Vector-based Terrain Modelling

Simon Perche, Eric Guérin, Eric Galin, Adrien Peytavie

Vector-based Terrain Modelling

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

[1]INSA Lyon, CNRS, LIRIS, UMR5205, F-69621 Villeurbanne, France
simon.perche@liris.cnrs.fr, eric.guerin@liris.cnrs.fr
[2]Université Claude Bernard Lyon 1, CNRS, LIRIS, UMR5205, F-69622 Villeurbanne, France
eric.galin@liris.cnrs.fr, adrien.peytavie@liris.cnrs.fr

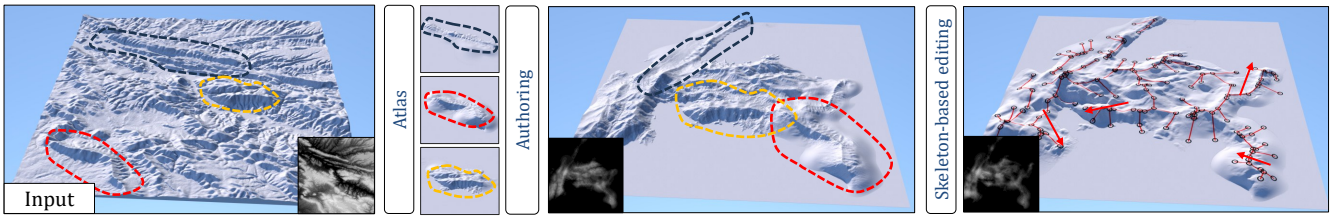

Figure 1: *Our framework transforms an input grid-based elevation model into a vector-based representation of ellipsoid-based primitives. Users can interactively edit the vector model using geomorphological skeleton-based tools, as well as standard tools such as stamping, copying, and pasting. The advantage of the vector-based representation is its ability to depict landforms at various levels of resolution while allowing for real-time editing of large-scale terrains.*

Abstract

Vector-based graphics offer numerous advantages over grid-based models, including resolution independence and ease of manipulation. Despite these benefits, their use in landscape modeling remains uncommon because of a lack of direct editing and interactive feedback, essential for matching the artist's vision. We introduce a new vector-based model for creating digital terrains based on computationally efficient primitives. We propose a method to convert grid-based digital elevation maps to this representation with a user-defined level of accuracy. Once vectorized, the terrain can be authored using interactive high-level skeleton-based tools adapted to the primitive representation, allowing local deformations that automatically adapt to underlying geomorphological structures and landforms of the terrain.

CCS Concepts

• ***Computing methodologies*** → ***Shape modelling;***

1. Introduction

Despite the vast amount of existing methods, authoring high-resolution large-scale terrains remains a challenging task for artists and designers. The difficulty finds its foundations not only in the diversity of landforms and the variety of details at different scales but also in the ability to control the shape and location of landforms to follow the designer's intent.

Designers traditionally start with an initial low-resolution model represented as an elevation grid organizing the principal landforms such as mountain ranges, valleys, canyons, and coastlines, by using procedural generation or combining patches extracted from digital elevation models. This initial draft is then refined to a detailed high-resolution grid by subdivision and interpolation and augmented with multi-resolution details such as creases and ravines produced by surface erosion [SGG*24].

Our work comes from the observation that modelling systems often relies on uniform elevation grids. While versatile and adapted to interactive editing and erosion simulations, this representation comes with several limitations. From a designer perspective, geomorphology-based edits or standard operations like warping or copy-and-paste of landforms are difficult to implement on uniform grids and require complex algorithms to seamlessly blend selected landforms into the terrain, such as operating in the gradient space [GPM*22]. Another drawback of grid-based models is the high complexity of terrain authoring algorithms, often $O(kn^2)$ where n denotes the size of the terrain and k the number of iterations, which limits their effective use to a limited resolution. Per-

formance significantly drops at 4096×4096 and becomes unusable beyond this limit. Even a parallel implementation on graphics hardware partially alleviates the problem, as memory becomes another limit. Finally, decomposing a large terrain into smaller patches and blending them only partially addresses the issue. The blending process can be computationally expensive or create visual artefacts, which may restrict the artist's creative flexibility.

Even though some methods exist for modelling terrains with procedurally-defined primitives [GGP*15], interactive authoring and generation from grid-based models have received little attention. Moreover, this vector-based model depends on a powerful but highly complex construction tree, which is challenging to work with and discourages designers from adopting it.

We address these limitations by proposing a novel vector-based approach for modeling complex terrains, bridging the gap between procedural primitives and interactive authoring (Figure 1). Our approach is, in essence, a primitive-based model that defines the elevation as a sum of compactly supported elevation functions. Instead of a hierarchical construction tree with warping, blending, and Boolean operators, we define the elevation as a sum of modified continuous Gaussian primitives complemented with procedural or data-driven detail primitives. This simple definition provides us with several advantages from an interactive authoring perspective. This continuity allows for the creation of a geomorphology-based skeleton, providing high-level structural control. Second, by interacting with this skeleton and using other specialized tools, designers can concentrate on semantic forms without the need to manage the low-level parameters of the underlying primitives. Lastly, our approach is distinctive because it incorporates vectorization, which converts traditional grid-based terrains into a primitive-based format, ensuring compatibility with existing terrain data.

The construction of the vector-based model takes its inspiration from implicit surfaces built from primitives with a compact support [GGP*15], and from the Gaussian model [KKLD23] that approximates the three-dimensional geometry of objects by a sum of Gaussian primitives. Our contributions are as follows: 1) We propose a vector-based terrain representation, defined as a sum of primitives, and an efficient algorithm for converting standard grid-based elevation models. 2) We introduce a skeleton-based rigging of primitives using a ridge graph, providing an editable high-level structure. 3) We develop several algorithms for interactively authoring vector-based terrains. Images illustrating the method and the accompanying video demonstrate that our model allows for high-level, intuitive, and interactive editing of digital terrains. We will first review previous work and its implications, then describe our model and associated tools. Finally, we will discuss and compare previous methods. The project page, providing the code and presentation video, is available at this address.

2. Related work

Here we focus on previous work on interactive terrain authoring frameworks and vector-based models. Table 4 (see Appendix B) presents a comparison of several systems. A complete review of terrain models and generation methods, beyond the scope of this related work, can be found in [GGP*19].

Simulations aim at reproducing the physical processes behind the landforms. Surface erosion simulations add details such as ravines produced by hydraulic and thermal erosion to an otherwise low-resolution terrain. They are computationally intensive, difficult to control, and therefore do not lend themselves to interactive editing. A notable exception is the work of Cordonnier *et al.* [CCB*18] who first proposed to compute the formation of mountain ranges formed by the compression and folding of colliding tectonic plates using an interactive modelling technique where the user intuitively defines the movement of plates. Inspired by the same concept, Schott *et al.* [SPF*23] set aside modelling in the elevation domain and interactively simulated mountain formation by controlling the uplift shaping mountain ranges.

Example-based techniques rely on existing terrain data to generate new terrains that resemble exemplars. Texture synthesis methods were first used to assemble terrain patches according to a user-prescribed sketch [ZSTR07]. This work was further improved by extending the controls to a broader set of vector-based manipulation tools [GMM15]. A sparse modelling approach was used to select atoms from a dictionary that best matches low-resolution elevation data in [GDGP16, AAC*17].

Curve-based approaches rely on sparse controls often expressed in terms of primitives such as points identifying peaks or curves delineating plateau contours, ridges, or rivers. Terrain elevation is generated from those control curves either by solving gradient and elevation constraints located at specific points and curves [GPM*22] or by using learning-based techniques to create terrains sketched either from contours, ridges, and river network lines [GDG*17, LGP*23], or low-resolution terrains [PPB*23]. Unlike geomorphological editing, these indirect editing techniques can lead to unintended structures or artifacts when constraints are undefined or poorly specified.

Primitive-based models represent terrain using procedurally-defined vector-based primitives [GGP*15]. While the authors propose a complex construction tree, working with it to produce the desired output can be challenging. Current vector-based representations exhibit at least one of the following limitations: they are often computationally demanding, lacking a straightforward method for computing elevation and requiring global simulation optimization [SPF*23, GMM15, HGA*10, GPM*22]. This significantly hinders interactive feedback and control, especially for large resolutions beyond the traditional 1024×1024 scale. Another key limitation is their lack of intuitive controls [GGP*15], which prevents designers from interactively editing the terrain without detailed knowledge and understanding of the underlying model.

One crucial advantage of vector-based models over discrete elevation models is their independence of scale and continuity. Genevaux *et al.* [GGP*15] introduced a general framework for modelling terrains that combines primitives in a construction tree using operators. While successfully employed in sparse modelling, the potential expression of function-based descriptions of terrains remains combined with interactive authoring and editing remains unexplored. Our model aims at bridging the gap between primitive based modelling and interactive editing by proposing various standard and geomorphology-based tools to edit vector terrains.

3. Overview and notations

The authoring framework, as shown in Figure 2, is based on both grid-based and vector-based models that complement each other. If we start with a digital elevation model, we can easily convert it into a vector-based model through an optimization process. Once in the vector domain, the designer can make iterative modifications to the terrain by the mean of high-level tools that automatically adjust primitives' parameters. To visualize the terrain, the vector-based model is converted into a grid-based elevation representation in real time using graphics hardware.

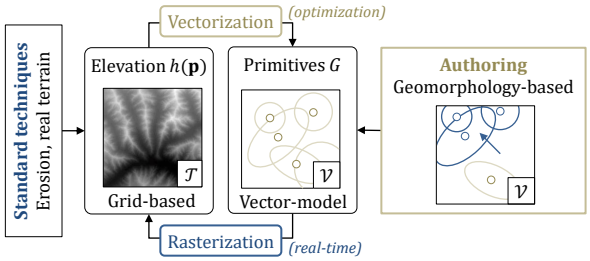

Figure 2: *The authoring process involves extracting Gaussian primitives from source elevations, applying local or global tools to modify the resulting primitive-based model, and reconstructing the resulting elevation.*

3.1. Workflow

Figure 2 shows the architecture of our framework. First, the vectorization transforms a given grid-based terrain $\mathcal{T}$ into a vector-based model $\mathcal{V}$. Once vectorized, the user can manipulate $\mathcal{V}$ using a set of tools operating on skeletons obtained from the analysis of the main landforms of the terrain. Tools also include vector-based copy-and-paste, scaling, warping and stamping, designed to maintain terrain coherency.

Reverting from the vector-based representation $\mathcal{V}$ to a grid-based one, denoted as $\widetilde{\mathcal{T}}$, can be done at any time and allows the user to perform all the usual edits associated with conventional digital elevation models, such as erosion. The vectorization is enabled by a gradient descend optimization, which relies on computing a distance between the input terrain $\mathcal{T}$ and the one derived from the vector representation $\widetilde{\mathcal{T}}$ with respect to the parameters of $\mathcal{V}$.

Many modelling tools can be favourably transferred to the vector domain, for instance copy-and-paste operations, scaling, and warping. Whenever the grid-based representation is necessary, such as for hydraulic erosion, we rasterize primitives to an elevation grid using evaluation, and perform operations before returning to the vector-based domain using vectorization and detail kernels (as depicted in Figure 2).

3.2. Primitive representation

We represent the terrain elevation $h(\mathbf{p})$ as a sum of n primitives denoted as $\mathcal{P} = \{\mathcal{E}, \mathcal{D}\}$. We identify two types of primitives: modified Gaussian primitives on ellipse domains, denoted as $\mathcal{E}$ guided by the function $g(\mathbf{p})$, that lend themselves for large-scale landforms with steep slopes, and detail primitives defined by the function $\delta(\mathbf{p})$, denoted as $\mathcal{D}$ for procedurally-defined or data-driven details. This vector-based representation is independent of the resolution. We parametrize primitives by their orientation, scale, and signed amplitude that may define both positive and negative elevation variations to allow for different types of elevation prescription. They form the core building block for modelling landscapes across a range of scales, from large mountain ranges to small-scale landforms such as ravines. The elevation is defined as the sum:

$$h(\mathbf{p}) = \sum_{i=0}^{n-1} h_i(\mathbf{p})$$

Terrains often exhibit specific landforms with asymmetric steep slopes that are difficult to define using traditional Gaussian functions. Thus, we rely on generalized Gaussian kernels introduced by Hamdi *et al.* [HMKQ*24] to handle such features that arise at high spatial frequencies, and reduce the number of needed primitives. Formally:

$$g(\mathbf{p}) = ae^{-\left(\frac{1}{2}d(\mathbf{p})^2\right)^{\beta}} \qquad d(\mathbf{p}) = \|\mathbf{A}^{-1}(\mathbf{p}-\mathbf{c})\|$$

d denotes the distance to the centre $\mathbf{c}$, with $\mathbf{A} = \mathbf{R}(\theta)\mathbf{S}(\sigma)$, $\mathbf{R}$ and $\mathbf{S}$ the rotation and scaling matrices respectively, where $\sigma = (\sigma_x, \sigma_y)$ is the scaling vector and θ the angle in radians. The amplitude a may be negative to carve the terrain. Figure 3 shows the effect of β on the shape of the function. The steepness parameter β provides optimization with larger degree of freedom to better fit grid-based input terrains. This representation allows for any values for the parameters without range constraints, which is important for the computation of the gradient ∇h and optimization (see Section 4) as all parameters can be optimized independently.

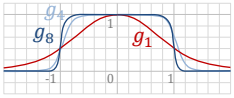

Figure 3: *Modified kernels g_β with the variation of β parameter*

Figure 4 reports the parameters of the primitive, the support of the ellipse domain is computed using $(3\sigma_x, 3\sigma_y)$ as half-width and height. Following statistics theory, 3σ around a 2D Gaussian distribution includes 98.9% of values.

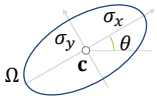

Figure 4: *Primitive parameters.*

High-frequency details are difficult to reproduce, even with generalized Gaussian primitives. Therefore, we introduce detail primitives $\mathcal{D}$ to add details to the vector approximation. They are typically defined by their location, scale, rotation parameters, but have procedurally defined amplitude $a(\mathbf{p})$ whose influence is weighted by a Gaussian fall-off:

$$\delta(\mathbf{p}) = a(\mathbf{p})e^{-\frac{1}{2}d(\mathbf{p})^2} \qquad d(\mathbf{p}) = \|\mathbf{A}^{-1}(\mathbf{p}-\mathbf{c})\|$$

In contrast to the set of Gaussian primitives $\mathcal{E}$, detail primitives $\mathcal{D}$ are not positioned by the optimization procedure since their main goal is to achieve a uniform distribution across the terrain. To maintain consistent spatial coverage and avoid undesirable amplification caused by the cumulative effect of all primitives in the final output, their placement is determined through Poisson sampling. This sampling method is configured with a radius that allows for slight

overlap between disks, effectively filling any potential gaps without resulting in excessive accumulation.

A procedural noise function could be employed to introduce these details, with amplitude and frequency parameters carefully modulated according to the terrain's topography. This approach allows the detail intensity and scale to adapt dynamically to the terrain's elevation and slope, creating a more natural and cohesive integration of features across varying landscapes.

Another approach consists in using the error $\Delta\mathcal{T}$ to directly guide the details. It is simply defined as the difference $\Delta\mathcal{T} = \mathcal{T} - \widetilde{\mathcal{T}}$ between the reference terrain $\mathcal{T}$ used in the optimization, and the rasterized approximation $\widetilde{\mathcal{T}}$ from the vector representation $\mathcal{V}$. In this case, detail primitives encode a position within the error map $\Delta\mathcal{T}$, with default scale and rotation. During rasterization, detail primitives query $\Delta\mathcal{T}$ at its corresponding position to retrieve the error, which is then added to the results. While this introduces raster information in the model, it allows for accurate reconstruction. Furthermore, the details map can be later refined using an erosion algorithm to enhance the terrain with more details (see Section 6).

4. Vectorization

The optimization process involves two main steps: initializing primitives and a loop of optimization and regularization iterations.

Initialization defines the starting distribution of primitives over the domain with a user-controlled number of Gaussian primitives $\mathcal{E}$ located at random positions, rotation, and β parameters. The initial scale is randomly chosen from five predefined values, ranging from large to small (specifically 0.06, 0.04, 0.03, 0.02, and 0.01 in our implementation). The terrain is normalized within the range $[-1,1]$, meaning that primitives are evaluated with values scaled to this fixed interval for consistency and simplicity. The amplitude a of the primitive is then defined from the terrain elevation $\mathcal{T}$ by smoothing the elevation around its centre $\mathbf{c}_i$: $a \propto h * k$ where k denotes a smoothing kernel over the ball $B(\mathbf{c},\sigma)$.

The optimization phase often generates very few negative elevation primitives as a result of regularization constraints unless they are specifically set during the initialization process. It is possible to ensure their presence by identifying and placing negative elevation primitives in ravines or erosion areas. The process starts by placing large primitives with amplitudes matching the smoothed ground truth. The error is then calculated as the difference between the placed primitives and the ground truth. Based on this error, the second level of primitives is generated. This procedure is repeated iteratively for all subsequent levels.

Optimization takes place after the initialization and optimizes the primitive parameters using gradient descent (Figure 5). We start by rasterizing the primitives $\mathcal{E}$ into scalar fields $\widetilde{\mathcal{T}}$ at a defined resolution. We then compute the influence of every primitive within the limits of the terrain Ω, and sum the contributions to obtain the final elevation h. The optimization is performed using a loss function defined as the weighted average of three terms:

$$\ell = \alpha_1\,\ell_1 + \alpha_2\,\ell_{LPIPS} + (1-\alpha_1-\alpha_2)\,\ell_{SSIM}$$

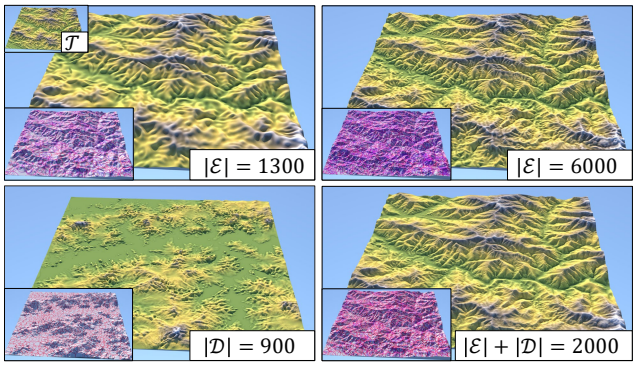

Figure 5: *Comparison between the ground truth and terrain approximations using different numbers of primitives.*

The cell-wise ℓ_1 loss is defined as:

$$\ell_1 = \sum_{i,j} |\widetilde{\mathcal{T}}_{ij} - \mathcal{T}_{ij}|$$

The second term relies on the learned perceptual image patch similarity (LPIPS) loss proposed in [ZIE*18], which demonstrates to reconstruct images with high perceptual quality and that proves to be accurate in a digital elevation model context [PPB*23]:

$$\ell_{LPIPS} = \sum_{i,j} (F(\widetilde{\mathcal{T}}_{ij}) - F(\mathcal{T}_{ij}))^2$$

The term F denotes the perceptual feature extractor, a VGG [LD15] pre-trained network. Finally, we use a structural similarity (SSIM) loss to take into account contrast and structure:

$$\ell_{SSIM} = \frac{(2\mu_x\mu_y + a)(2\sigma_{xy} + b)}{(\mu_x^2 + \mu_y^2 + a)(\sigma_x^2 + \sigma_y^2 + b)}$$

The terms μ and σ denote the mean and the variance of cell elevation, σ_{xy} the correlation coefficients between cells, and a, b numerical constants, respectively set to 0.01^2 and 0.03^2.

In our experiments, we use $\alpha_1 = 0.5$, $\alpha_2 = 0.25$ (thus $1-\alpha_1-\alpha_2 = 0.25$ for SSIM loss). Note that the optimization code will be released upon acceptance.

Regularization steps improve the quality of the reconstruction. We conform to the cloning and splitting described in [KKLD23], with a dynamic number of primitives during optimization. We propose the following additional regularization operations specific to our use case. During the optimization phase, the user chooses a predetermined fixed cell size c to compute the loss terms with the reference terrain $\mathcal{T}$ (Figure 6). Primitives with a radius smaller than c cause aliasing artefacts such as small spikes or holes, which are only visible when rasterizing the vector-based terrain at a higher resolution. We address this issue by discarding primitives whose scale satisfies the condition $3\sigma \leq c$.

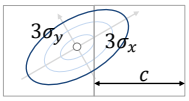

Figure 6: *Primitive pruning.*

We also introduce the following two regularization steps independent of terrain resolution. First, we remove the elongated prim-

itives: $\mathcal{E}_i$ is discarded if $\sigma_x/\sigma_y > t_e$ or $\sigma_y/\sigma_x > t_e$ with t_e denoting the elongation threshold (set to $t_e = 20$ in our experiments). Second, we discard primitives with a small amplitude, if $|a_i| < t_a$ with t_a an elevation threshold (set to $t_a = 10^{-3}$ in our experiments).

5. Vector-based terrain authoring

Here we present specific tools that take advantage of the vector-based representation for editing large-scale terrains. More precisely, we propose tools for manipulating the set of primitives $\mathcal{P} = \{\mathcal{E}, \mathcal{D}\}$ without the need for manual specification or modification of their parameters, as was required in [GGP*15]. This involves displacing and deforming these primitives using a skeleton $\mathcal{S}$ derived from the characteristic landforms of the terrain.

This method allows for meaningful and geomorphologically consistent authoring, which is difficult to accomplish directly on grid-based terrains because they lack inherent structural information. In contrast, vector-based representation is ideal for skeleton-based deformations, providing a flexible and straightforward way to make various modifications to the terrain by altering the primitives connected to its edges.

5.1. Geomorphology-based authoring

Our approach comes from the observation that peaks, saddles and other terrain features such as crest lines can be connected into geometric graph structures as described in [AGP*19]. Therefore, taking inspiration from rigging in animation, we propose high-level tools (Figure 7) based on skeletons derived from the geomorphological structure of the terrain. The process consists of three steps: creating the graph, attaching primitives to graph edges, and propagating graph deformations.

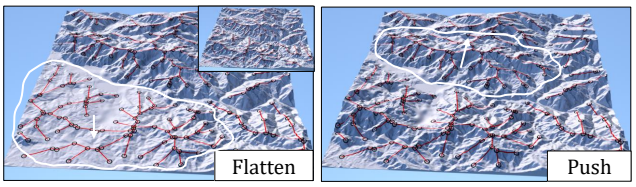

Figure 7: *High-level tool for editing the geomorphological structure of the terrain: deformations propagate along the graph and conform to landforms such as mountain ranges of valleys.*

Crest graph creation. Given an input raster terrain, we invert elevations and identify crest cells using a river detection algorithm described in [BLM14] and [PDG*19]. We obtain an initial graph $\mathcal{G}$, composed of a set of nodes $\mathcal{N}$, connecting cells that we simplify to get the final vector-based geometric graph that will be used as a manipulation skeleton $\mathcal{S}$. We rely on an edge-collapse strategy to perform the graph simplification. Let d_0 denote a minimal distance threshold; we remove an edge $\mathbf{e}_{ij}$ connecting two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ if its length is larger than d_0 (Figure 8). $\mathcal{N}_i$ and $\mathcal{N}_j$ are merged by this process to form a single new node. This process is sufficient to obtain a coherent manipulation skeleton with almost evenly spaced nodes.

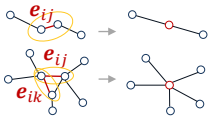

Figure 8: *Graph simplification.*

Graph deformations. We first link the set of primitives $\mathcal{P}$ to the skeleton $\mathcal{S}$ by computing the minimum distance between the centre $\mathbf{c}_i$ of every primitive $\mathcal{P}_i$ and $\mathcal{S}$ (Figure 9). The projection of $\mathbf{c}_i$ onto the skeleton is noted $\pi(\mathbf{c}_i)$ and the associated graph distance to the modification $l(\pi(\mathbf{c}_i))$.

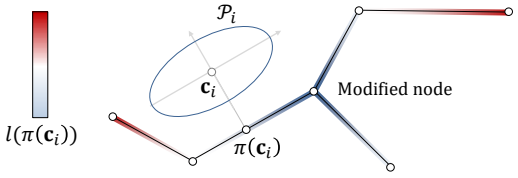

Figure 9: *Primitives are linked to the edges of the geometric graph, an associated graph distance to the modified node is computed.*

Based on this graph distance, we build an attenuation factor α_i with a step function s:

$$\alpha_i = s\,(1 - l(\pi(\mathbf{c}_i))/d_m)$$

d_m is a user-defined threshold that will influence the rigidity and locality of modifications. By default, it is initialized with the maximum graph distance.

Move. When a node of the graph is moved by a translation vector $\mathbf{v}$, the whole set of nodes is modified according to the attenuation factor: $\mathbf{n}_i' = \mathbf{n}_i + \alpha_i \mathbf{v}$, with $\mathbf{n}_i$ the position of the node $\mathcal{N}_i \in \mathcal{N}$. It is necessary to adjust the scale, rotation, and position of each primitive according to the movement of the nodes and edges. Let $\pi(\mathbf{c}_i)$ be the projection of the centre of $\mathcal{P}_i$ on the edge $\mathbf{e}_j$ and $\pi(\mathbf{c}_i')$ the projection on $\mathbf{e}_j'$, the translated edge. The vector $\mathbf{v}_i$, where $\mathbf{v}_i$ is the specific translation vector for the primitive $\mathcal{P}_i$, is defined by $\pi(\mathbf{c}_i') - \pi(\mathbf{c}_i)$ and θ is the angle between $\mathbf{e}_j$ and $\mathbf{e}_j'$. The primitive $\mathcal{P}_i$ is updated by applying the translation vector $\mathbf{v}_i$ on its centre $\mathbf{c}_i$ and rotation θ on its angle θ_i. It is important to take scaling into account in order to deform large structures while maintaining global coherence. Scaling a rotated ellipse with respect to an axis changes its rotation and scale independently (see Appendix A).

Amplitude modification. The graph allows users to directly modify the amplitude of specific parts of the terrain. The new amplitude a_i' of the primitive $\mathcal{P}_i$ is defined as $a_i' = a_i\,(\alpha_i\,u + 1)$ where a_i is the initial amplitude and $u \in [-1; +\infty[$ the user modification factor.

5.2. Landforms tools

In addition to the previous tools based on a geomorphological skeleton, we offer more conventional editing features. The tools we have developed enable essential functions such as deleting shapes and selecting specific areas, which can then be moved, rotated, or resized. To enhance user-friendliness, users can interact using draggable regions on the terrain. They can activate and apply the selected tool by clicking and moving the selected region.

Region selection. The proposed toolset enables various operations, including free placement (translation, rotation, scaling), warping (twist and squeeze), and editing with erase, copy-paste, or cut-paste functions specifically adapted to landforms.

We have incorporated two different selection shapes for these tools: circular regions and free-form contours defined through a lasso-like selection directly on the terrain (see accompanying video). Let Ω denote the selected region (Figure 10). We define $\mathcal{P}(\Omega) = \{\mathcal{P}_i, \mathbf{c}_i \in \Omega\}$ as the set of primitives whose centres are inside the region. In contrast, $\overline{\mathcal{P}}(\Omega)$ represents the set of primitives with $\mathbf{c}_i$ outside Ω. We have $\overline{\mathcal{P}}(\Omega) \cup \mathcal{P}(\Omega) = \mathcal{P}$ and $\overline{\mathcal{P}}(\Omega) \cap \mathcal{P}(\Omega) = \varnothing$.

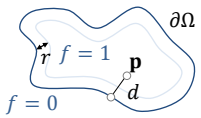

Figure 10: *Region definition.*

Similarly to skeleton-based deformations, the impact of changes is attenuated (Figure 10) by applying a compactly supported step function s to the distance to the boundary of the domain $d(\partial\Omega, \mathbf{p})$; let r denote a user-defined influence radius, we have:

$$f(\mathbf{p}) = s\left(\frac{d(\partial\Omega, \mathbf{p})}{r}\right)$$

Landform tools are also used in combination with geomorphological-based tools. To avoid unnatural overlapping primitives when the user edits the crest graph, we propagate modifications in an influence region around the convex hull of modifications.

Landforms stamping. Once the region is selected, the user can move, scale, and rotate it (Figure 11). This directly impacts the centre $\mathbf{c}_i$, the scale σ_i, and the rotation angle θ_i of primitives in $\mathcal{P}(\Omega)$. We denote $\widetilde{\Omega}$ the transformed region. It is likely that $\overline{\mathcal{P}}(\Omega) \cap \mathcal{P}(\widetilde{\Omega}) \neq \varnothing$, which means that the primitives that are transformed overlap other (non-transformed) primitives. To avoid unwanted blending and bulges, we dampen amplitude in $\overline{\mathcal{P}}(\Omega)$ by using factor $f(\mathbf{c}_i)$:

$$\forall \mathcal{P}_i \in \overline{\mathcal{P}}(\Omega) \qquad h_i = f(\mathbf{c}_i)\, h_i$$

Another option proposed to the user is to simply vanish the amplitude of primitives in $\overline{\mathcal{P}}(\Omega)$, which is equivalent to having a radius r that tends to infinity. Remember that when we modify the parameters of primitives, it does not harm the smoothness of the terrain, unlike standard grid-based heightfield editing methods, because primitives define a continuous elevation function. We also included a tool to delete primitives within a selected region to free up space for rearranging the surrounding terrain and optimizing the scene organization.

Landform stamping can be done using areas from the existing terrain or landforms from a precomputed vector-based atlas, which facilitates terrain authoring. Our implementation includes a variety of mountain ranges and landform archetypes created from digital elevation models. The user can also enhance the atlas by adding a structure $\mathcal{P}(\Omega)$ from a selected region.

Twisting and squashing. Contrary to landform stamping, warping deforms the terrain within the control region Ω while maintaining its consistency outside (Figure 12). The squeeze tool achieves a

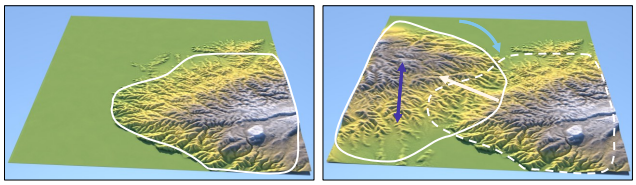

Figure 11: *A selected region Ω is translated, scaled and rotated, facilitating terrain authoring from selected patches.*

translation weighted by $f(\mathbf{p})$. When primitives get closer or farther, the scaling needs to be updated (see Section 5.1). Thus primitives are scaled in the direction of the translation and by a factor of $f(\mathbf{c}_i) - f(\mathbf{c}'_i) + 1$ where $\mathbf{c}'_i$ is the centre after the transformation. The twist tool is defined similarly to the twisting operator introduced in [GGP*15] that rotates the points with an angle proportional to the distance to a reference point inside Ω. The fall-off function f guarantees smooth continuity with the surrounding terrain.

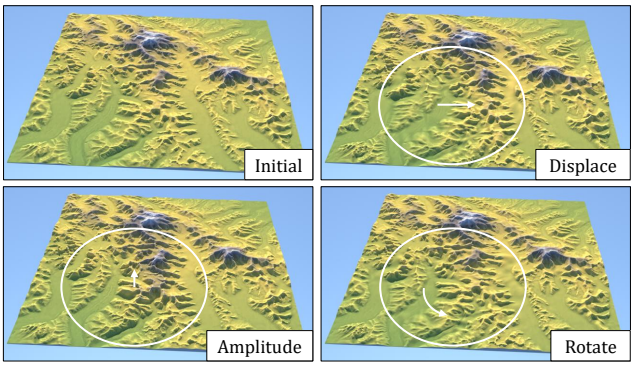

Figure 12: *Operators modify the terrain by changing the location, angle of rotation and the amplitude of the primitives within the range of Ω.*

6. Results and discussion

Vector-based representation is a powerful model for encoding topography, offering significant advantages over traditional digital elevation-based approaches. It allows for the development of tools that can interact with the terrain at a structural level, providing more precise and flexible editing capabilities (Figure 13 and 14). In contrast, raster terrains, which lack an underlying structure, often show challenges in editing as modifications can be cumbersome and prone to introducing artefacts or discontinuities. Vector-based representation addresses these issues by providing an underlying framework where primitives can be directly assigned to the structural elements of the terrain.

Our model supports outputs at any resolution, facilitating real time editing of large-scale terrains. From an editing standpoint, the workflow starts either with a grid-based terrain, which is then vectorized to create the error map $\Delta\mathcal{T}$, or with an atlas of existing terrain fragments. The artist utilizes the tools described in Section 5, such as the geomorphological skeleton, along with various moving

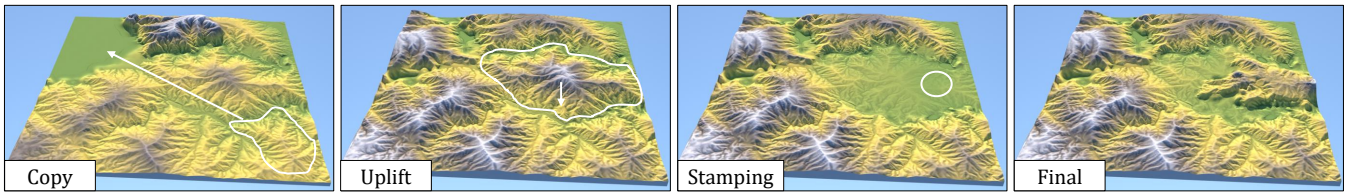

Figure 13: *Snapshots of a terrain editing sequence with the successive application of copy, terrain uplifting, and stamping tools.*

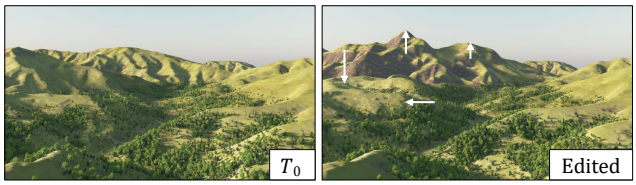

Figure 14: *Comparison between an input raster terrain and its vector-based edited version.*

Optimization		Rasterization (1024×1024)		
$\#\mathcal{E}$	Time (s)	$\#\mathcal{D}$	$\#\mathcal{E} + \#\mathcal{D} = \#\mathcal{P}$	Time (ms)
$\approx$ 8k	58 s	2k	10k	15 ms
15k	59 s	2k	17k	16 ms
30k	69 s	2k	32k	17 ms
60k	85 s	2k	62k	21 ms

Table 1: *Vectorization and rendering times (in seconds and in milliseconds respectively) for different scene complexities.*

and warping operations, to shape the terrain according to their creative vision. Once the editing process is complete, the vector-based terrain can be converted back into a grid format, allowing for a wide range of downstream applications.

During the editing process, the terrain is rasterized at a normal resolution (typically 1024×1024) to ensure responsiveness. Once editing is complete, the terrain can be exported at a higher resolution. Users may also apply a simulation algorithm to refine and enhance terrain features further. Figure 15 illustrates the workflow for terrain editing and high-resolution export. Following Schott et al. (2024) [SGG*24], a simulation method generates an eroded version of the edited terrain, after which an error map $\Delta\mathcal{T}$ is computed. This error map is then leveraged to introduce detail primitives $\mathcal{D}$ into the vector representation, enabling high-resolution terrain export at 4096×4096 or higher.

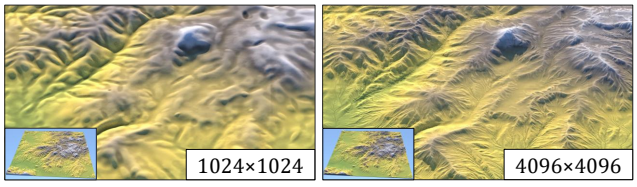

Figure 15: *Real-time terrain editing is conducted at a lower rasterized resolution to ensure responsiveness, while a higher-resolution version is generated for final export.*

6.1. Implementation and performance

Our implementation utilizes PyTorch for the optimization task. We perform the optimization process using the ADAM optimizer, with a learning rate of 10^{-2}. The vectorization of a 1024×1024 terrain runs in approximately ≈ 60 seconds with $\approx 10\,000$ primitives on an NVidia 3090 with 24GB of dedicated memory. Building on the research by [KKLD23], we have developed a differentiable rasterizer using C++ and CUDA adapted to our case for the optimizer.

In order to edit the vector representation without the need to differentiate it, we utilize a compute shader to accelerate elevation computation during terrain rasterization. First, we organize the primitives into a loose grid acceleration structure to compute the elevation function h_i only for the primitives whose region of influence intersects the cells. This method enables real-time rasterization on an NVIDIA 3090, achieving 21 ms for a 1024×1024 terrain resolution. The time increases linearly to 388 ms for a 4096×4096 terrain, still allowing for interactive time editing. Rasterization and rendering of the terrain on the GPU enables real-time feedback and editing, significantly enhancing the user experience (see accompanying video). Timings for various landscapes, with varying numbers of primitives, are reported in Table 1.

6.2. Comparisons

We compared our approach to other interactive terrain authoring methods. A key feature of the vector-based model is its ability to author high-resolution terrains in real time. In contrast, existing methods are often restricted to a resolution of 1024×1024, and they experience a significant drop in speed at higher resolutions.

We assess our approach by comparing it with several existing methods, trying to reproduce a lambda-shaped mountain range as depicted in Figure 16. The methods we considered include fully automatic terrain generation techniques that utilize sketch inputs and require no manual intervention [PPB*23, LGP*23, GDGP16, SPF*23], along with approaches that involve detailed manual editing, such as those in [GPM*22] and our own. To ensure a fair comparison, we tailored the input data to meet the requirements of each method. For instance, [PPB*23] was given a low-resolution

Article	Data	*SC*	Resolution	Scale	Time
[GDGP16]	●		∞	30 km	1 Hz
[GPM*22]		○	1024^2	$\sim$ 30 km	1 Hz
[PPB*23]	●		1024^2	30 km	2 Hz
[LGP*23]	●	○	256^2	5 km	20 Hz
[SPF*23]		○	2048^2	100 km	60 Hz
Ours	●	●	∞	$\sim$ 30 km	60 Hz

Table 2: *Comparison of recent methods for creating and editing terrain.*

sketch, while [LGP*23] and [GDGP16] received curve constraints extracted from the same sketch. For all methods that involved manual editing, we aimed to reproduce the lambda shape as accurately as possible.

Table 2 outlines the key differences between the evaluated methods. The *Data* column indicates whether each method requires example data as input. *Structural Control* (*SC*) refers to the level of manipulation possible with terrain features: a bullet point (●) signifies direct and structural control, while a circle (○) represents indirect or limited control. Methods that can maintain an interaction rate of 60 Hz or higher are deemed capable of supporting real-time editing.

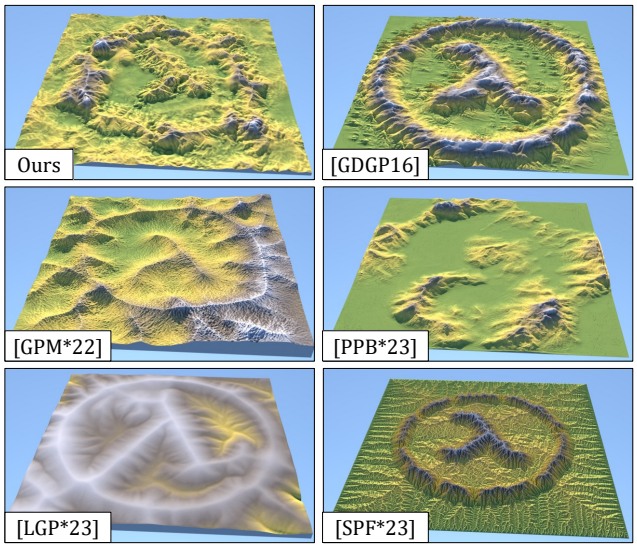

Figure 16: *Comparison of terrain generation and editing methods for modeling a lambda-shaped feature.*

While techniques like [PPB*23] and [SPF*23] generally preserve geomorphological coherence—sometimes at the expense of reconstruction fidelity or repetition—our method does not emphasize this criterion. Instead, it uniquely enables direct, real-time structural editing while supporting output at arbitrary resolutions. It is also worth noting that each method targets a different terrain scale, as indicated in Table 2, which may affect both the perceived quality and suitability of the results for specific applications.

Learning approaches There have been proposals for interactive editing systems based on deep learning methods to incorporate style [PPB*23, LGP*23]. These systems are implemented as add-ons for Blender and allow for direct terrain synthesis and amplification. Perche *et al.* [PPB*23] use a modified StyleGAN model to match an input sketch in the latent space and produce a terrain with landform details at a moderate resolution. This type of terrain sketching is not achievable with our framework. However, users can choose from a selection of precomputed vector-based landforms sourced from various places, including vectorized digital elevation model datasets or synthetically generated atlases, and position them. Lochner *et al.* [LGP*23] introduced a curve-based sketching model that generates a terrain using a diffusion method. While we do not utilize guiding curves, the vector-based representation allows for placing and assembling complex landforms and deforming them using geomorphologically coherent skeletons. Among these methods, only the diffusion-based method operates in real time, albeit with an asynchronous update during user interaction. Moreover, they are limited to 256×256 [LGP*23] and 1024×1024 [PPB*23] resolution patches and require stitching or blending neighbouring patches to handle larger resolutions, at the price of a higher response time.

Sketches approaches Guerin et al. [GDGP16] introduced a sparse modeling technique that synthesizes terrains using patches selected from an example-learned dictionary. However, a significant limitation of this approach is its tendency to produce repetitive artifacts, especially in large, flat areas such as plains or plateaus. Furthermore, user control is somewhat indirect, as interactions are mainly conducted through a sketching interface. The gradient representation proposed by Guerin *et al.* [GPM*22] synthesizes terrain using a set of curve-based hard (elevation) and soft (slope) constraints by solving a Poisson equation. However, support for high-level tools is limited to generating mountain ranges with a dedicated brush, and once created, modifications to the terrain are not easy.

Simulations Schott *et al.* [SPF*23] presented an interactive terrain modelling system based on stream power erosion. This approach ensures that the synthesized terrain remains geomorphologically consistent, as elevation is computed by solving the stream power equation based on a prescribed uplift map. While this method is intuitive for large-scale terrains, it does not allow for precise manipulation and control of landforms.

Image-based approaches Gain *et al.* [GMM15] developed a technique for generating terrains using texture synthesis and interactive editing. In contrast, manipulating the guiding curves in their method only indirectly influences the result, which is closely tied to the provided exemplars. Tasse *et al.* [TEC*14] introduced a first-person interface that enables users to draw silhouette lines and automatically create mountain features at inferred locations. This approach is practical for visualizing and authoring directly from an end-user perspective. However, our model does not support such interaction, as the tools we have implemented are designed for an upper or diagonal view.

6.3. Validation

Validating an interactive authoring tool is always a challenge. Instead of conducting a user study with novice users, we opted to gather feedback from professional landscape technical artists in the entertainment industry. We conducted one-hour interviews with them, during which they tested the application, using all the tools and techniques outlined in this work, from editing existing terrains to modeling with saved atlases. To ensure a smooth testing process, we addressed all questions the experts had regarding the application's functionality. Their feedback highlighted both the strengths and challenges of our vector-based terrain model.

Designers were enthusiastic about the vector-based model due to its ease of manipulation, particularly for creating large terrains. They emphasized that while low-resolution editing is acceptable during the creation process, high-resolution outputs are essential for final validation, which our vector-based approach can provide. In contrast, concerns were raised regarding the challenges of modifications using the skeleton-based tools, as large deformations can lead to inconsistent or undesirable topography. Designers recommended implementing landform-based rigging safeguards and developing procedural generation methods to explore the diverse range of models that can be created from basic shapes.

Finally, the system received positive feedback for its potential to facilitate the creation and editing of user-generated content, with its simplicity making it accessible to non-experts.

6.4. Limitations

Our method, while effective, is not without limitations. First and foremost, the optimization process does not establish a bijective mapping between grid-based and vector representations. In this regard, the gradient representation [GPM*22] demonstrates superior properties. Figure 17 and Table 3 illustrate the loss as the number of primitives increases. As the number of primitives rises, the error decreases, resulting in models that are sufficiently accurate for precise editing purposes. Furthermore, detail primitives are employed to capture this vectorization error, allowing for a near-perfect recovery of the initial terrain.

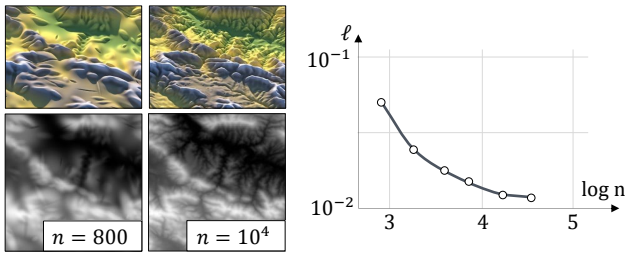

Figure 17: *Visualization of the approximation of an initial terrain with an increasing number of primitives (left). As the number of primitives increases, the final loss decreases (right) and the terrain reconstruction is more accurate.*

Another limitation is that extensive user modifications may compromise the consistency of the terrain (Figure 18). Altering primitives or making significant changes to the skeleton can disrupt the coherence of the model, resulting in inconsistencies such as basins or unnatural terrain features. While basins can be edited or filled with data and are not a major concern, colliding mountains may cause a loss of geomorphological structure in the terrain. A common challenge faced by editing tools that do not inherently ensure consistency within the model is maintaining global geomorphological and hydrological coherence, as discussed in [SPF*23].

$\#\mathcal{P}$	Loss $\times 10^{-3}$			
	ℓ_1	ℓ_{SSIM}	ℓ_{LPIPS}	ℓ
0,8 k	5.65	5.65	47.2	58.6
1,5 k	4.08	4.06	30.3	38.5
3 k	2.92	2.62	16.8	22.5
6 k	2.24	1.63	8.92	12.8
11 k	2.21	1.31	7.64	11.1
20 k	2.78	1.25	6.11	10.1

Table 3: *Metrics for different numbers of primitives $\#\mathcal{P}$.*

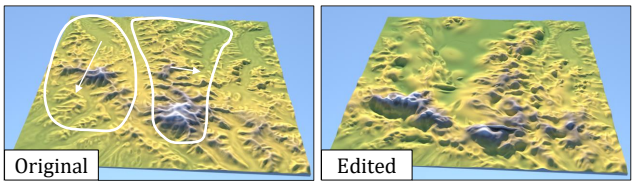

Figure 18: *Our method may produce inconsistencies or empty regions in the terrain when skeletal modifications are overly extensive, leading to colliding features.*

7. Conclusion

Vector-based representations form a strong foundation for creating large-scale high-resolution terrains (Figure 19). We have developed a vector-based representation consisting of a combination of simple primitives that allow for both the creation and modification of existing terrains. These primitives enable smooth blending of terrain sections and provide a wide range of tools for modelling landforms at a high level. We have also presented an efficient optimization process that allows the conversion of real terrains or terrains generated from various procedural and simulation frameworks.

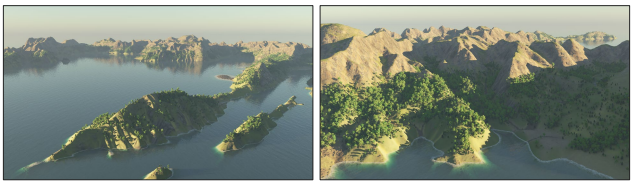

Figure 19: *Example of volcanic islands authored by combining and deforming predefined models selected from an atlas.*

Our approach is the first to be based on geomorphological skeletons for terrain editing. Through experiments with professional

landscape artists, we have demonstrated that our approach is effective for creating and editing large-scale digital terrains with a high level of detail. Future work could involve expanding this vector model by providing advanced tools that enable users to edit terrains while maintaining geomorphological accuracy. Additionally, implementing a procedural generation method for placing primitives could significantly enhance the possibilities for creating terrains. Finally, detailed primitives use an error map that matches the size of the original terrain. Multiple layers of noise or a sparse dictionary-based approach [GDGP16] could be employed to encode details in a more efficient manner.

Acknowledgments

This work was part of the projects AMPLI ANR-20-CE23-0001 and EOLE ANR-23-CE56-0008, supported by Agence Nationale de la Recherche Française.

Conflicts of Interest

The authors declare no conflicts of interest for this work.

Data Availability Statement

The project page, available here, provides the source code and the presentation video.

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. 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. 5

[BLM14] BARNES R., LEHMAN C., MULLA D.: An efficient assignment of drainage direction over flat surfaces in raster digital elevation models. *Computers and Geosciences 62* (2014), 128–135. 5

[CCB*18] CORDONNIER G., CANI M.-P., BENES B., BRAUN J., GALIN E.: Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE Transactions on Visualization and Computer Graphics 24*, 5 (2018), 1756–1769. 2, 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 36*, 6 (2017). 2, 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, 7, 8, 10

[GGP*15] GÉNEVAUX J.-D., GALIN E., PEYTAVIE A., GUÉRIN E., BRIQUET C., GROSBELLET F., BENES B.: Terrain Modeling from Feature Primitives. *Computer Graphics Forum 34*, 6 (2015), 198–210. 2, 5, 6

[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) 38*, 2 (2019), 553–577. 2

[GMM15] GAIN J., MERRY B., MARAIS P.: Parallel, Realistic and Controllable Terrain Synthesis. *Computer Graphics Forum 34*, 2 (2015), 105–116. 2, 8, 11

[GPM*22] GUERIN E., PEYTAVIE A., MASNOU S., DIGNE J., AND-JAMES GAIN B. S., GALIN E.: Gradient Terrain Authoring. *Computer Graphics Forum (proceedings of Eurographics 2022) 44*, 2 (2022), 85–95. 1, 2, 7, 8, 9, 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, 11

[HMKQ*24] HAMDI A., MELAS-KYRIAZI L., QIAN G., MAI J., LIU R., VONDRICK C., GHANEM B., VEDALDI A.: Ges: Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024). 3

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics 42*, 4 (2023). 2, 4, 7

[LD15] LIU S., DENG W.: Very deep convolutional neural network based image classification using small training sample size. In *3rd IAPR Asian Conference on Pattern Recognition* (2015), pp. 730–734. 4

[LGP*23] LOCHNER J., GAIN J., PERCHE S., PEYTAVIE A., GALIN E., GUÉRIN E.: Interactive authoring of terrain using diffusion models. *Computer Graphics Forum 42*, 7 (2023). 2, 7, 8, 11

[PDG*19] PEYTAVIE A., DUPONT T., GUÉRIN E., CORTIAL Y., BENES B., GAIN J., GALIN E.: Procedural riverscapes. *Computer Graphics Forum 38*, 7 (2019), 35–46. 5

[PPB*23] PERCHE S., PEYTAVIE A., BENES B., GALIN E., GUÉRIN E.: Authoring Terrains with Spatialised Style. *Computer Graphics Forum* (2023). 2, 4, 7, 8, 11

[SGG*24] SCHOTT H., GALIN E., GUÉRIN E., PEYTAVIE A., PARIS A.: Terrain Amplification using Multi-scale Erosion. *ACM Transactions on Graphics 43*, 5 (2024). 1, 7, 11

[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, 7, 8, 9, 11

[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. 8, 11

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018). 4

[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. 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. 2

Appendix A: Ellipse scaling

Without loss of generality, consider the equation of a rotated ellipse centered at the origin:

$$(\cos\theta x + \sin\theta y)^2/a^2 + (\sin\theta x - \cos\theta y)^2/b^2 = 1$$

The coefficients of the corresponding polynomial $c_{00}x^2 + c_{10}xy + c_{01}y^2$ are as follows:

$$c_{00} = c^2/a^2 + s^2/b^2 \quad c_{10} = cs/a^2 - 2sc/b^2 \quad c_{01} = s^2/a^2 + c^2/b^2$$

It is possible to retrieve the coefficients $1/a^2$, $1/b^2$, and θ by computing the eigenvalue and the first eigenvector $(\cos\theta, \sin\theta)$ of the

2×2 matrix:

$$\begin{pmatrix} c_{00} & c_{10}/2 \\ c_{10}/2 & c_{01} \end{pmatrix}$$

If we scale an ellipse by a factor (u, v), then the parameters of the scaled ellipse $\tilde{a}$, $\tilde{b}$ and $\tilde{\theta}$ can be found by computing the eigenvalue and the first eigenvector of:

$$\begin{pmatrix} c_{00}/u^2 & c_{10}/2uv \\ c_{10}/2uv & c_{01}/v^2 \end{pmatrix}$$

From this we compute $\tan\tilde{\theta}$, and finally $\tilde{\theta}$.

Appendix B: Overview of authoring techniques

Table 4 compares the different types of user control provided and assesses their relative performance when applied to heightfields at a resolution of 1024×1024. *Structural* control refers to the ability to manipulate terrain features: ● stands for full control, whereas ○ denotes indirect control.

	Article	Amplification	Edition	Real-time	Copy-paste	Vector-based	Structural
Procedural	Ours		●	●	●	●	●
	[HGA*10]					●	○
	[TEC*14]		●				○
	[GPM*22]		●		●		○
Example-based	[GMM15]			●		●	
	[GDG*17]	●	○				○
	[ZLZ*22]						
	[PPB*23]	●	●		●		
	[LGP*23]	●	●	●			○
Simulation	[CCB*18]	●	●				
	[SPF*23]		○		●	○	○
	[SGG*24]	○					○

Table 4: *Comparison of interactive tools available in different terrain authoring methods.*

The columns represent the following criteria. Amplification: can details be added to an existing terrain? Editing: is it possible to work with an actual input terrain? Real-time: does the method allow for real-time ($\geq$ 10Hz) or interactive ($1-10$Hz) feedback for modeling operations? Copy-paste: do the tools include a copy-paste operation? Vector-based: is the model off-lattice (as opposed to grid-based)? Structural: does the model allow for high-level tools related to the geomorphological structure of the terrain?