\mathcal{W}$ vectors for each layer. The components in $\mathcal{W}^+$ control the generation of every layer in the hierarchical process.

Encoder The StyleGAN2 architecture has a great synthesis power but lacks a so-called *inversion* process: what is the representation of an existing terrain in the latent space? Two major categories of inversion methods exist. Optimisation-based approaches compute the loss between the generated and target terrain. Starting from an initial random vector in $\mathcal{W}^+$ and using an optimiser and back-propagation, the system performs an optimisation over $\mathcal{W}^+$. Despite its high-quality results, this method is computationally intensive, up to several minutes per image, which is prohibitive for an interactive application. We prefer the encoder-based approach for inverting a terrain, which implements another neural network, denoted as $\mathcal{E}$, encoding an image into latent-spaces. It necessitates preliminary training and thus operates on pairs of terrain model and latent representations. We refer the reader to a review of GAN inversion from [XZY*] for more details. The efficient encoder option lends itself to the interactive generation. Moreover, it allows for generalisation as it can invert various inputs, including high or low-resolution DEMs or sketches. This property is essential and allows for various use cases adapted to authoring.

We incrementally build upon the pSp (pixel2style2pixel) architecture proposed in [RAP*] that allows the StyleGAN2 to produce an output image based on an input image using the latent intermediate representation $\mathbf{w} \in \mathcal{W}^+$. In our experiments, we used L2 and LPIPS losses for the encoder, which provided the best results.

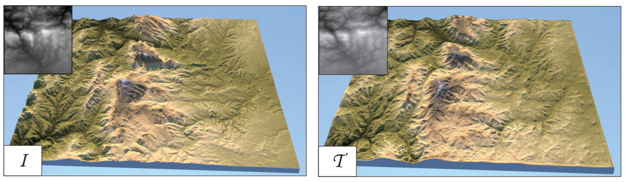

Figure 4: *The encoder inverts an input terrain (left) into latent space, before the generator synthesises an approximation (right) using the latent vector representation.*

Encoding the feature maps. While producing approximations of the input terrain, the $\mathcal{W}^+$ latent space fails at localising the features correctly. Contrary to the common usage of StyleGAN2 (faces, cars, building), landforms such as crests or rivers can be placed anywhere onto the domain. This constraint demonstrates the crucial need for localisation, which is difficult to represent as a one-dimensional latent vector. Therefore, we introduce the feature maps $\mathcal{F}$ space, a richer representation used to transmit information between blocks of various resolutions in StyleGAN2 (see Figure 3).

We developed a novel network for encoding the feature maps. The network is fully convolutional since the desired output is composed of 2D feature maps. The network has multiple branches, one for each resolution of the features maps. During the inference process, the user selects a resolution x, and the input image $\mathcal{I}$ is streamed to the network to encode it into feature maps $(\mathbf{w}, \mathbf{f}_x) = \mathcal{E}(\mathcal{I})$. Conversely, we enrich the generator with the capacity to generate a terrain by bypassing the lower part of the generator and using the feature maps directly:

$$\mathcal{T} = \mathcal{G}(\mathbf{w}, \mathbf{f}_x)$$

We train the network by picking one resolution randomly for one step, and the backward pass optimises this branch. Over time, we train all branches with the same number of steps. The inversion into features maps is crucial to retrieve accurately the features from the input. While the vector $\mathbf{w}$ has a small dimension, feature maps are considerably richer and represented by 2D tensors with many channels. Increasing the dimensions allows the spatialisation of features into the latent space and a better recovery of the features of the input (see Figure 4). Both latent representations are complementary: $\mathbf{w} \in \mathcal{W}^+$ is adapted for style modification and $\mathbf{f} \in \mathcal{F}$ for features conservation and localisation.

4. Terrain authoring with style

We developed and studied a variety of authoring tools that benefit from both latent space representations and the generalisation possibilities of the encoder. Since the information given to the encoder uses the same format as the generator output, it intrinsically allows the interactive edition of the output terrain $\mathcal{T}$ and iteration through the process.

4.1. Versatility

We trained the encoder (Section 5) using real DEMs as input, $\mathbf{w} \in \mathcal{W}^+$ latent vector and $\mathbf{f} \in \mathcal{F}$ features maps as outputs. Every vector $\mathbf{w}$ encodes a high-resolution topography in the high-dimensional latent space of the generator and thus embeds a consistent representation of geomorphological properties. The inference capabilities enable us to feed it with new inputs that have never been seen during the training phase. This generalisation allows the user to sketch low-resolution maps and edit existing DEMs while keeping consistency and generating necessary details. The encoding can be seen as a projection of the input into the latent space of the generator. To target different terrain scales and offer more flexibility to the user, we trained two different generators and encoders: at 5 meters per pixel (5km terrain) and 30 meters per pixel (30km terrain). This is complemented by the tiling approach that further extends the scale possibilities by stitching multiple terrains together (section 4.3).

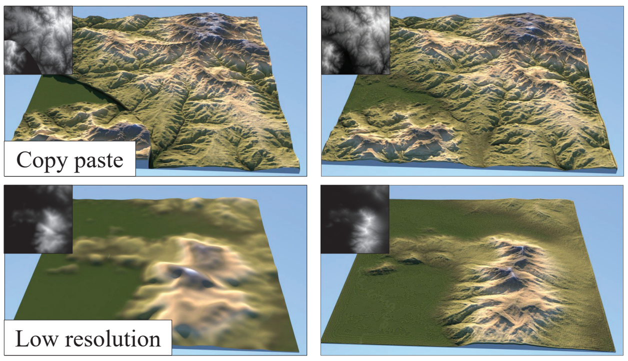

Figure 5: *Different categories of input terrains (left) and their corresponding topographies produced by the full encoder and generator process (right): synthesised models exhibit more small-scale details and landforms while maintaining a global consistency.*

Figures 1, 4, 5, and 6 demonstrate the variety of inputs handled by the model: a DEM from a real terrain, a copy-and-paste editing, a user-defined sketch, or a low-resolution DEM.

A frequently used authoring paradigm is to select elements from a first map and paste them onto another map. This process produces seams that previous works usually fix using blending zones. Unless performed in the gradient domain as in [GPM*22], this results in a smooth interpolation of elevations [GGP*15] that blurs the landforms. In contrast, our method produces patterns that conform to the essence of the pasted elements.

In addition to the semantic representation in $\mathcal{W}^+$ space, we add an $\mathcal{F}$ space output in the encoder that provides a better reconstruction by encoding spatial information, but that can sometimes fail at representing geomorphological consistency. Using the $\mathcal{F}$ space also allows reconstructing more accurately features from input. One potential drawback occurs when a seam is intended not to be reproduced: the user may want to add geomorphological details at the seam location while preserving the input terrain elsewhere. Therefore, we introduce a way to mix up latent, taking inspiration from [PLL*22], by using a mask to merge different feature maps. The entire $\mathbf{w} \in \mathcal{W}^+$ is retrieved and injected into the first layers of the generator. At a given resolution determined by the user, we merge the feature maps and the current image using linear interpolation with a user-defined mask to better reconstruct geomorphological details while maintaining crucial landforms.

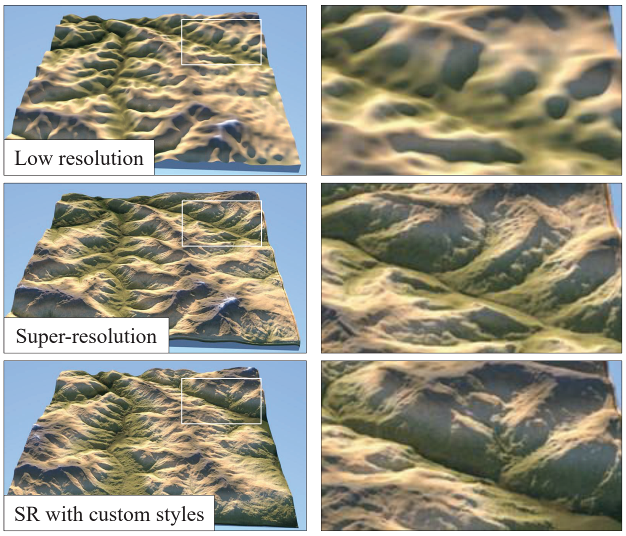

Figure 6: *Our encoder accepts low-resolution terrains as input and allows for super-resolution. Here, the initial resolution was* 64×64 *(top), illustrating a* $\times 16$ *amplification factor (middle). We further control the super-resolution amplification process by a style exemplar (bottom).*

4.2. Style mixing

In essence, the style mixing proceeds as follows (Figure 7). Two latent vectors $\mathbf{u}$ and $\mathbf{v} \in \mathcal{W}$ are computed using the encoder with an additional $\mathbf{f} \in \mathcal{F}$ to keep terrain $\mathcal{I}_1$ structure. Two generators are then run in parallel and input with $\mathbf{f}$. Vectors $\mathbf{u}$ and $\mathbf{v}$ are then fed into each generator. At every level, the features are merged before being streamed to the *toDEM* layer.

The generator inherits from the StyleGAN2 architecture and is structured in 18 layers controlled by the latent vector $\mathbf{w}$. While the same latent vector controls the different layers during the training phase, extending the latent space to 18×512 elements allows to select different ones. This is particularly true when using the encoder that produces an extended latent vector $\mathbf{w} \in \mathcal{W}^+$ to increase the expressivity (see Section 3). Furthermore, the additional latent space $\mathcal{F}$ better reproduces the structure and delivers style mixing capacities: the large-scale structure and landmarks of one terrain can be mixed with the details of another one. The structure is represented

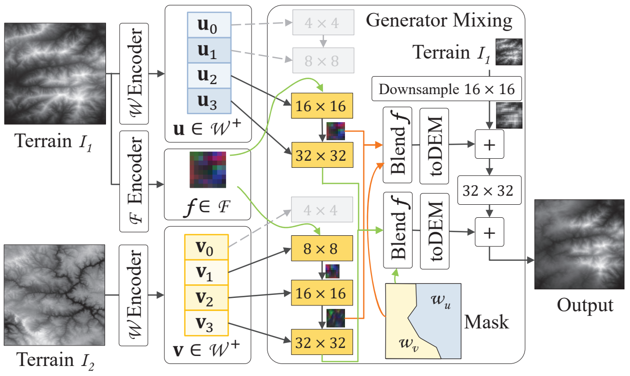

Figure 7: *Synthetic overview of the style mixing architecture.*

in the $\mathcal{F}$ space as well as the upper layers of the latent vector $\mathbf{f}$, whereas details are connected to the lower ones.

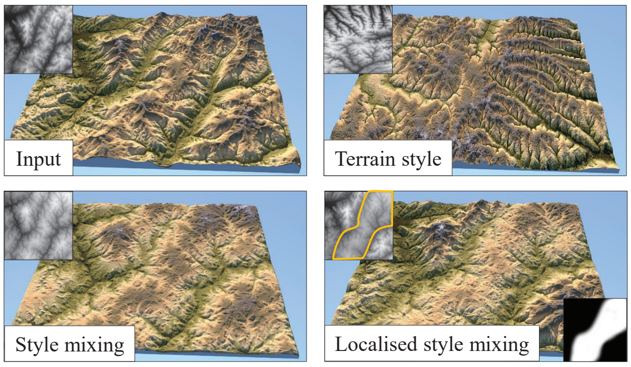

Figure 8: *To modify the style over a region* Ω*, we run two versions of the generator in parallel, one with* $\mathbf{f} \in \mathcal{F}$ *extracted from the input and the other from the style. Before invoking the toDEM layer, we blend the feature maps using a user-defined mask derived from* Ω*.*

We designed a tool that combines the global structure from an input terrain with the details extracted from another one (Figure 8). The network's first layers (low-resolution) contain large-scale features, and the scale of the features decreases with the successive layers [KLA19], a consequence of the growth of the characteristics of the generator. Therefore, the latent vector $\mathbf{w}$ at a given resolution controls the style at the corresponding scale. Here, it corresponds to decreasing spatial features from large-scale landforms, such as mountains and valleys that define the global geomorphology, to small-scale details, such as ravines erosion landmarks.

To retrieve the structure of the original terrain, we employ the latent space $\mathcal{F}$ that corresponds to the feature maps of the generator. Features maps are conditioned using the latent space $\mathcal{W}$ and then projected into pixel space using a toDEM layer. We stream the output of the feature maps encoder directly to the generator. We then run two generators in parallel, the first with the $\mathbf{u}$ and the second with $\mathbf{v}$. Before being transferred to the toDEM layer, which outputs the difference of the image for this resolution, we blend feature maps spatially using a user-defined mask, which allows for maintaining a global structural coherence since the input is the same for each generator while differentiating the style spatially. This pro-

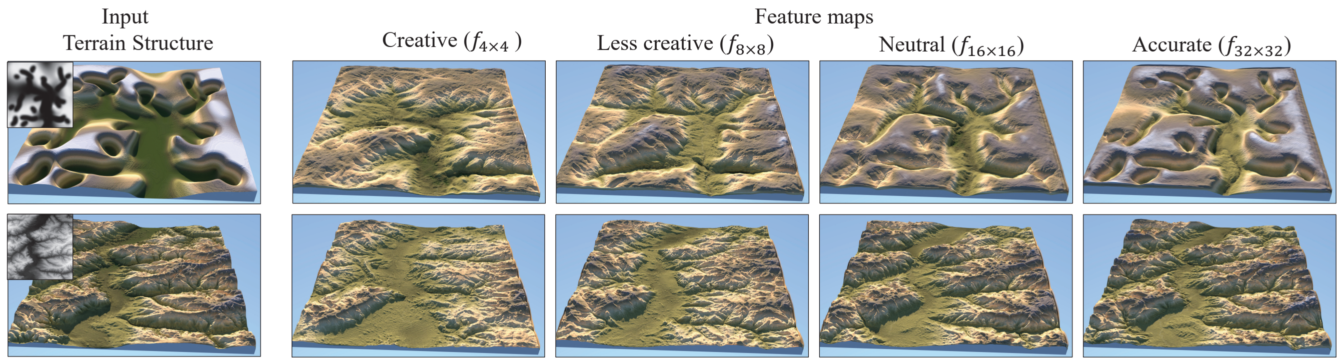
Input
Terrain Structure
Feature maps
Creative ($f_{4\times4}$)
Less creative ($f_{8\times8}$)
Neutral ($f_{16\times16}$)
Accurate ($f_{32\times32}$)

Figure 9: *Influence of the feature maps insertion at the i-th level: the user can adjust the creativity or prefer adherence to the input. If they prefers reproducing the terrain as accurately as possible, they can employ the fourth level. In contrast, if they wants to draw a draft, they may use a low-level feature map.*

cedure can be easily extended to multiple styles in one terrain by running multiple generators.

The intrisic style embedding allows quick prototyping by changing details according to an input style, potentially resulting in a completely different visual perception of the initial terrain. The user selects the number of layers needed to apply the desired effect, the level of features maps (independently for the input terrain and styles), and the mask for styles, which allows for a balance between global and local control. Figure 9 shows the impact of spatial style and the level of details. The generator is fast enough to perform those operations interactively.

4.3. Tiling

Modelling large-scale landscape at a high resolution is crucial, but computationally intensive and time consuming for designers. Augmenting the resolution automatically saves a lot of time for artists and allows them to focus on the prominent structures and patterns such as ridge lines, river networks or plateaus. We exploit the generalisation capabilities of our decoder and particularly the way our method synthesises details with a low-resolution input. In the current implementation, we can handle two different output resolutions based on the trained models: 30 and 5 meters resolution.

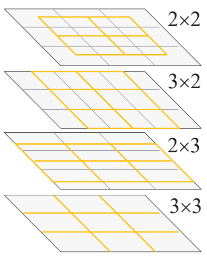

Figure 10: *Patch decomposition.*

Because the model has a fixed resolution of $1,024 \times 1,024$, generated terrains have a size of about $s \approx 30$ and $s \approx 5$ kilometres respectively. To overcome this limitation, we handle larger resolutions and sizes by dividing them into patches of a size s (see Figure 11). We decompose input terrains $\mathcal{T}$ of arbitrary resolution, *i.e.*, larger than the $1,024 \times 1,024$ resolution required by the networks, into a grid of k^2 patches. Detailed patches produced by the encoder have a normalised elevation range. Therefore, we need to adapt each patch elevation to the original patch using a histogram matching, which guarantees the recovery of the original elevation range and distribution. This strategy still produces discontinuities at the borders of the independently-generated patches. Therefore, we use a half-size $s/2$ overlapping and blending. We compute intermediate patches covering the boundaries as illustrated in Figure 10 and combine the three layers of intermediate patches with offset vectors $(s/2)\,\mathbf{x}$, $(s/2)\,\mathbf{y}$ and $(s/2)\,(\mathbf{x}+\mathbf{y})$ respectively. The process yields $(2k-1)^2$ patches. We finally stitch them together using the minimum error boundary cut from [EF01].

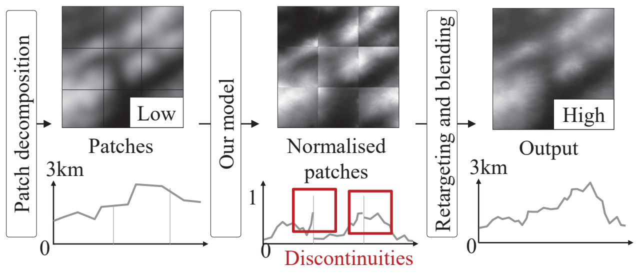

Figure 11: *The input terrain is divided into $n \times n$ patches to adapt the size of the trained model. After processing by the networks, we adapt the heights of the (normalised) high-resolution patches and finally blend patches together.*

Figure 12 illustrates the combinations of the two terrains and tiling. The same input sketch can produce terrains of sizes ranging from 5 km to 180 km. Terrains of 5 km and 30 km use model trained with the corresponding resolution while terrains of 90 km, 180 km or larger are composed using the proposed tiling method.

5. Results and discussion

The source used for creating datasets is composed of publicly available raster images of Digital Elevation Models (DEMs). We trained two networks at 5 and 30 meters resolutions and built the datasets accordingly. Training multiple models at different resolution allow a larger range of usage since features of 5×5 kilometres terrains are

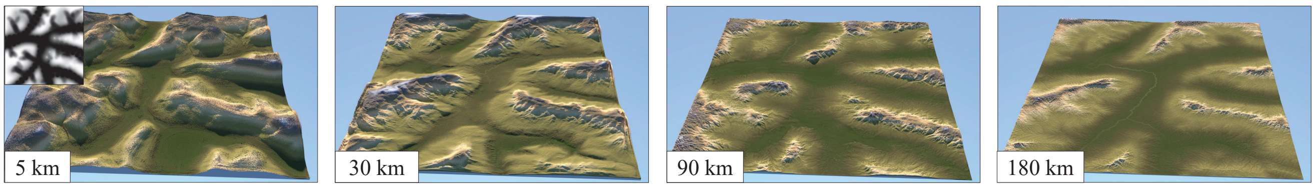

Figure 12: *Our framework can adapt to any terrain size by using one tile of the two models trained natively at* 5 km *or* 30 km*. It is also possible to use the* 30 km *model combined with the tiling process detailed in section 4.3 to compose larger terrains,* 90 km *and* 180 km*, respectively with* 3×3 *and* 6×6 *tiles, that correspond to* $3,072\times3,072$ *and* $6,144\times6,144$ *pixels.*

sensibly different than 30×30 kilometres terrains, where mountain chains are clearly visible. In particular, we used the IGN RGE ALTI database composed of 5×5 kilometres patches with a five-meter precision and a spatial resolution of $1,000\times1,000$. We resampled the tiles to $1,024\times1,024$ resolution using bi-cubic interpolation to adapt to the network requirements, as the generator produces images with a power-of-two resolution. We downloaded 5,600 DEMs covering western Europe and performed a selection based on their dynamic range so that flat terrains should not be over-represented and over-generated. We created elevation histograms and selected the same number of representatives for each class to avoid bias. More precisely, we classified every tile according to its elevation range rounded to the nearest ten meters. We then randomly selected ≈ 20 terrains to maintain the balance between classes, resulting in 1,760 patches.

The second dataset for the 30×30 m resolution, uses data from NASA SRTM consisting of 344 DEMs covering parts of Europe, cut into 3,096 patches at 30 meters precision. We applied the same process to handle the dynamic range and produced 1,902 patches.

We modified the StyleGAN2 to support a 16-bit grey-scale precision format for terrain images, which were normalised to unit intervals for training purposes. The generator $\mathcal{G}$ was trained on four Nvidia V100 GPUs with 16 GB of memory during 35 hours for the five meters precision and 40 hours for 30 meters precision, using the 5 meters as pretraining. After completing the generator-training process, we generated 20,000 synthetic terrains to train the encoder using randomly selected latent vectors, divided into 16,000 images for training and 4,000 for testing. The second training was performed on a single NVidia V100 GPU with 16 GB of memory over 12 hours. We released the source code and the trained model together with the training data.

5.1. Implementation and performance

The scripts for generating the data sets were coded in Python, and we used PyTorch for machine learning. StyleGAN2 [KLA*20] and the encoder pSp [RAP*] were adapted from the author's implementation. The generator and the pSp encoder were first trained as presented and then frozen to train the additional network. Recall that the encoder has multiple branches, one for each feature maps spatial resolution, built with layers of convolutions and skip connections. Each layer reduces the spatial resolution by two and increases depth by two until the target resolution and channels are reached. We use a Parametric Rectified Linear Unit (PReLu) as an activation function and batch normalisation. We used the same data set for the encoder. The encoding and generation process run at ≈ 48 ms on the same Nvidia GPU hardware (see Table 1), thus providing interactive feedback to the user. The implementation of the Blender add-on (see accompanying video) inherently adds a processing overhead to this raw computation time (900 ms for data preparation, 250 ms for postprocessing, and 900 ms for UI displacement update).

Tool	RTX3090	$\#\mathcal{G}$	$\#\mathcal{E}$
Generator	15 ms	1	0
Encoder	32 ms	0	1
Generator + Encoder	48 ms	1	1
Style mixing	99 ms	1	2
Tiling 3×3	6.7 s	25	25
Tiling 6×6	17.7 s	121	121

Table 1: *Performance comparison for different operations: the last two columns report the number of passes of the generator* $\mathcal{G}$ *and the encoder* $\mathcal{E}$*. Tiling includes other processes such as blending or retargeting, which count towards the overhead.*

All the terrains presented in this work were rendered using Mitsuba with cartographic shading and generated at 30 meters per pixel (unless stated otherwise) using the corresponding model. Textures are created automatically with a mix of detection of rivers, hypsometric colours, and Laplacian contrast enhancement. This rendering emphasises landforms, improving visual inspection of topography and avoiding aesthetic shading.

5.2. Control

We developed a plugin for the open-source modelling software Blender that grants accessibility to our algorithms to unfamiliar users. It integrates all the functionalities and interactive tools described in Section 4 and can be used in all environments: modelling, shaders and render engine (see the accompanying video for examples of the user interaction).

Figure 13 shows the impact of the feature maps encoding on the resulting DEM. The user can reproduce the terrain faithfully by choosing an accurate level (f_{32x32}), or let the model generate new

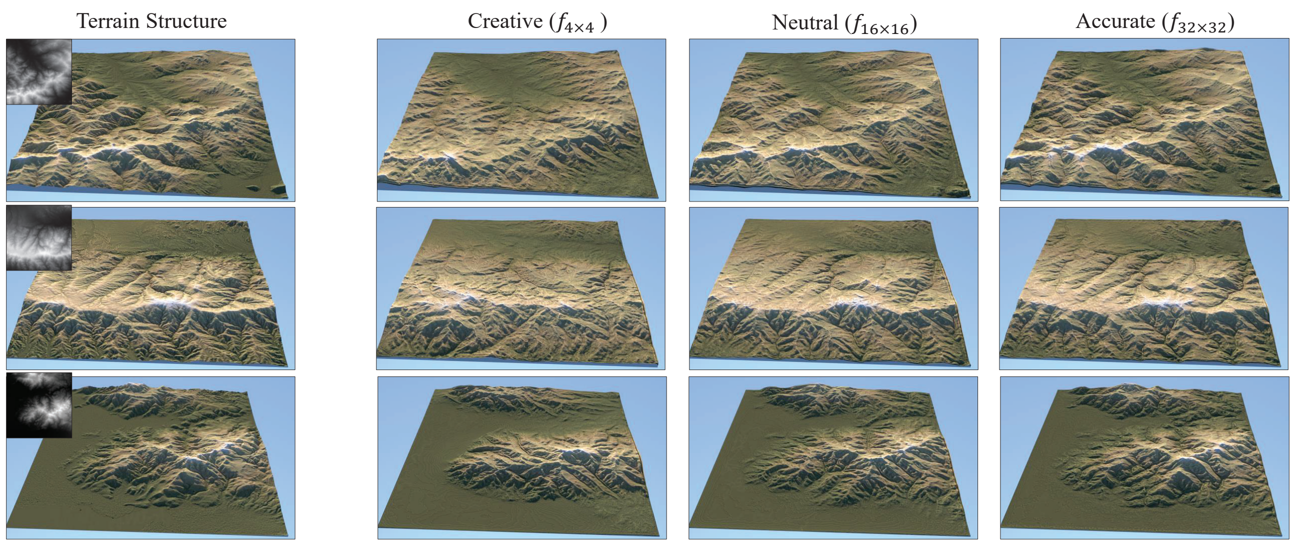
Terrain Structure
Creative ($f_{4\times4}$)
Neutral ($f_{16\times16}$)
Accurate ($f_{32\times32}$)

Figure 13: *Effect of the fidelity vs. expressiveness control parameter on three different input terrains.*

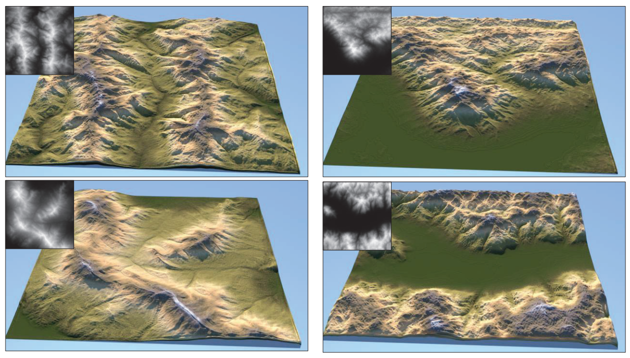

Figure 14: *Landscapes obtained by non-artist users during the qualitative testing process, using sketching tools only.*

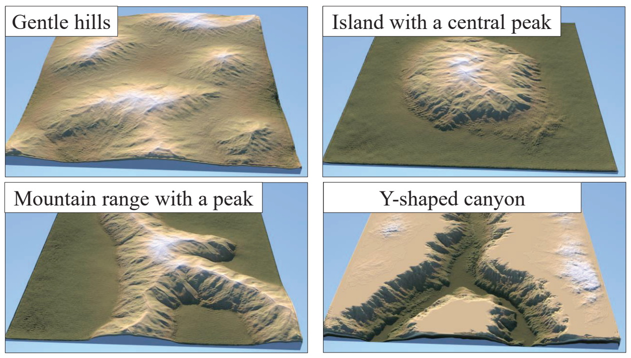

Figure 15: *Samples from a small-scale study where users are asked to reproduce a given prompt.*

details with the creative level (f_{4x4}) respecting the underlying terrain. Our method allows the combination of multiple level according to the user intent. A terrain can be faithfully reproduced using the accurate level then edited partially with the creative level.

We conducted a qualitative study with three untrained non-artist users who were asked to evaluate the modelling tools and comment on their effectiveness. Figure 14 displays results obtained after a ≈ 5 minutes editing session only, starting from scratch. All users reported that they managed to compose realistic terrains following their intent with different styles. In addition, we performed a small-scale user study where users were asked to reproduce a prompt describing a terrain. Figure 15 shows four examples of terrains obtained in this experiment.

5.3. Comparison

Our approach lends itself to terrain authoring and amplification and compares favourably to state-of-the-art methods with similar goals. The Generative Adversarial Terrain Amplification (GATA) [ZLB*19], a GAN architecture for style embedding and amplification, is considered the state-of-the-art method for terrain amplification. While producing high-quality results by amplifying a low-resolution terrain with a specific input style, it does not include interactive editing. In contrast, our pipeline offers style transfer and super-resolution and proposes multiple authoring tools assembled into a unified framework for building new reliefs or modifying existing ones. Moreover, the latent vector representation allows for transfer style at any scale.

Sparse modelling is an alternative strategy to learning for amplification and authoring explored by [GDGP16] and [AAC*17]. However, the sparse reconstruction process does not provide generalisation since patches selected among exemplars are not modified. Defining terrain style could be achievable by carefully selecting the set exemplars. Combining different styles would be possible at the expense of a database with a considerable number of examplars,

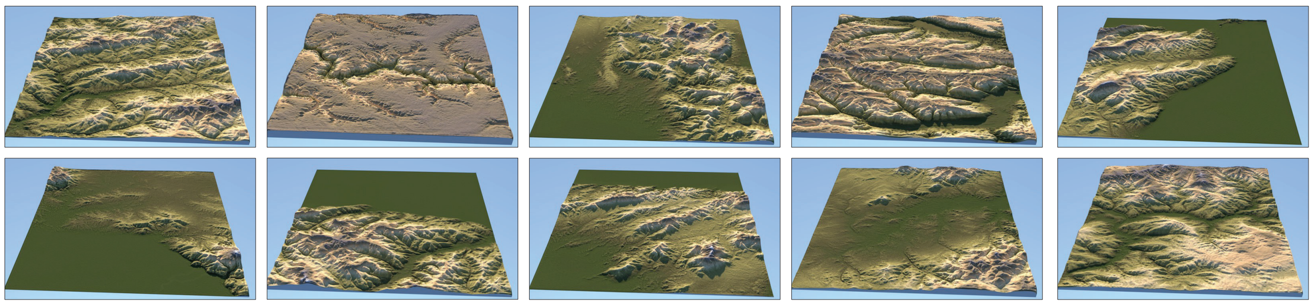

Figure 16: *Demonstration of the diversity of the StyleGAN2 generation: we can generation different types of terrain such as plains with smooth hills, high mountains with dendritic river and ridge networks, and canyon lands.*

which intrinsically limits the performance of the selection in the dictionary. Eventually, [GDGP16] may introduce repetition artefacts on planar sketch surfaces, particularly on flat input parts such as plains and plateaus, whereas our method manages to generate a palette of consistent landforms, as can be seen in Figure 17.

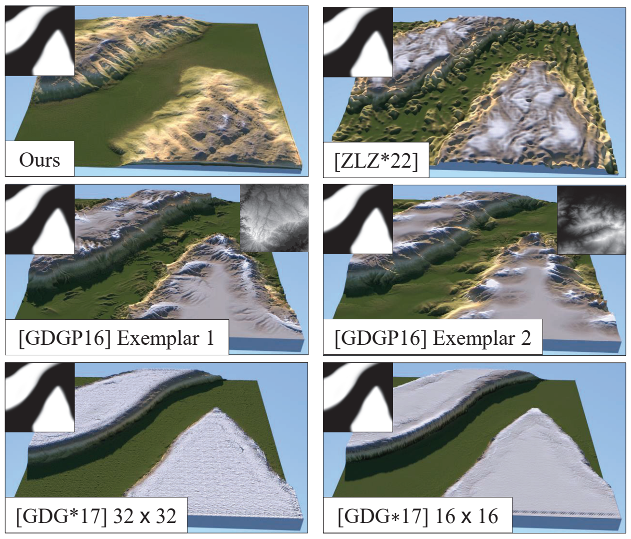

Figure 17: *Comparison of our method (top left) with style transfer, sparse modelling and cGAN.*

A recent technique [ZLZ*22] implemented a style transfer approach using GAN to encode global and local styles independently. The GAN takes a level set as input and generates a terrain by combining two levels of details producing variants of different styles. The main limitation comes from the limited number of available styles learned from specific hand-made datasets with evident generalisation limitations. In contrast, our method produces a large variability by automatically detecting the characteristics of relief, which were never provided during training.

We compared results with the vork of Guerin *et al.* [GDG*17], which was, to the best of our knowledge, the first technique proposing a deep-learning framework to terrain generation, adopting a conditional GAN architecture and providing sketches of ridges and rivers networks, and generating high-resolution heightfields. This method does not accept low-resolution elevation maps as input, therefore we had to adapt the pipeline for a fair comparison. We trained a cGAN using the code provided in [GDG*17] on our dataset based on low-resolution terrain inputs (16×16 and 32×32). We observed that while the cGAN models synthesise consistent results with the test dataset, they fail at generalising with real user sketches. In contrast, our method offers a versatile approach for authoring terrains, accepting a variety of input types and including generalisation capabilities coupled with style possibilities in a single framework. Figure 17 presents the comparison with sparse modelling [GDGP16], cGAN [GDG*17], and the recent style transfer approach [ZLZ*22]. Contrary to other methods that may show repetition artifacts, our model adds consistent details and features while keeping consistency and respecting the user sketch.

One crucial facet of our method is the versatility of tools. A single pipeline, with one training, offers extensive possibilities to the artists, as demonstrated in Table 2 that lists available features in previous works. We did not include erosion-simulation-based methods that are not relevant in this comparison.

5.4. Validation

Hydrological consistency is an important criterion for determining not only the geological but also the perceptual realism of a synthetic terrain. One form of evaluation is to compute and visualise the drainage of a terrain. Recall that the upstream drainage area a of a point $\mathbf{p}$ is the amount of water that flows through $\mathbf{p}$, and is proportional to the area of the surface where every downstream route passes through $\mathbf{p}$.

Figure 18 provides a visual comparison of the drainage for various existing terrain generation techniques whose topographies were obtained from the online repository of [GGP*19]. Shading varies from blue to red, corresponding to small and high drainage area values respectively. Unwanted endorheic pits and bumps blocking the water flow result in an inconsistent drainage. They appear in the form of many white and red streamlines blocked without reaching the border of the domain.

Figure 18 also reports the average breaching volume $\bar{b}$ for each case. It is defined as the volume of material removed by the

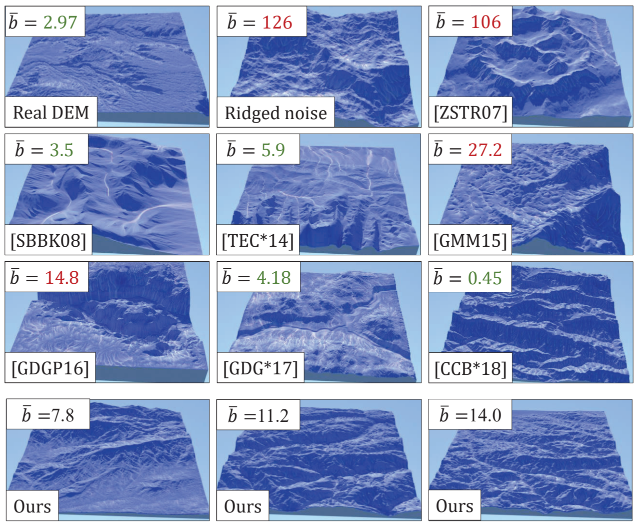

Figure 18: *Drainage area and breaching volume $\bar{b}$ for various terrain generation techniques and Real DEMs; a lower breaching volume, $\bar{b}$, is better.*

minimal breaching required to ensure free drainage of the terrain [BLM14] divided by terrain area. Our method performs better than standard ridged noise and other example-based methods [ZSTR07, GDGP16, GMM15] except for [GDG*17] that integrates post-processing on the generated DEM (a river has been carved manually). The authoring method by Tasse *et al.* [TEC*14] relies on manual editing of Real DEMs which explains the consistent drainage and low average breaching volume $\bar{b}$. Simulations [ŠBBK08, CGG*17] perform better than learning-based approaches as they implicitly embed drainage consistency.

Since our method relies on learned DEMs, we also compared the average breaching volume $\bar{b}$ for a collection of 50 real terrains against 50 generated terrains. The mean score for real DEMs is 2.77 whereas our method gives a 13.3 mean: not as good as real terrains as one could expect, but often better than several other generation techniques.

The training of the StyleGAN2 network is the first step in the training of our model and its generation capacity is a key component. Figure 16 shows a large variety of terrains structures that can be created with the generator. We choose an encoder approach to retrieve the latent vector $\mathbf{w}$ and $\mathbf{f}$ inside the StyleGAN2 latent space. An alternative method consists in iteratively optimising random latent vectors to match the input data using a standard loss based on the difference between input and the produced terrain.

Theoretically, this method converts a real terrain into a vector in the latent space $\mathcal{W}$ more precisely and therefore lends itself for style mixing with high-resolution models. In the case of low-resolution inputs or sketches, the optimisation faithfully reproduces the input without introducing any learned landforms. By default, it uses a perceptual loss based on the mean squared error on VGG16 features. Adding an L2 loss directly on images yields better results since the loss is evaluating individual pixels. This loss places landforms such as mountains more faithfully and removes some noise. To compare both optimiser and encoder, we added the optimisation of a $\mathbf{f} \in \mathcal{F}$ latent space in addition to the $\mathbf{w}$ vector.

Figure 19 shows a comparison between encoder and optimiser-based methods. The optimiser delivers good results using real terrains and tends to be more accurate. The encoder is a competitor adapted to using sketches and performs better in this configuration by generating features according to the prescribed inputs.

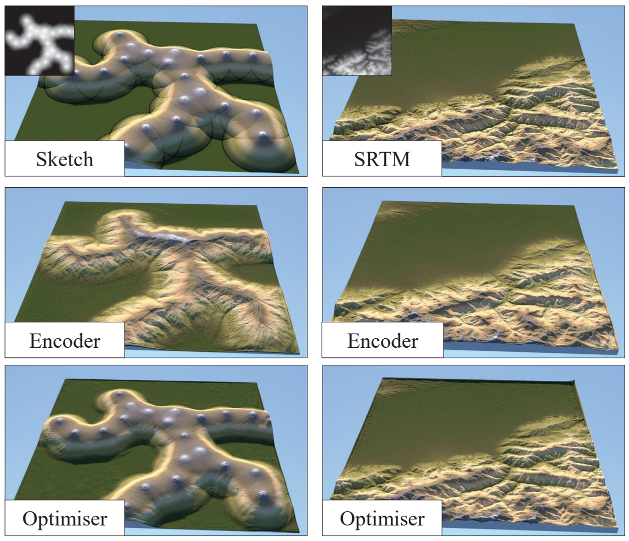

Figure 19: *Comparison between encoder and optimiser-based methods for a sketch input (left) with $\mathbf{f} \in \mathcal{F}_8$ and a SRTM DEM (right) with $\mathbf{f} \in \mathcal{F}_{16}$.*

In contrast, the encoder trained with heightfields allows for a broader range of applications, as exemplified in Section 4.1. The optimiser takes ≈ 60 s to converge to a solution, which is to be compared to ≈ 48 ms using the encoder. Only the latter is compatible with interactive feedback.

We also performed an ablation study to evaluate the importance of the feature maps inversion. Figure 20 shows that even if $\mathcal{W}^+$ latent space generates details, large structures might be altered while the features space $\mathcal{F}$ preserves spatialisation and allows adding details by using masks.

5.5. Limitations

Our approach has several limitations. The first one comes from the specific size of landforms used to train the generator: it needs training a specific encoder and generator for every resolution, which requires intensive learning and an increasing amount of exemplars. This limitation might be compensated by utilising the tiling method.

The second limitation comes from the encoder: the user selects between a creative tool and an accurate encoding process, depending on the target application. In practice, the lowest level ($\mathbf{f}_4$) is creative and innovative, therefore adapted to low-resolution sketches. In that case, the encoder generates various landforms. In contrast, if

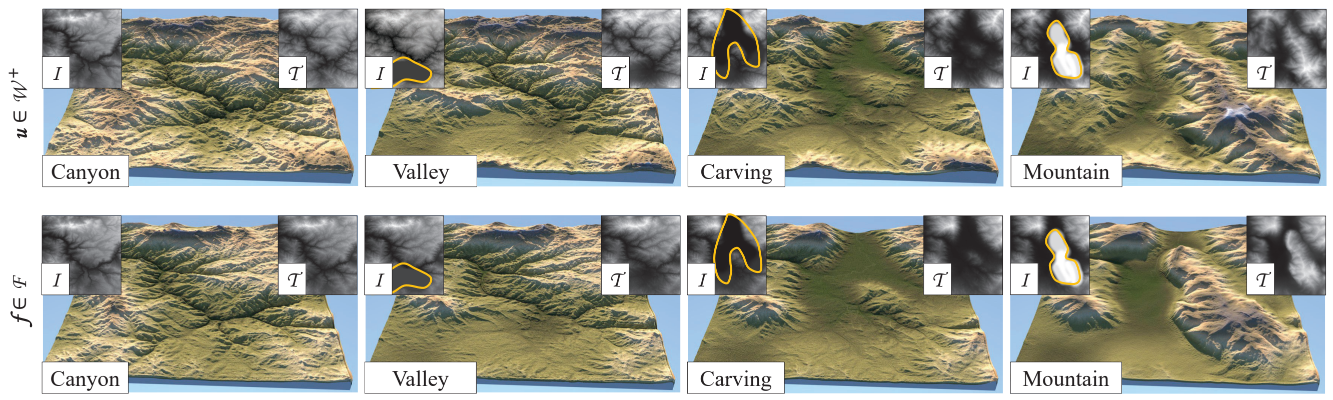
$\boldsymbol{u} \in \mathcal{W}^+$
$\boldsymbol{f} \in \mathcal{F}$
I
$\mathcal{T}$
Canyon
Valley
Carving
Mountain

Figure 20: *An ablation study on the importance of the features maps: when using feature maps (bottom), large structures of the terrain are faithfully reconstructed while keeping details where the user edited using a mask and a combination of multiple levels of features maps.*

the input is considered accurate, we observed that choosing a level of $\mathbf{f}_{32}$ reproduces the input with high fidelity. Intermediate levels proved very effective for modelling phases, particularly when combined with style-mixing features.

6. Conclusion

We introduced a versatile deep neural model for authoring terrain that allows designers to perform numerous editing tasks in the latent space while conforming to the overall style of the terrain. From sketch-based authoring to style transfer, by way of interpolation and super-resolution, the model comprises a description of terrains that inherently encompasses its geomorphological characteristics and guarantees consistency during generation. We present a novel encoding performed in the feature maps domain that yields better accuracy and spatialisation in the inversion process. Combined together, these two latent representations bring a complete toolset for terrain authoring. Experiments and a small-scale user study demonstrate its effectiveness as well as its advantages toward previous methods, including machine learning ones.

Future work in AI-supported terrain modelling should focus on combining local and global changes into a single framework. The user should be able to define the scope of the changes, and the model would operate in the given domain while providing a seamless connection with the rest. Also, AI models often require large sets of terrains to be trained. Having a single-shot model that would not require demanding training and would allow for generalisation and the creation of novel terrains with the same style would be valuable. Finally, a textual form similar to stable diffusion models could replace the traditional sketch-based control.

Acknowledgments

This work is funded by the project AMPLI ANR-20-CE23-0001, supported by Agence Nationale de la Recherche. This work was granted access to the HPC/AI resources of IDRIS under the allocation 2023-AD011013212R1 made by GENCI.

	Article	Amplification	Sketches	Interpolation	Style-mixing	Spatialised style	Curve Constraints	Copy-paste	Code / Dataset
Example-based	Ours	•	•	•	•	•		•	• / •
	[ZLZ*22]		•	•	•	•			• / •
	[ZLB*19]	•		•	•				• /
	[GDG*17]	•	•					•	• / •
	[GDGP16]	•	•		•		•		• /
	[GMM15]		•		•	•	•	•	/ -
Procedural	[GPM*22]	•	•				•	•	• / -
	[GBG*19]		•						• / -
	[GGP*15]						•	•	/
	[AGP*19]		•		•		•	•	• / •
	[GMS09]						•		/ -
	[TEC*14]						•		• / -
	[dCB09]		•						/ -
	[GGG*13]		•				•		/ -
	[ZSTR07]		•						/ -

Table 2: *Comparison of tools available in different terrain synthesis models. Dataset release and trained models are only compatible for machine learning methods, '-' stands for not applicable.*

References

[AAC*17] ARGUDO O., ANDUJAR C., CHICA A., GUÉRIN E., DIGNE J., PEYTAVIE A., GALIN E.: Coherent Multi-Layer Landscape Synthesis. *The Visual Computer 33*, 6 (2017), 1005–1015. 8

[ABGT20] ABADY L., BARNI M., GARZELLI A., TONDI B.: GAN generation of Synthetic Multispectral Satellite Images. In *Image and Signal Processing for Remote Sensing XXVI* (2020), Bruzzone L., Bovolo F., Santi E., (Eds.), vol. 11533, International Society for Optics and Photonics, SPIE, p. 115330L. 2

[AGP*19] ARGUDO O., GALIN E., PEYTAVIE A., PARIS A., GAIN J., GUÉRIN E.: Orometry-Based Terrain Analysis and Synthesis. *ACM Transactions on Graphics 38*, 6 (2019), 199:1–12. 1, 11

[AQW19] ABDAL R., QIN Y., WONKA P.: Image2stylegan: How to Embed Images into the Stylegan Latent Space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 4432–4441. 3

[BLM14] BARNES R., LEHMAN C., MULLA D.: Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers & Geosciences 62* (2014), 117–127. 10

[CGG*17] CORDONNIER G., GALIN E., GAIN J., BENES B., GUÉRIN E., PEYTAVIE A., CANI M.-P.: Authoring Landscapes by Combining Ecosystem and Terrain Erosion Simulation. *ACM Transactions on Graphics 36*, 4 (2017), 134:1–12. 2, 10

[CJP*23] CORDONNIER G., JOUVET G., PEYTAVIE A., BRAUN J., CANI M.-P., BENES B., GALIN E., GUÉRIN E., GAIN J.: Forming Terrains by Glacial Erosion. *ACM Transactions on Graphics 42*, 4 (2023). 2

[dCB09] DE CARPENTIER G. J. P., BIDARRA R.: Interactive GPU-based Procedural Heightfield Brushes. In *Proceedings of the International Conference on Foundations of Digital Games* (Orlando, USA, 2009), ACM, pp. 55–62. 2, 11

[EF01] EFROS A. A., FREEMAN W. T.: Image Quilting for Texture Synthesis and Transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, Association for Computing Machinery, pp. 341–346. 6

[GBG*19] GAILLARD M., BENES B., GUÉRIN E., GALIN E., ROHMER D., CANI M.-P.: Dendry: a Procedural Model for Dendritic Patterns. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Montreal Quebec Canada, May 2019), ACM, pp. 1–9. 11

[GDG*17] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A., WOLF C., BENES B., MARTINEZ B.: Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks. *ACM Transactions on Graphics (proceedings of Siggraph Asia 2017) 36*, 6 (2017). 2, 9, 10, 11

[GDGP16] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A.: Sparse Representation of Terrains for Procedural Modeling. *Computer Graphics Forum (proceedings of Eurographics 2016) 35*, 2 (2016), 177–187. 2, 8, 9, 10, 11

[GGG*13] GÉNEVAUX J.-D., GALIN É., GUÉRIN É., PEYTAVIE A., BENES B.: Terrain Generation Using Procedural Models Based on Hydrology. *ACM Transaction on Graphics 32*, 4 (2013), 143:1–143:13. 2, 11

[GGP*15] GÉNEVAUX J.-D., GALIN É., PEYTAVIE A., GUÉRIN É., BRIQUET C., GROSBELLET F., BENES B.: Terrain Modeling from Feature Primitives. *Computer Graphics Forum 34*, 6 (2015), 198–210. 2, 4, 11

[GGP*19] GALIN E., GUÉRIN E., PEYTAVIE A., CORDONNIER G., CANI M.-P., BENES B., GAIN J.: A Review of Digital Terrain Modeling. *Computer Graphics Forum (Proceedings of Eurographics 2019) 38*, 2 (2019), 553–577. 2, 9

[GMM15] GAIN J., MERRY B., MARAIS P.: Parallel, Realistic and Controllable Terrain Synthesis. *Computer Graphics Forum 34*, 2 (2015), 105–116. 2, 10, 11

[GMS09] GAIN J. E., MARAIS P., STRASSER W.: Terrain Sketching. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (Boston, USA, 2009), ACM, pp. 31–38. 2, 11

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative Adversarial Nets. In *Advances in Neural Information Processing Systems* (2014), Ghahramani Z., Welling M., Cortes C., Lawrence N., Weinberger K. Q., (Eds.), vol. 27, Curran Associates, Inc. 3

[GPM*22] GUERIN E., PEYTAVIE A., MASNOU S., DIGNE J., ANDJAMES GAIN B. S., GALIN E.: Gradient Terrain Authoring. *Computer Graphics Forum (proceedings of Eurographics 2022) 44*, 2 (2022), 85–95. 2, 4, 11

[HGA*10] HNAIDI H., GUÉRIN É., AKKOUCHE S., PEYTAVIE A., GALIN É.: Feature Based Terrain Generation Using Diffusion Equation. *Computer Graphics Forum 29*, 7 (2010), 2179–2186. 2

[KHWM17] KLEIN J., HARTMANN S., WEINMANN M., MICHELS D. L.: Multi-scale Terrain Texturing using Generative Adversarial Networks. In *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)* (2017), pp. 1–6. 2

[KLA19] KARRAS T., LAINE S., AILA T.: A Style-Based Generator Architecture for Generative Adversarial Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 4401–4410. 2, 3, 5

[KLA*20] KARRAS T., LAINE S., AITTALA M., HELLSTEN J., LEHTINEN J., AILA T.: Analyzing and Improving the Image Quality of StyleGAN. In *Proc. CVPR* (2020). 7

[KW13] KINGMA D. P., WELLING M.: Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2013). 2

[MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S.: The Synthesis and Rendering of Eroded Fractal Terrains. *Computer Graphics 23*, 3 (1989), 41–50. 2

[NJSR] NAIK S., JAIN A., SHARMA A., RAJAN K. S.: Deep Generative Framework for Interactive 3D Terrain Authoring and Manipulation. `arXiv:2201.02369`. 2

[PC20] PANAGIOTOU E., CHAROU E.: Procedural 3D Terrain Generation using Generative Adversarial Networks. *arXiv preprint arXiv:2010.06411* (2020). 2

[PLL*22] PARMAR G., LI Y., LU J., ZHANG R., ZHU J.-Y., SINGH K.: Spatially-Adaptive Multilayer Selection for GAN Inversion and Editing. pp. 11389–11399. 5

[RAP*] RICHARDSON E., ALALUF Y., PATASHNIK O., NITZAN Y., AZAR Y., SHAPIRO S., COHEN-OR D.: Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2287–2296. ISSN: 2575-7075. 4, 7

[ŠBBK08] ŠŤAVA O., BENES B., BRISBIN M., KŘIVÁNEK J.: Interactive Terrain Modeling Using Hydraulic Erosion. In *Proceedings of the Symposium on Computer Animation* (2008), pp. 201–210. 10

[SD21] SCOTT J. J., DODGSON N. A.: Example-based terrain synthesis with pit removal. *Computers and Graphics 99* (2021), 43–53. 2

[SPF*23] SCHOTT H., PARIS A., FOURNIER L., GUÉRIN E., GALIN E.: Large-Scale Terrain Authoring through Interactive Erosion Simulation. *ACM Transactions on Graphics 42*, 5 (2023), 162:1–15. 2

[TEC*14] TASSE F. P., EMILIEN A., CANI M.-P., HAHMANN S., BERNHARDT A.: First Person Sketch-based Terrain Editing. In *Proceedings of Graphics Interface* (Montreal, Canada, 2014), Canadian Information Processing Society, pp. 217–224. 2, 10, 11

[XZY*] XIA W., ZHANG Y., YANG Y., XUE J.-H., ZHOU B., YANG M.-H.: GAN Inversion: A Survey. `arXiv:2101.05278`. 3, 4

[ZLB*19] ZHAO Y., LIU H., BOROVIKOV I., BEIRAMI A., SANJABI M., ZAMAN K.: Multi-Theme Generative Adversarial Terrain Amplification. *ACM Transaction on Graphics 38*, 6 (2019). 2, 8, 11

[ZLZ*22] ZHANG J., LI C., ZHOU P., WANG C., HE G., QIN H.: Authoring multi-style terrain with global-to-local control. *Graphical Models 119* (2022), 101122. 2, 9, 11

[ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M.: Terrain Synthesis from Digital Elevation Models. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 834–848. 10, 11