DOI: 10.1111/cgf.70389

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Terrain Synthesis and Authoring based on Iso-Contours

B. Huftier[1], H. Schott[1], E. Galin[1], O. Argudo[3], A. Peytavie[1] and E. Guérin[2]

[1] LIRIS, Université Claude Bernard Lyon 1, CNRS, France [2] LIRIS, INSA-Lyon, CNRS, France [3] Universitat Politècnica de Catalunya, Spain

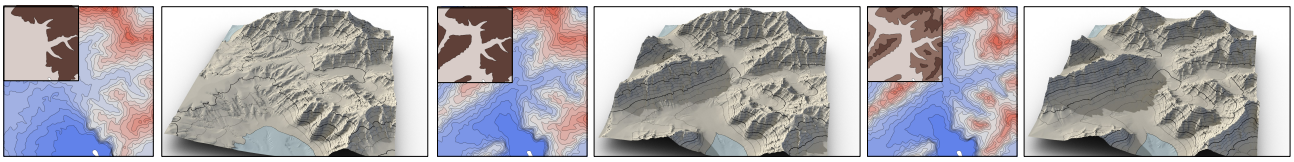

Figure 1: *Given a sparse set of nested input contours, our method automatically synthesizes a dense contour-based representation of the terrain using an Open Eden Growth approach. It supports the generation of complex landforms such as mesas, coastlines, and ridgelines.*

Abstract

Digital terrains are central to realistic landscape depiction, yet authoring tools must balance perceptual realism with intuitive artistic control. We propose a compact vector-based representation that models terrain as nested iso-contours, inspired by geomorphology and cartography. Our method departs from traditional grid-based elevation models by generating contours through an inward Open Eden Growth simulation, followed by marching-triangles reconstruction into a Triangulated Irregular Network. This contour framework supports direct editing such as warping, slope modulation, and smoothing, while allowing reconstruction of a standard elevation map for downstream processing, including erosion and amplification. The approach enables the creation of diverse, realistic terrains from minimal user input and offers simple yet powerful control for designers.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;***

1. Introduction

Terrain synthesis remains a central topic in computer graphics, with applications ranging from open-world games and simulation to virtual cinematography and geographic visualization. Early approaches focused on procedural fractal noise to mimic elevation patterns such as peaks and valleys. Subsequent research emphasized more hydrologically coherent modeling through physically-based simulations of geological processes or by assembling terrains from real-world elevation exemplars.

However, the synthesis of realistic terrains that balances scalability, control, and realism remains an open challenge. Natural landforms exhibit considerable structural diversity – from canyons and mesas to wetlands and alpine ridges – which are difficult to reproduce within a unified framework. Authoring tools should support intuitive workflows while ensuring physically plausible results, particularly for large-scale or real-time development of environments. In this context, the need persists for modeling techniques that combine topological flexibility, interactive performance, and compatibility with both procedural and simulation-driven augmentation.

A wide range of techniques has been proposed to address these goals, including procedural terrain generation with erosion models, large-scale tectonic simulations, and data-driven approaches ranging from texture-synthesis-inspired methods [ZSTR07] to sparse representations [GDGP16], curve-based models [HGA*10, GPM*22] and deep generative models [PPB*23, LGP*23]. Despite significant advances, most existing methods rely on grid-based elevation maps, typically limited in resolution due to memory and computation constraints. Real-time synthesis is generally restricted to resolutions of 1024×1024, or up to 4096×4096 using amplification techniques [SGG*24] taking advantage of parallelization on graphics hardware.

Our research stems from the observation that the characteristics of terrain can be effectively represented using a set of nested iso-contours, as seen in geomorphology and cartography. We propose an alternative vector-based representation for terrain modeling, using nested iso-contours as the primary elevation descriptor. We further introduce a procedural method for generating terrain contours, offering direct control over elevation distributions and supporting

the synthesis of complex landforms such as mesas, coastlines, and ridgelines (Figure 1). Moreover, this contour-based formalism enables intuitive editing operations such as warping, and integrates seamlessly with simulation-based and procedural modeling application. An elevation model can always be recovered through terrain reconstruction, which can be further enhanced with erosion simulations. Our formulation bridges the gap between vector-based models and grid-based terrain synthesis, offering a flexible, extensible, and designer-controllable approach to terrain generation.

2. Related work

Terrain generation and amplification techniques in computer graphics can be broadly classified into three categories: procedural generation, example-based, and erosion simulation. While erosion-based methods have received significant attention for their ability to simulate geomorphological processes, our work focuses instead on the modeling and synthesis of terrains based on curves and sketches. These methods directly manipulate contours as user-defined curves to describe key terrain features such as ridge lines, river networks, and coastlines, offering designers a direct control over landforms and style. This complements erosion approaches such as [SGG*24, TGSC24], which are physically motivated but often less intuitive for design tasks. For a comprehensive overview of digital terrain modeling, we refer the reader to the survey by Galin *et al.* [GGP*19].

Learning-based techniques. Several methods use machine learning to generate digital elevation models from user sketches or control curves. Guérin *et al.* [GDG*17] pioneered this direction using a Conditional Generative Adversarial Network, automatically deriving sketches from elevation data and inverting the process to let users create terrains interactively. Zhang *et al.* [ZLZ*22] introduced the notion of style in the process by expanding the concept to multiple discriminators specialized in different styles at different scales. Perche *et al.* [PPB*23] adapt the StyleGAN model to digital terrains to enable the generation of terrains from sketches with spatialized and style-oriented features. Cai *et al.* [CXY*22] introduced self-attention mechanisms in the architecture to capture global features and improve consistency. Recently, Liu *et al.* [LB25] addressed the data-set generation issue inherent to learning-based methods by proposing an algorithm with a single terrain exemplar. These methods are capable of generating visually plausible and hydrologically consistent terrains. In particular, the work of Lochner *et al.* [LGP*23] uses a diffusion model to demonstrate the ability to produce terrains free of endorheic basins, while conforming to observed power laws for drainage networks.

Despite their strengths, learning-based methods suffer from limitations in reproducibility, often producing varied outputs from identical inputs due to stochastic elements in their inference process. Moreover, the reliance on large-scale training datasets can limit generalizability and introduce biases. In contrast, our method is purely procedural, requiring no training phase and offering high predictability and reproducibility.

Curve-based terrain models. Several authors have explored the use of control curves as direct geometric controls to guide terrain synthesis. Zhou *et al.* [ZSTR07] introduced a texture-based synthesis approach that preserves structural features across synthesized patches. The generation process is guided by ridge curves and patches are seamlessly assembled using a Poisson reconstruction. Hnaidi *et al.* [HGA*10] made use of diffusion curves to interpolate elevation between feature curves. The feature curves are completed with slope control and the terrain details are produced by spatially varying noise. More recently, Guérin *et al.* [GPM*22] have proposed a modeling paradigm in the gradient domain, which allows seamless blending of disparate terrain features across multiple sources. In that framework, elevation is recovered via Poisson reconstruction, combining hard altitude constraints (Dirichlet boundary conditions) and soft gradient guiding. Another interesting approach consists in generating a river network and then synthesize a coherent terrain with local primitives [GGG*13]. Argudo *et al.* also provide a way to generate a terrain that matches orometric properties, and more precisely a divide tree that represents the crests hierarchy [AGP*19].

Those methods all have in common that they propose a curve-based representation of features at the core of the model, either representing hydrological or geomorphological features such as ridges and rivers. In contrast, we model terrains using iso-contours, closed elevation curves, as the fundamental primitive. This enables a more intuitive editing paradigm and facilitates procedural generation through shape composition and hierarchical refinement. To our knowledge, our approach is the first to focus explicitly on the procedural generation of contour hierarchies for terrain modeling.

Reconstruction. Although iso-contour reconstruction is not inherently a generative process, several techniques have been proposed for reconstructing continuous elevation fields from contours. The elevation of the terrain can be obtained by solving partial differential equations constrained by the elevation and gradient at the contours lines [CMN98]. Hormann *et al.* [HSS03] proposed an interpolation method that computes the height using an Hermite interpolation based on the shortest distances to the contours, and the height and derivative information. The generated surfaces are generally C^1 except at terrain characteristics such as ridges and valleys which are reconstructed as sharp features. The work of Guérin *et al.* [GPM*22] can also be used in a reconstruction process by assigning constraints to the contour lines and let the Poisson method interpolate between them. Constructive and primitive-based approaches are also good candidates for reconstructing terrains with procedural primitives [GGP*15] or using an optimized dictionary [GDGP16, AAC*17].

Our approach is different, instead of interpolating or reconstructing a terrain from iso-contours, we use them as a guide for a procedural terrain generation method.

3. Overview

Here we provide a high-level overview of our model along with definitions and notations, and an overview of our contour-based generation method.

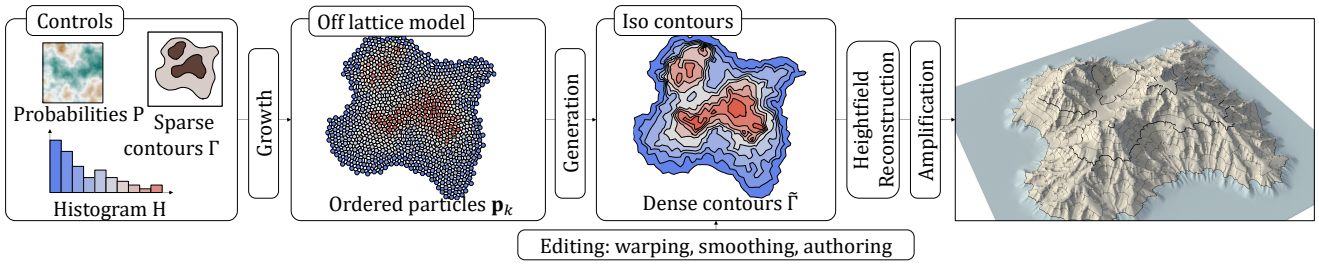

Figure 2: *Given an input set of sparse contours Γ, a histogram $\mathcal{H}$ and a probability map $\mathcal{P}$, our method synthesizes a dense hierarchy nested iso-contours $\widetilde{\Gamma}$ using an Eden Growth-based strategy. The designer can edit and re-synthesize contours $\widetilde{\Gamma}$, then convert the result into a standard heightfield $\mathcal{T}$ for post-processing effects, such as multi-scale erosion [SGG*24].*

3.1. Contour-based model

We depart from the traditional grid-based representation and consider the terrain $\mathcal{T}$ as a continuous vector-based scalar field $h : \Omega \rightarrow \mathbb{R}$ where $\Omega \subset \mathbb{R}^2$ is the compact region where $\mathcal{T}$ is defined.

Iso-contours provide an efficient and compact representation of this field. We define an iso-contour Γ_i as the closed continuous border of a compact region $\Omega_i \in \mathbb{R}^2$ at a specific height h_i, formally $\Gamma_i = \partial\Omega_i$ and $\forall \mathbf{p} \in \Omega_i$, $h(\mathbf{p}) \geq h_i$. The entire set of contours is denoted as Γ. For any height h_i, the region Ω_i is a subset of the overall domain Ω, *i.e.* $\Omega_i \subseteq \Omega$.

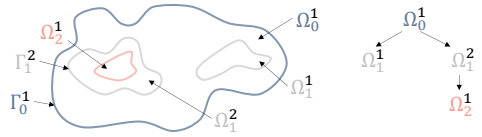

Figure 3: *A terrain defined by a nested contour hierarchy.*

For a given height h, we denote the set of contours Γ_i at elevation $h_i = h$ as Γ_h as illustrated in Figure 3. We also define the corresponding region Ω_i as the union of the interiors of all contours at height h_i: $\Omega_i = \cup_j \Omega_i^j$. Moreover, we enforce the property that higher elevation regions are nested within lower ones, which is equivalent to stating that there are no endorheic regions:

$$h_i > h_j \Rightarrow \Gamma_i \cap \Omega_j = \varnothing$$

An endorheic region (Figure 4) refers to a closed drainage basin where water does not flow out to external water bodies such as seas or oceans. This assumption is motivated by the fact that we want to obtain a hydrologically consistent terrain such as the ones produced by stream power erosion [CBC*16, SPF*23].

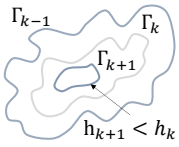

Figure 4: *Unwanted endorheic regions.*

3.2. Reconstruction from contours

Our framework rests on the foundation that the iso-contour and the heightfield models are complementary terrain representations. A digital elevation model can be easily converted to its iso-contour counterpart by using a marching triangle algorithm (see Section 4.4). For visualization, erosion simulation and related tasks, iso-contours can be efficiently converted back to a heightfield.

An efficient method for defining h consist in interpolating the elevations of the two closest iso-contours as proposed in [HSS03]. Let $d(\mathbf{p}, \Gamma_{k+1})$ and $d(\mathbf{p}, \Gamma_k)$ denote the distances from $\mathbf{p}$ to the corresponding curves respectively. Recall that h_k and h_{k+1} denote the heights of Ω_k and Ω_{k+1}, the interpolating height is defined as:

$$h(\mathbf{p}) = \frac{d(\mathbf{p}, \Gamma_{k+1})\, h_{k+1} + d(\mathbf{p}, \Gamma_k)\, h_k}{d(\mathbf{p}, \Gamma_{k+1}) + d(\mathbf{p}, \Gamma_k)}$$

We also implemented a faster and straightforward method to create a stair-step heightfield (Figure 22). The function h is simply defined as the elevation of the closest contour that contains it, formally:

$$h(\mathbf{p}) = h_k \qquad k = \underset{i \,|\, \mathbf{p} \in \Omega_i}{\arg\min}\, d(\mathbf{p}, \Gamma_i)$$

While the resulting terrain is not continuous, it is sufficient for real-time interactive editing and effective for a visual inspection of the generated contours $\widetilde{\Gamma}$.

3.3. Workflow

Given a sparse input set Γ that define the boundary and internal guiding landforms, along with a user-specified height distribution function $f : \mathbb{R} \rightarrow [0, 1]$ specifying the relative proportion of terrain at each elevation, we generate a dense set of iso-contours $\widetilde{\Gamma}$ that represent the final terrain. In our implementation, the height distribution is provided as a histogram $\mathcal{H}$, but any input that allows sampling of height values can be used. Additionally, the user provides a probability map $\mathcal{P}$ that guides the terrain generation across the spatial domain. Together, Γ, $\mathcal{H}$, and $\mathcal{P}$ define the full set of user inputs for our terrain generation pipeline.

The workflow, exposed in Figure 2, proceeds in four steps. First, we construct a graph $\mathcal{G}$ that stores the terrain elevation values, based on the sparse contours set Γ and on the height histogram $\mathcal{H}$. The graph construction is inspired by Eden Growth-based techniques [Ede61, WLB95] and is controlled by the probability map $\mathcal{P}$ (Section 4). From this graph, we extract a set of dense iso-contours

$\widetilde{\Gamma}$ using a marching triangle algorithm, which form the basis of our terrain model.

The vector-based representation can be edited (Section 5) with authoring tools such as warping and amplification. Because the representation is resolution-independent, it allows seamless editing of disparate landforms from procedural, simulated, and real-world sources. Moreover, the contour representation can be efficiently converted back into a traditional digital elevation model by a fast reconstruction process. This ensures compatibility with existing terrain modeling tools, such as erosion simulation.

4. Generation

The generation method simulates a growth process inspired by the Eden algorithm [Ede61, WLB95] (Section 4.1), but instead it progresses inward on a Triangulated Irregular Network $\mathcal{G}$ that covers the region Ω drawn by the user (Section 4.2). Once the growth is complete, each node in $\mathcal{G}$ is assigned a height from the input distribution $\mathcal{H}$, from lower to higher based on the order in which the nodes were visited during the growth process (Section 4.3). Finally, the dense set of generated iso-contours $\widetilde{\Gamma}$ is extracted from $\mathcal{G}$ (Section 4.4). The designer can control the growth order using a probability map $\mathcal{P}$ or by outlining multiple regions, which requires a more advanced generation process (Section 4.5).

4.1. Open Eden Growth

The Eden algorithm [Ede61] was originally introduced to simulate the growth of a cluster C on a regular grid. The process starts with a single seed particle s forming the initial cluster. At each iteration, a new particle $\mathbf{p}$ is added to the cluster as follows. A boundary particle $\mathbf{b} \in B$ of the current boundary of the cluster is randomly selected. A direction $\mathbf{d}$ is then chosen randomly among the four cardinal directions allowed by the grid, and the new particle is placed at $\mathbf{p} = \mathbf{b} + \mathbf{d}$. If all neighboring grid cells of $\mathbf{b}$ are occupied, $\mathbf{b}$ is removed from the boundary B.

In the traditional Eden algorithm, a particle is selected uniformly at random from the current boundary. The choice of the particle can also be non-uniform, *e.g.* driven by a probability map $\mathcal{P}$. This approach is often referred to as an *Open* Eden Growth.

However, since this method is restricted to a grid, the resulting shapes tend to exhibit unnatural axis-aligned structures as seen in Figure 5. To address this, [WLB95] introduced an off-lattice variant of the Eden algorithm. In this version, after selecting a boundary particle $\mathbf{b} \in B$, a direction $\mathbf{d}$ is sampled uniformly at random over the unit circle. A new particle $\mathbf{p} = \mathbf{b} + \mathbf{d} \cdot r$ is tentatively placed at this location, where r is the desired minimum spacing. If the new position does not collide with existing particles, it is accepted; otherwise, the process is repeated several times; if no valid position can be found after several attempts, the particle $\mathbf{b}$ is removed from the boundary.

While the off-lattice approach allows for more organic and anisotropic growth patterns, it is limited by a computationally intensive collision detection step, which often requires the use of dynamic accompanying data structures to accelerate the spatial queries.

We propose an accelerated variant of the Eden Growth algorithm that retains artifact-free properties of the off-lattice version while being significantly faster to compute. This is achieved by relying on a randomly generated graph $\mathcal{G}$. This graph is a precomputed Triangulated Irregular Network from a set of randomly sampled locations. The neighborhood structure of $\mathcal{G}$ eliminates the need for collision detection, and its inherent randomness produces a more chaotic, natural appearance.

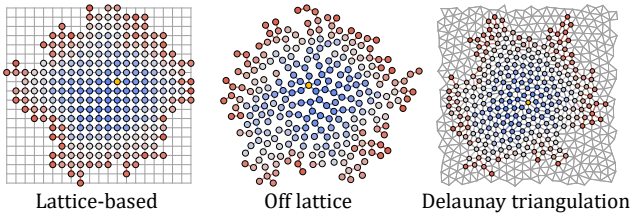

Figure 5: *Variants of the Eden Growth process with $\#\mathcal{G} = 250$ particles over a regular grid, off-lattice growth and our approach using a Triangulated Irregular Network.*

Figure 5 shows a comparison of different Eden Growth variants. The left cluster illustrates a regular lattice-based growth as described in [Ede61], which can be computed efficiently but produces aliasing artifacts due to the underlying regular grid structure. In contrast, the cluster (center) produced by an off-lattice version of the algorithm [WLB95] avoids aliasing but is computationally intensive. The right cluster illustrates our solution: by introducing an underlying graph $\mathcal{G}$ to accelerate the growth process, we achieve the speed of lattice-based approaches with the natural appearance of off-lattice methods.

4.2. Inward Eden Growth

We first describe the inward Eden Growth process for the case of a single user-defined region, *i.e.*, $|\Gamma| = 1$. The extension to multiple regions is discussed in Section 4.5.

To generate iso-contours within a user-defined region Ω, we simulate a height growth process that starts from the boundary of Ω and gradually expands inward. Conceptually, this can be seen as a kind of invasion process: like rising sea levels flooding land, where each successive step corresponds to an increase in elevation.

Contrary to classical Eden Growth, the growth process is performed inward until the prescribed region Ω has been filled. The growth process determines an ordering of the $\#\mathcal{G}$ particles in $\mathcal{G}$ (Algorithm 1). We first generate a random graph $\mathcal{G}$ fully covering Ω. We then apply our modified Open Eden Growth algorithm on $\mathcal{G}$. Unlike traditional growth models that start from a single seed, we designate all particles along $\partial\mathcal{G}$, the boundary of $\mathcal{G}$, as initial seeds. From these, the growth proceeds inward until every node in $\mathcal{G}$ has been selected. At each step, the newly chosen particle $\mathbf{p}$ is assigned the current index i, ensuring that by the end every particle receives a unique value between 1 and $\#\mathcal{G}$. This index implicitly represents the particle's elevation: smaller indices correspond to lower elevations, while larger indices correspond to higher ones. The actual height values, however, are determined by the user-specified distribution and are only applied during iso-contours extraction (Section 4.4).

Algorithm 1: Inward Off-Lattice Open Eden Growth

Input: Region Ω , probability map $\mathcal{P}$
Output: Set of particles $\mathcal{G}$ with assigned indexes n from 1 to $\#\mathcal{G}$

1 Create $\mathcal{G}$ from Ω and initialize $B \leftarrow \partial\mathcal{G}$
2 Initialize current index: $i \leftarrow 1$
3 **for** $\mathbf{p} \in B$ **do**
4 $\quad n(\mathbf{p}) \leftarrow i$; $i \leftarrow i+1$
5 **end**
6 **while** $B \neq \varnothing$ **do**
7 $\quad$ Select particle $\mathbf{b} \in B$ according to $\mathcal{P}$
8 $\quad$ Select particle $\mathbf{p} \in \mathcal{G}$ in the neighbourhood of $\mathbf{b}$ with no assigned index
9 $\quad$ **if** $\mathbf{p}$ *exists* **then**
10 $\quad\quad$ Assigned index to $\mathbf{p}$: $n(\mathbf{p}) \leftarrow i$; $i \leftarrow i+1$
11 $\quad\quad$ Add $\mathbf{p}$ to boundary list: $B \leftarrow B \cup \{\mathbf{p}\}$
12 $\quad$ **end**
13 $\quad$ **if** *no valid* $\mathbf{p}$ *around* $\mathbf{b}$ **then**
14 $\quad\quad$ Remove $\mathbf{b}$ from boundary: $B \leftarrow B \setminus \{\mathbf{b}\}$
15 $\quad$ **end**
16 **end**
17 **return** $(\mathcal{G}, n)$

One major benefit of this method is that it avoids the formation of endorheic regions. Each particle is selected from the neighborhood of an already-processed particle which, by construction, has the same or a lower index. Therefore, every region in Ω remains connected in terms of elevation flow.

Selecting boundary particles with a uniform probability distribution produces orderings that lead to extracted iso-contours resembling the distance field to the border of Ω, an effect that becomes more visible the higher the node density is in $\mathcal{G}$. To overcome this, we select boundary particles according to the input probability map $\mathcal{P}$. For example, by using a fractional Brownian motion function, we can simulate a more organic selection pattern. As a consequence, certain nodes remain in the queue before selection, introducing natural variability in the growth while preserving the absence of endorheic regions.

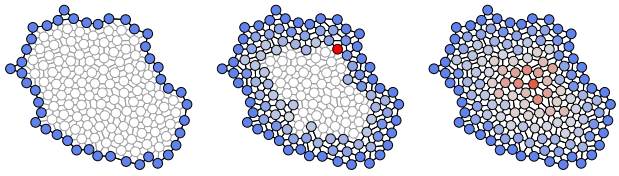

Figure 6: *Key steps in the growth process. Boundary particles B are highlighted with a thick black border, the most recently selected node is shown in red.*

Figure 6 illustrates the key steps of the growth process. The initial graph $\mathcal{G}$, with $\#\mathcal{G} = 155$ particles shows the index assignments after line 5 of Algorithm 1. The second graph depicts an intermediate step of the inward Eden Growth, and the last one the final ordering. The creation of the graph $\mathcal{G}$ and its boundary $\partial\mathcal{G}$ (line 1) are detailed in the following section.

4.3. Graph creation

To speed-up the growth process and avoid checking for collisions or neighborhood queries, we pre-compute a connected graph $\mathcal{G}$ within the user-defined region Ω. First, we perform a Poisson disc sampling of Ω to obtain a uniform and evenly spaced point distribution. We then perform a Delaunay triangulation, resulting in a Triangulated Irregular Network that covers Ω.

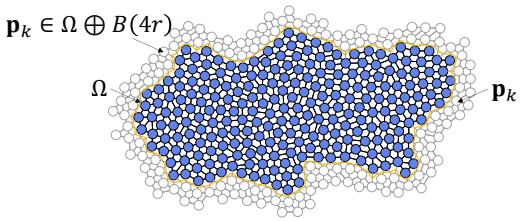

Figure 7: *The graph $\mathcal{G}$ is computed by performing a Poisson disc sampling inside $\Omega \oplus \mathcal{B}(4r)$, applying a Delaunay triangulation and keeping only the triangles within Ω.*

A common issue when applying Delaunay triangulation directly within Ω is the formation of poorly shaped triangles near the boundaries. Instead of using a constrained triangulation or add Steiner points, we opted for a simpler approach that facilitates the computation of $\partial\mathcal{G}$, and allows the extraction of iso-contours (see Section 4.4). We perform sampling and triangulation over an extended domain $\Omega \oplus \mathcal{B}(4r)$ where $\mathcal{B}(4r)$ is the disc of radius $4r$. After computing the triangulation over this extended domain, we retain only the nodes and triangles that lie within Ω (Figure 7).

This method not only improves triangle quality near the boundaries, but also allows us to clearly define the boundary of the network, denoted $\partial\mathcal{G}$, needed in Algorithm 1 in line 1. Specifically, $\partial\mathcal{G}$ consists of all nodes inside Ω that are adjacent in the Delaunay triangulation to at least one node outside Ω (*i.e.*, a node that was discarded during the filtering step).

4.4. Iso-contours extraction

To extract iso-contours, each particle must be assigned a height. However, the growth process described in Section 4.2 produces only indices ranging from 1 to $\#\mathcal{G}$. To convert these indices into heights, a set of $\#\mathcal{G}$ values is sampled from a user-provided distribution, sorted, and then mapped to the particles according to their indices. This procedure guarantees a monotonically increasing assignment, thereby preventing the creation of endorheic regions. Importantly, the user can modify the height distribution without recomputing particle indices, making it easy to explore and adjust the overall terrain shape (Figure 8). For the remainder of this work, we assume that height values are sampled from a user-specified histogram $\mathcal{H}$.

After assigning heights, we extract iso-contours using a marching-triangles algorithm, which produces polygonal curves at constant elevation.

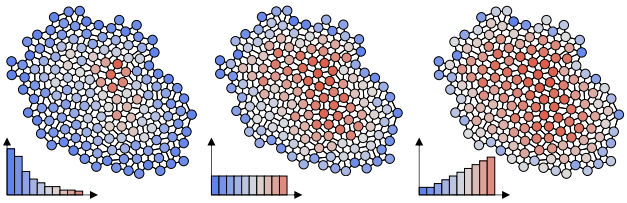

Figure 8: *Example of height assignments produced from three distinct user-defined distributions.*

Given an isolevel height l, we process every triangle of $\mathcal{G}$ as follows. Let **abc** denote a triangle and its corresponding heights $h(\mathbf{a})$, $h(\mathbf{b})$, $h(\mathbf{c})$. It can be classified into one of eight configurations, depending on the sign of $h-l$. These eight possible configurations resort to only two after rotations and symmetries as illustrated in Figure 9. For each straddling edge **ab**, we compute the intersection point at height l by bisection. The two intersection points found form a segment, and we obtain the set of polygonal iso-contours by connecting the segments together.

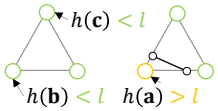

Figure 9: *Configurations for a triangle* **abc**.

The marching triangles method cannot extract iso-contours at heights lower than those assigned to the particles on $\partial\mathcal{G}$. To address this, we keep track of the particles generated outside $\mathcal{G}$ during its construction (shown as gray particles in Figure 7) and assign them the height $h_0-\varepsilon$, where h_0 is the smallest height assigned within $\mathcal{G}$. This ensures that marching triangles will extract a boundary iso-contour at level h_0, effectively enclosing all particles in the lowest region.

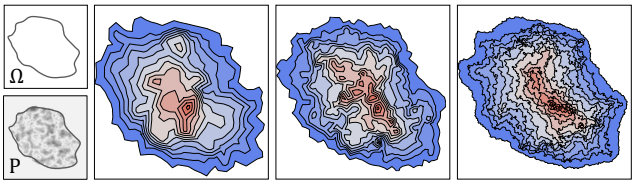

Figure 10: *Extracted iso-contours using different values of r for the Poisson disc sampling: from left to right* $\#\mathcal{G}=130$, $\#\mathcal{G}=2043$ *and* $\#\mathcal{G}=9877$.

Figure 10 shows some examples of iso-contours extracted at various sampling densities. Varying r changes the sampling density, which in turn influences the shape of the resulting iso-contours.

4.5. Generation using multiple regions

Users can provide their own probability map $\mathcal{P}$ to control the generation, for example for emphasizing riverbeds or crest lines, by assigning higher or lower probabilities to specific regions, as illustrated in Figure 11.

However, fine-tuning the probability maps towards a desired outcome can be challenging. To address this limitation and offer designers with better creative control, we extend our method to support multiple user-defined regions. We now assume the user provides a collection of m regions, denoted $\{\Omega_i\}_{i\leq m}=\{\Omega_1,\ldots,\Omega_m\}$. Without loss of generality, we consider Ω_i to represent a region of higher elevation than Ω_{i-1}.

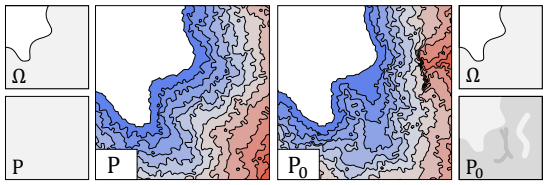

Figure 11: *Comparison of iso-contours generated with two different probability maps $\mathcal{P}$. The left image was generated with a uniform probability, whereas the right image shows the impact of the user-defined probability map $\mathcal{P}$: the high probability regions create valleys, whereas the low probability bar limits inward growth and produces cliffs.*

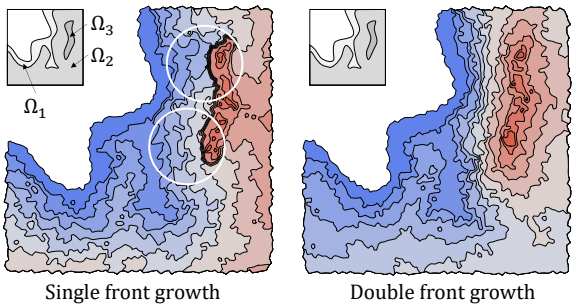

Figure 12: *Comparison of generated iso-contours using standard Eden Growth and two competing fronts.*

One might consider applying Algorithm 1 iteratively to each region Ω_i or, equivalently, assigning a lower probability to nodes in higher regions. This straightforward strategy often yields unsatisfactory results, particularly for elongated or irregularly shaped, which are common in practical scenarios. An example of such limitations is presented in Figure 12 (left).

Instead, we propose a more coherent strategy that treats the relationship between regions hierarchically. The underlying concept is to grow from several contours simultaneously. This results in smoother transitions and avoids abrupt height discontinuities. We create a global graph $\mathcal{G}$ that covers all regions, then partition it into m subgraphs $\mathcal{G}_i$, one for each region: $\mathcal{G}_i=\{\mathbf{p}\in\mathcal{G}\mid\mathbf{p}\in\Omega_i\}$.

When processing a region Ω_i, we assume it is bounded by a lower region Ω_{i-1} and an upper region Ω_{i+1} (special cases are discussed below). Rather than applying one unique growth process starting from $\partial\mathcal{G}_i$ as in Section 4.2, we apply two growth processes, one starting from $\partial\mathcal{G}_i^-$, the other from $\partial\mathcal{G}_i^+$. The boundary $\partial\mathcal{G}_i^-$ consists of particles in $\mathcal{G}_{i-1}$ adjacent to $\mathcal{G}_i$. Similarly, the boundary $\partial\mathcal{G}_i^+$ consists of particles in $\mathcal{G}_{i+1}$ adjacent to $\mathcal{G}_i$. An example is provided in Figure 13.

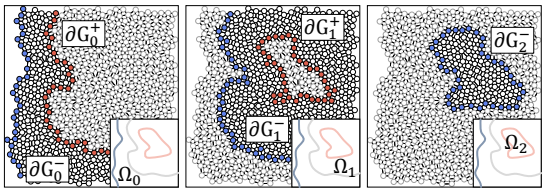

Figure 13: *Lower and upper boundaries for three user-defined regions. The lower boundaries $\partial\mathcal{G}_i^-$ are shown in blue, and the upper boundaries $\partial\mathcal{G}_i^+$ are shown in red.*

We then perform two Open Eden Growth processes within $\mathcal{G}_i$, one growing inward from $\partial\mathcal{G}_i^-$ and the other from $\partial\mathcal{G}_i^+$.

This results in two orderings for each particle $\mathbf{p} \in \mathcal{G}_i$: the step $n^-(\mathbf{p})$ at which it is reached from the outer boundary, and the step $n^+(\mathbf{p})$ at which it is reached from the inner boundary. These are then combined into a single score:

$$n(\mathbf{p}) = \frac{n^-(\mathbf{p})}{n^-(\mathbf{p}) + n^+(\mathbf{p})}$$

We could use $n(\mathbf{p})$ to sort the particles in $\mathcal{G}_i$ and assign heights. However, it does not guarantee that at least one of its neighbouring particles $\mathbf{p}'$ has a lower $n(\mathbf{p}')$. As a result, the algorithm could produce unwanted endorheic regions. To prevent this, we now perform an additional growth pass, this time starting from $\partial\mathcal{G}_i^-$ and progressing always toward the boundary particle with the lowest value $n(\mathbf{p})$. By construction, this ensures that all particles have at least a lower neighbor. The following cases deserve special consideration.

- If Ω_{i+1} does not exist (*i.e.* when $i = m$), we fall back to the standard inward Eden Growth, as this is equivalent to a single-region case (for example, see Ω_2 in Figure 13).
- If Ω_{i-1} does not exist (*i.e.* when $i = 1$), we define $\partial\mathcal{G}_1^-$ as the external boundary $\partial\mathcal{G}_1$.
- If a sub-region of Ω_i defines a local maximum (see Figure 15), then not all particles in $\mathcal{G}_i$ are reachable from $\partial\mathcal{G}_i^+$. For such particle $\mathbf{p}$ we define:

$$n(\mathbf{p}) = \frac{n^-(\mathbf{p})}{\#\mathcal{G}_i}$$

- Conversely, if a sub-region of Ω_i defines a local minimum (which is equivalent to an endorheic region), then not all particles in $\mathcal{G}_i$ are reachable from $\partial\mathcal{G}_i^-$, therefore:

$$n(\mathbf{p}) = \frac{\#\mathcal{G}_i - n^+(\mathbf{p})}{\#\mathcal{G}_i}$$

Although our algorithm is designed to prevent such cases in order to obtain a hydrologically consistent terrain [LGP*23], a designer may still intentionally define one. Therefore, our framework explicitly handles this situation.

Figure 12 illustrates the benefits of the improved method with a mask specifying a narrow valley and a high mountain range. The left image shows the result when applying the standard Eden Growth algorithm (Algorithm 1) iteratively to each region: a large and Énnatural Éliff Érea Éppears Éetween Ω_2 Énd Ω_3, És Ω_2 grows

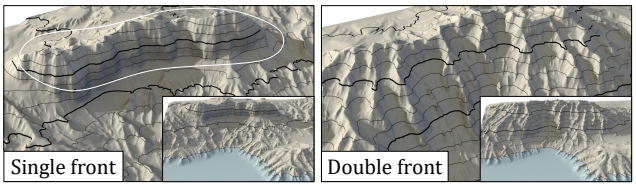

Figure 14: *3D view of the terrains in Figure 12.*

without considering the influence of Ω_3, resulting in a discontinuity in the height transition. In contrast, when using the double-front improved algorithm, the iso-contours successfully follow the river path, and the crest is clearly delineated. Figure 14 presents the 3D view of the result illustrating the difference in front growth.

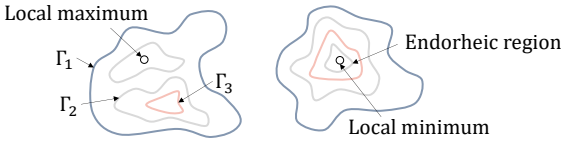

Figure 15: *Example of user-defined regions containing a local maximum (left) and a local minimum (right). Note that a local minimum corresponds to an endorheic region.*

5. Editing

We introduce a set of tools for editing a collection of iso-contours. They provide designers with intuitive and flexible methods to modify terrain via contour manipulation, thereby keeping the editing process both accessible and artistically expressive. Each tool is formalized as a transformation function acting on the set of generated iso-contours $\widetilde{\Gamma}$ to produce a new configuration.

5.1. Smoothing iso-contours

Generated iso-contours often exhibit sharp angles, mainly due to the discretisation of Ω into $\mathcal{G}$ and the marching triangles step to generate the polygonal shapes. To address this, we provide a classic smoothing tool that reduces angular artifacts while preserving the overall shape of the terrain (Figure 16).

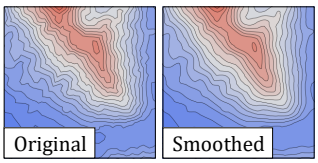

Figure 16: *Smoothing a set of iso-contours.*

The method operates by iteratively moving each vertex of an iso-contour towards the average position of its neighbouring vertices. Let $\mathbf{q}_i$ be a vertex of an iso-contour polygon, the update rule is:

$$\mathbf{q}_i \leftarrow (1-\alpha)\mathbf{q}_i + \frac{\alpha}{2k}\left(\sum_{j=1}^{k}\mathbf{q}_{i-j} + \sum_{j=1}^{k}\mathbf{q}_{i+j}\right)$$

The parameter α controls the smoothing strength (larger α produces faster smoothing), and $k \in \mathbb{N}$ defines the neighborhood size, allowing either more local or more global smoothing effects. By adjusting α and k, users can balance between preserving fine details and achieving a more globally smoothed contour. Note that k can also be obtained given a radius of influence of the tool.

5.2. Coherent iso-contour warping

This first tool allows designers to stretch or warp iso-contours along a user-defined path, while preserving their topological nesting (i.e., inclusion) and create terrain features such as ridges, valleys, rifts, or riverbeds. The designer simply draws a curve from one point to another, which then serves as the axis of deformation. The iso-contours are adjusted accordingly to follow the path in a natural and coherent manner.

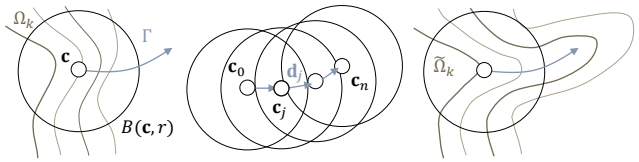

Figure 17: *Warping deforms space so that the iso-contours smoothly follow the user-prescribed trajectory* Γ.

Figure 17 illustrates how the iso-contours stretch along the user-defined trajectory of the warping tool. Let Γ be a curve drawn by the user. The goal is to construct a continuous and bijective (to avoid contours to cross after deformation) function $\omega : \mathbb{R}^2 \to \mathbb{R}^2$ that warps the space around Γ, allowing the iso-contours to follow its shape.

We divide Γ into n segments of length ε, and choose a deformation radius $r > \varepsilon$. Let $\mathbf{v}_i$ denote the $n+1$ vertices of Γ, and $\mathbf{d}_i = \mathbf{v}_{i+1} - \mathbf{v}_i / ||\mathbf{v}_{i+1} - \mathbf{v}_i||$ be the unit direction vector of the i-th segment. We define ω as the composition of n bijective functions f_i, where each f_i represents a local displacement confined to the ball $\mathcal{B}(\mathbf{v}_i, r)$ in the direction $\mathbf{d}_i$:

$$\omega(\mathbf{p}) = f_1(\mathbf{p}) \circ \cdots \circ f_n(\mathbf{p})$$

Let $\mathbf{p} \in \mathcal{B}(\mathbf{v}_i, r)$, we define two intersection points $\mathbf{a}$ and $\mathbf{b}$ such that the line $\ell_{\mathbf{p}}(t) = \mathbf{p} + t\mathbf{d}_i$ intersects the boundary of $\mathcal{B}(\mathbf{v}_i, r)$ at $\mathbf{a}$ and $\mathbf{b}$ respectively (Figure 18). The normalized interpolation parameter $u = ||\mathbf{p} - \mathbf{a}|| / ||\mathbf{b} - \mathbf{a}|| \in [0,1]$ represents the position of $\mathbf{p}$ along the segment $[\mathbf{a}, \mathbf{b}]$. The functions f_i are defined as:

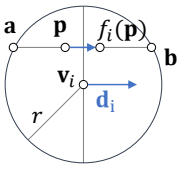

Figure 18: *Notations for warping.*

$$f_i(\mathbf{p}) = \begin{cases} \mathbf{p} + \mu\delta(u)(\mathbf{b} - \mathbf{a}) & \text{if } \mathbf{p} \in \mathcal{B}(\mathbf{v}_i, r) \\ \mathbf{p} & \text{otherwise} \end{cases}$$

$\delta : [0,1] \to [0,1]$ determines the displacement factor and $\mu \in [0,1]$ is a control parameter representing the *strength* of the warping effect. The intuition is that each point within $\mathcal{B}(\mathbf{v}_i, r)$ is shifted along the direction $\mathbf{d}_i$, with the magnitude of the shift defined by $\delta(u)$ and the user-provided constant μ. Appendix A provides information on the properties that δ must satisfy, and particularly the boundary conditions $\delta(0) = \delta(1) = 0$. We propose the following formulation for $\delta(x)$, with $k \in \mathbb{N}^*$:

$$\delta(x) = \frac{\varepsilon(4x(1-x))^k}{2r}$$

Our function satisfies the requirements for continuity and bijectivity of ω. Indeed, $\delta(0) = \delta(1) = 0$ and, for a given user-defined radius r, bijectivity holds if the segment length is:

$$\varepsilon \leq \frac{2r\sqrt{2k-1}(4k-2)^{k-1}}{4^k k(k-1)^{k-1}}$$

As examples, for $k = 1$, $\varepsilon \leq 0.5r$; for $k = 2$, $\varepsilon \leq 0.64r$; and for $k = 3$, $\varepsilon \leq 0.58r$. The derivation of this bound is given in Appendix B. δ also exhibits properties that make it well-suited for deformation, notably, the maximum displacement occurs at $\mathbf{v}_i$, where $\delta(0.5) = \varepsilon/(2r)$ (seeAppendix A). If the warping strength is set to maximum ($\mu = 1$), we obtain $f_i(\mathbf{v}_i) = \mathbf{v}_i + \varepsilon\mathbf{d}_i = \mathbf{v}_{i+1}$. Figure 19 presents a deformation example.

While we use this specific formulation, any function that satisfies the required constraints can be used. Appendix B also describes how to construct alternative δ functions with tailored properties.

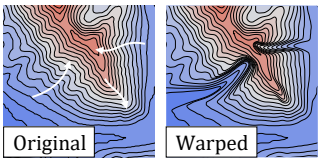

Figure 19: *Deformation of a set of iso-contours using three warps* Γ*: using a sequence bijective transformations guarantees that the deformed iso-contours do not self-intersect.*

Although the function ω is defined to warp the entire space $\mathbb{R}^2$, in practice we only need to deform the set of iso-contours $\tilde{\Gamma}$. Since the iso-contours generated in Section 4 are represented as sets of polygons, we can apply ω directly to the vertices of these polygons. However, if the polygonal sampling is too sparse within the disc $\mathcal{B}(\mathbf{v}_i, r)$ – meaning that the distance between consecutive vertices is large – the deformation may become inaccurate or visually unsmooth.

Moreover, in Appendix C, we show that in order to avoid crossings between iso-contours, we would require prior knowledge about the horizontal spacing between them. However, in our tests, we have not observed artifacts using a sampling distance of 0.1ε.

5.3. Slope modulation

A set of iso-contours provides a quick visual cue about terrain slope: closely spaced contour lines indicate a steep slope, while widely spaced ones correspond to gentler slopes. To give users direct control over slope, we introduce a modulation tool that adjusts the spacing between iso-contours (Figure 21).

The approach is conceptually similar to the warping tool described in the previous section, in that it deforms a local region

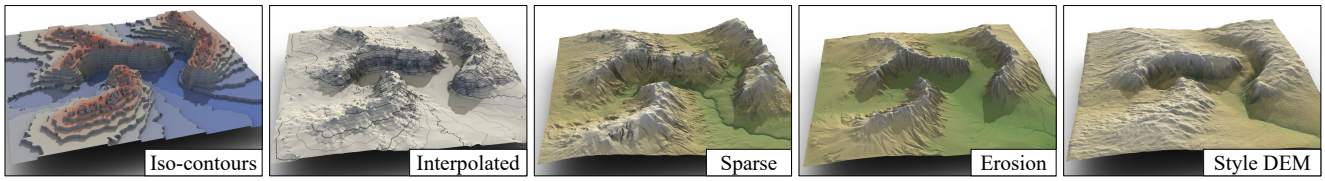

Figure 20: *The iso-contours are shown on the left, followed by the resulting interpolation reconstruction. On the right, several amplification results are shown based on the interpolation, using (from left to right): Sparse amplification, Multi-Scale Erosion, and StyleDEM.*

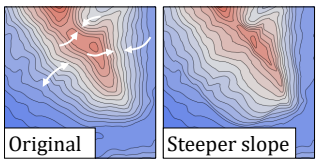

Figure 21: *Modulation of slope using two control curves. In the second image, the left slope is reduced, while the slope on the right one is increased.*

specified by the user in a radial manner rather than rectilinear. Let $\mathbf{a}$ denote the closest point to $\mathbf{p}$ on Γ and set $\mathbf{b}$ as a point at distance r from $\mathbf{a}$ in the direction $\mathbf{p}-\mathbf{a}$. We define $u = ||\mathbf{p}-\mathbf{a}||/r$ and $\delta(x) = x(1-x)$, so that we arrive at the following expression for the radial stretch:

$$\omega_s(\mathbf{p}) = \begin{cases} \mathbf{p} - \mu\delta(u)(\mathbf{p}-\mathbf{a}) & \text{if } ||\mathbf{p}-\mathbf{a}|| \leq r \\ \mathbf{p} & \text{otherwise} \end{cases}$$

If the curve Γ degenerates to a single point, our definition of ω_s satisfies the continuity and bijectivity requirements. For polygonal or continuous curves, it satisfies the continuity but bijectivity is subject to the radius of curvature of Γ being larger than r.

6. Results

We implemented our method in C++, and all the figures presented in this paper were generated using our application. The code is available at https://github.com/Arches-Team/Contours. All experiments were conducted on a machine equipped with an AMD Ryzen 5 5600X 6-core processor and 32 GB of RAM.

6.1. Generation examples

The output of our generation algorithm is a set of iso-contours, from which we can reconstruct a stair-step terrain representation, as shown in the left-most image in Figure 20. This representation provides immediate feedback to the artist. By interpolating between the two closest iso-contours (as described in Section 3.2), we obtain a continuous terrain, shown in the second image of Figure 20. This continuous representation can then serve as input for post-processing with state-of-the-art amplification techniques based on height fields, such as Sparse Terrain Reconstruction [GDGP16], Multi-Scale Erosion [SGG*24] or learning-based methods like the StyleDEM network [PPB*23]. Examples of these results are presented in the last three images of Figure 20. Therefore, our method complements most existing approaches in terrain modeling. Unless otherwise specified, all rendered terrains shown in the figures include an erosion post-processing step.

The expressiveness of our approach makes it possible to generate a wide variety of terrains from simple inputs. Figure 22 demonstrates results such as volcanoes, canyons, plateaus, and mountain ranges. A few sparse iso-contours are sufficient to produce coherent terrains and large-scale landforms.

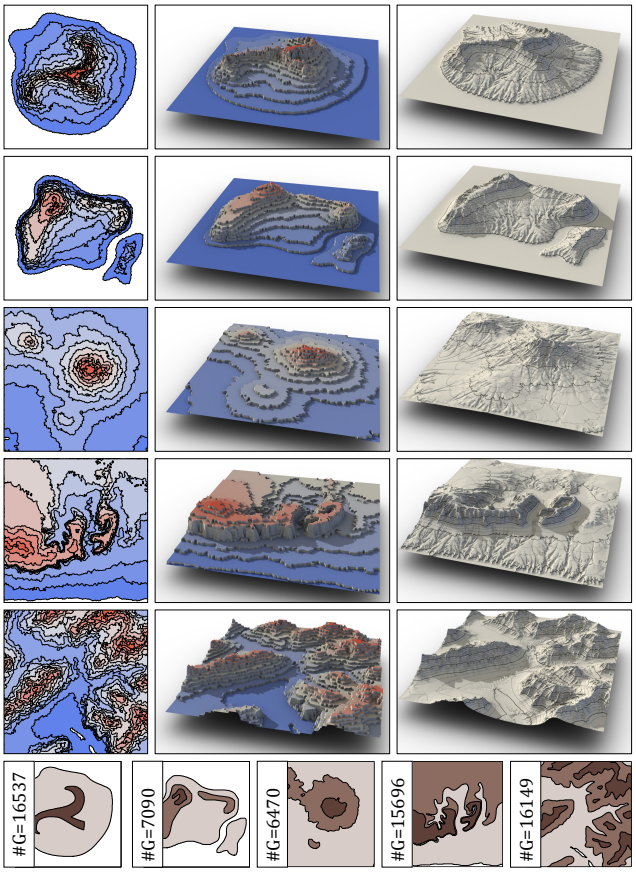

Figure 22: *Varied terrains generated by our method; the last row reports the input maps and number of particles #$\mathcal{G}$*

Further control over terrain generation is driven by three main parameters: the sparse user-defined iso-contours Γ, the target histogram $\mathcal{H}$, and the probability map $\mathcal{P}$. Figure 23 illustrates the influence of the histogram $\mathcal{H}$ and the probability map $\mathcal{P}$ on the

obtained terrain. Adjusting the histogram modifies elevations and slopes, while the overall drainage network is preserved. In contrast, modifying the probability map changes the ordering of the particles, leading to different terrain structures.

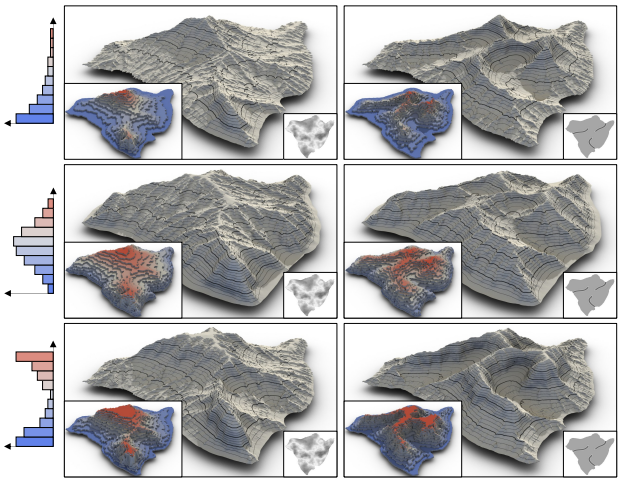

Figure 23: *Examples with a single user-defined control region Γ_0 and different probability maps $\mathcal{P}$: fractal Brownian motion, and a user-drawn map. Three different histograms were used. For each configuration, we show the generated iso-contours corresponding to a generation using $\#\mathcal{G} = 8812$ particles.*

Figure 24 demonstrates that input iso-contours need not be highly detailed or numerous to produce realistic terrains. The bottom row shows a terrain generated from iso-contours extracted from a real elevation map of the Western Alps. The top row, on the other hand, uses a simple hand-drawn sketch that approximates the shape of the mountain range. Remarkably, even with a rough sketch, the resulting terrain appears realistic and richly detailed.

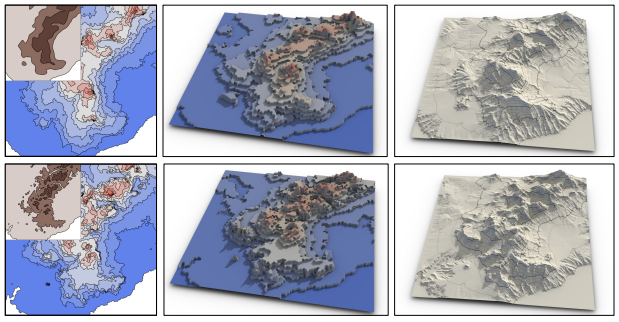

Figure 24: *Comparison of obtained terrain from sketched contours (top) and real contours (bottom).*

Finally, we present an example of an interactive editing session in Figure 27. The artist starts with the low-resolution outline of a coastline (image 1), which generates a detailed shoreline and increasing elevations inland. Two inner regions are then added (image 2) to produce mountain ranges. As expected from the growth process, without additional constraints the bottom-left range reaches higher elevations since it spans a larger area. The artist corrects this simply by sketching a higher contour inside the right-hand range (image 3). At this stage, the left range is also modified with an endorheic basin, forming an enclosed depression resembling a crater. Next, the terrain is altered by adjusting the histogram distribution (image 4), reducing the frequency of lower elevations so that the land near the coastline rises more abruptly. Finally, local refinements are applied (image 5) using the warping tool to add crests in the lower-right region and a small ravine descending from the bottom mountain range. Prior to warping, the iso-contours were smoothed to enhance their readability and facilitate control during editing.

6.2. Validation and expert feedback

We gathered qualitative feedback from a professional terrain artist to assess the usability and expressiveness of our approach. The expert reported that representing terrain through iso-contours is an unconventional yet compelling paradigm for terrain modeling. In particular, this representation was identified as well suited for user-generated content, as it produces coherent terrain structures by construction and reduces the risk of inconsistent or implausible results. They noted that such properties make the approach especially relevant for applications such as god games, real-time strategy games, and city-building environments.

The expert also pointed out inherent limitations of the proposed representation. Terrain features characterized by strong verticality, such as deep canyons or cliffs, are less naturally captured due to the predominantly horizontal structure of the the underlying iso-contour model. Despite this limitation, the continuous and vector-based nature of the representation was highlighted as a key advantage, as it enables efficient scalability to very large terrains and naturally supports multiple levels of detail. This allows artists to visualize, edit, and author terrain features at different spatial scales.

Finally, the expert suggested a promising extension of our approach: exploiting real-world terrain data by extracting iso-contours and using them as a library of reusable patterns. Such patterns could be directly reused or combined by designers within synthetic terrains, an idea that naturally aligns with and complements the proposed representation.

6.3. Performance

Table 1 reports the runtime of our method, where $\#\mathcal{G}$ denotes the number of particles and $t_\mathcal{G}$ the time required for the Eden Growth process. Table 1 also reports the times required to extract 10, 20 and 30 iso-contours, using the marching triangles algorithm. Each experiment was repeated 20 times on different graphs, and we report the mean runtime over these runs.

We observe that runtime grows as $\mathcal{O}(\#\mathcal{G}\sqrt{\#\mathcal{G}})$. The square root term arises from selecting the next particle from the current boundary B, which typically contains $\sqrt{\#\mathcal{G}}$ elements. Although this step could be optimized with a more efficient particle-selection structure, such as a segment tree, allowing an $\mathcal{O}(\log(\#\mathcal{G}))$ selection time, we did not pursue this optimization since using $10-20$k particles only already generated high-quality results. The cost of extracting iso-contours grows linearly with $\#\mathcal{G}$.

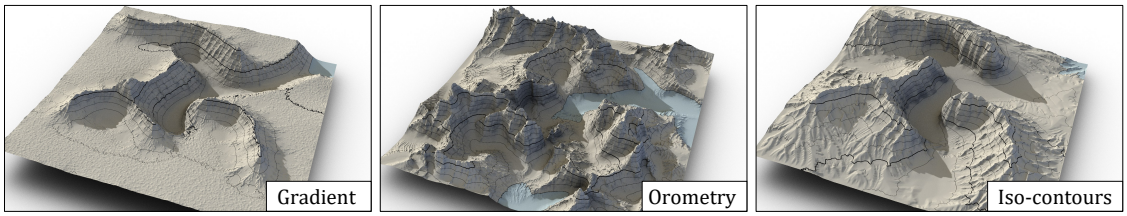

Figure 25: *Comparison between gradient-based modeling, orometry, and our method.*

Particles #$\mathcal{G}$	Time $t_{\mathcal{G}}$	Iso-contours extraction time		
		10	20	30
$\sim$ 10k	86	10	19	29
$\sim$ 20k	297	16	31	46
$\sim$ 50k	1 880	32	64	95
$\sim$ 100k	5 827	71	142	212
$\sim$ 500k	91 790	386	766	1 140

Table 1: *Runtime analysis of our method; all timings are reported in milliseconds (ms).*

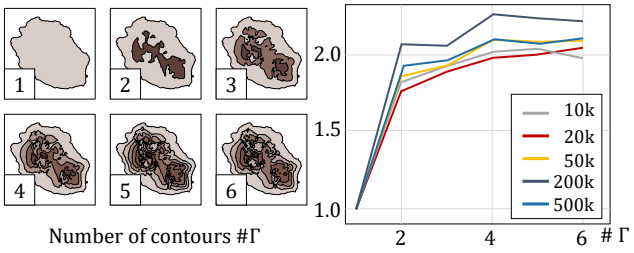

Figure 26: *Relative execution time, normalized by the single-region baseline. Each poly-line shows the slowdown factor as the number of regions increases for a fixed #$\mathcal{G}$.*

Table 1 shows that the runtime increases linearly with the number of extracted iso-contours. This behavior is expected, as each iso-contour requires a single marching triangles extraction.

The timings reported in Table 1 were obtained using a single input region Ω (left input in Figure 26). Increasing the number of regions m does not increase the runtime by more than a factor of two compared to the single-region case (Table 1). This behavior arises because the difference between the single and multi-region configurations lies in the number of Eden Growth runs performed per region (bounded by two in the multi-region process). Figure 26 reports the processing time for up to six regions, measured using the same timing and averaging protocol as in Table 1. We observe an initial increase in execution time when moving from one to two regions, followed by a stable runtime for additional regions.

6.4. Comparison with other methods

Our method bears some similarity to earlier approaches in procedural terrain modeling and editing, particularly those relying on vector representations. Below we compare it to three representative techniques: the growing-strategy terrain generator of Genevaux *et al.* [GGG*13], the orometry-based synthesis of Argudo *et al.* [AGP*19], and the gradient-based authoring framework of Guérin *et al.* [GPM*22].

River networks Genevaux et al. [GGG*13] generate river networks by expanding a river graph within a user-specified iso-contour Γ_0, ensuring hydrological consistency in the resulting terrain. While effective for river formation, their rule-driven growth algorithm makes it difficult to control large-scale elevations and to design specific landforms such as ridges or mesas.

Our approach avoids grammar-based rules and instead accepts a sparse set of user-defined contours Γ, enabling explicit placement of plateaus, valleys, and mountain ranges while still guaranteeing hydrological consistency.

Orometry The orometry-based synthesis of Argudo et al. [AGP*19] generates terrains that match statistical distributions derived from real digital elevation models. It targets mountainous landscapes and incorporates histograms of peak prominence and isolation to produce a graph of connected saddles and peaks over a domain Ω_0. We ran their available code with the same sparse contour set and elevation histogram used in our experiments, retaining their default Pyrenees-based distributions for all other parameters. While the synthesized terrain generally respects the control map (see Figure 25), the final shapes of peaks and saddles remain difficult to predict because of the stochastic procedural placement.

By contrast, our technique allows designers to specify elevation histograms and sparse iso-contours for finer control over the resulting landforms, albeit with a weaker adherence to orometry.

Gradient-based Both diffusion-equation methods [HGA*10] and the gradient-based framework of Guérin *et al.* [GPM*22] reconstruct heightfields from sparse iso-contours used as elevation constraints. These approaches support user-defined ridge lines, river networks, and landform contours, but rely on solving diffusion or Poisson equations, which yield very smooth surfaces and require substantial computation. To warrant a fair comparison with our method (see Figure 25), we defined additional gradient directions around the prescribed elevation constraints for iso-contours to obtain an terrain free of artifacts.

In contrast, our method produces realistic, varied terrains from even sparser inputs, while offering more intuitive, interactive control than these partial differential equation-based techniques.

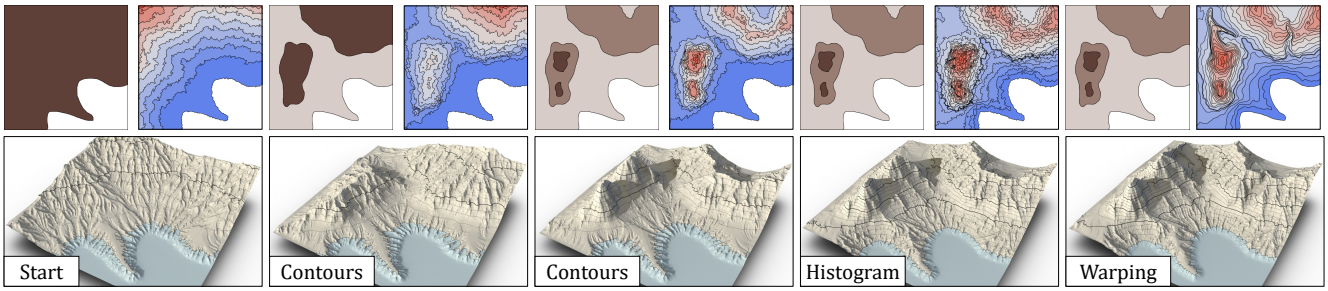

Figure 27: *Steps of an editing session. From left to right: 1) The user first sketches the coastline; 2) Two regions are added to describe two mountain ranges; 3) The left range is raised by adding a nested region, while an endorheic region is introduced in the upper region to represent a crater. 4) Modifying the histogram alters the overall appearance, particularly near the coastline. 5) The iso-contours are smoothed to improve readability and prepare for further edits, after which multiple warpings are applied to add crests and a small ravine.*

6.5. Comparison with real terrains

Although our method is designed for synthetic terrain generation, we perform a quantitative comparison with real-world terrain using standard terrain descriptors [AGSG25]. Figure 28 shows histograms of elevation, slope, and drainage area computed on the two synthetic terrains presented in Figure 24, as well as on a corresponding region of the Alps. The real terrain elevation map is sampled at a resolution comparable to that of our generated terrains.

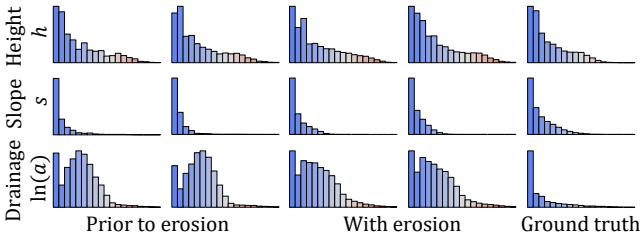

Figure 28: *Descriptors computed on the results shown in Figure 24 and the corresponding real elevation map from the Western Alps.*

As expected, the elevation distributions closely match those of the real terrain, since the target elevation histogram is provided as an explicit input to our method. Slope distributions exhibit a slight shift toward lower values, primarily due to the reconstruction using only 12 iso-contours over a vertical range of 5 000 m. The erosion post-processing partially mitigates this effect by introducing locally steeper features. For drainage area, we observe a more pronounced discrepancy in the distributions. Because a multiple-flow-direction routing algorithm is used, the smooth interpolated reliefs produced by our method promote flow spreading with fewer well-defined channels. In contrast, the real terrain contains numerous small valleys and gullies, resulting in more strongly channeled flow and a higher proportion of low drainage area values. The erosion post-processing recovers part of this fine-scale structure, shifting the distribution toward lower values.

Beyond these descriptors, we also compared two metrics characterizing the structure of surface networks. For the river network, the maximum Strahler order measured on our terrains matches that of the reference terrain, indicating comparable large-scale network organization. However, for the ridge network, our terrains exhibit between two and ten times fewer peaks, depending on the prominence threshold. This discrepancy is mainly due to the preference of our growth algorithm for smooth, continuous gradients, and to the removal of local maxima whose elevations fall between two consecutive iso-contours.

6.6. Limitations and future work

To our knowledge, this is the first work to address terrain generation and editing directly through iso-contours. Our method can produce coherent landscapes from as little as a single iso-contour, but the stochastic nature of the Open Eden Growth algorithm and its probability maps can occasionally yield unexpected results. Selecting an appropriate probability map to achieve a specific design intent may require some experimentation. This limitation is largely alleviated by our multi-region strategy: sketching several nested iso-contours is straightforward and leads to more predictable outcomes.

Generating rivers of varying widths and slopes remains a challenge, as it would require probability maps that change with river characteristics or a large number of user-defined regions. A promising direction is to allow users to provide a hand-drawn river network with explicit riverbed elevations, which the system could then incorporate to produce more naturally shaped river iso-contours.

As future work, we plan to investigate a multi-level generation pipeline that progressively increases geometric detail. An initial pass would use a large Poisson-disc radius r to establish large-scale structures, followed by successive passes with decreasing radii to refine the terrain and enrich the iso-contour hierarchy. Although we did implement standard editing operations such as iso-contour selection, deletion, or cut-and-paste operations, these features should be straightforward to add. A complementary capability would be a filling tool to regenerate the interior of a deleted contour or to blend pasted regions while preserving the surrounding elevation distribution. Such a filling operation could even be powered by a database: the user would sketch or select a region, and the system would retrieve and adapt a matching terrain patch to fit the target area. Additional tools, such as contour fractalization for fine-scale adjustments, could further enhance authoring flexibility. Finally, porting

both generation and editing to the graphics hardware is a natural next step to achieve real-time performance.

7. Conclusion

We presented a new method for procedurally generating iso-contours guided by user-specified curves and a target height distribution. Our approach combines a particle-based growth process with controllable parameters, such as probability maps and Poisson-disc sampling, to give designers intuitive, flexible control over large-scale terrain landforms and structures. We demonstrated results on a variety of scenarios, from coastal islands to mountain ranges, and showed that the generated contours integrate naturally with existing terrain amplification and erosion techniques to produce detailed heightfields.

Our method bridges artistic control, using iso-contours, with procedural terrain generation, nonetheless some limitations suggest directions for future research. While our method is not intended to produce fully realistic terrains on its own, it offers an efficient tool for early-stage terrain design and interactive authoring. Current limitations include handling complex riverbeds and achieving higher iso-contour realism. Therefore, future work will focus on improved river modeling, multiple contour generation, hierarchical refinement for added detail, and enhanced interactivity through real-time feedback and graphics hardware acceleration.

Acknowledgments

This work is funded by the project EOLE ANR-23-CE56-0008, supported by Agence Nationale de la Recherche (France), and Grant PID2021-122136OB-C21 funded by MICIU/AEI/10.13039/501100011033 and ERDF/EU. Open access publication funding provided by COUPERIN CY26.

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*, 6-8 (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). 2, 11

[AGSG25] ARGUDO O., GUÉRIN E., SCHOTT H., GALIN E.: Terrain descriptors for landscape synthesis, analysis and simulation. *Computer Graphics Forum 44*, 2 (2025), e70080. 12

[CBC*16] CORDONNIER G., BRAUN J., CANI M.-P., BENES B., GALIN E., PEYTAVIE A., ÉRIC GUÉRIN: Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum 35*, 2 (2016), 165–175. 3

[CMN98] CHAI J., MIYOSHI T., NAKAMAE E.: Contour interpolation and surface reconstruction of smooth terrain models. In *Proceedings of Visualization (Cat. No. 98CB36276)* (1998), IEEE, pp. 27–33. 2

[CXY*22] CAI X., XI M., YU N., YANG Z., SUN H.: A terrain elevation map generation method based on self-attention mechanism and multifeature sketch. *Computational Intelligence and Neuroscience 2022*, 1 (2022), 9481445. 2

[Ede61] EDEN M.: A two-dimensional growth process. *Dynamics of fractal surfaces 4*, 223–239 (1961), 598. 3, 4

[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

[GDGP16] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A.: Sparse representation of terrains for procedural modeling. *Computer Graphics Forum 35*, 2 (2016), 177–187. 1, 2, 9

[GGG*13] GÉNEVAUX J.-D., GALIN É., GUÉRIN É., PEYTAVIE A., BENEŠ. 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 E., PEYTAVIE A., GUÉRIN E., BRIQUET C., GROSBELLET F., BENES B.: Terrain Modelling from Feature Primitives. *Computer Graphics Forum 34*, 6 (2015), 198–210. 2

[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 38*, 2 (2019), 553–577. 2

[GPM*22] GUERIN E., PEYTAVIE A., MASNOU S., DIGNE J., SAUVAGE B., GAIN J., GALIN E.: Gradient Terrain authoring. *Computer Graphics Forum 44*, 2 (2022), 85–95. 1, 2, 11

[HGA*10] HNAIDI H., GUÉRIN E., AKKOUCHE S., PEYTAVIE A., GALIN E.: Feature based terrain generation using diffusion equation. *Computer Graphics Forum 29*, 7 (2010), 2179–2186. 1, 2, 11

[HSS03] HORMANN K., SPINELLO S., SCHRÖDER P.: C1-continuous terrain reconstruction from sparse contours. In *Proceedings of the Vision, Modeling, and Visualization* (2003), vol. 3, pp. 289–297. 2, 3

[LB25] LIU Y., BENES B.: Single-shot example terrain sketching by graph neural networks. *Computer Graphics Forum 44*, 1 (2025), e15281. 2

[LGP*23] LOCHNER J., GAIN J., PERCHE S., PEYTAVIE A., GALIN E., GUÉRIN E.: Interactive authoring of terrain using diffusion models. *Computer Graphics Forum (Proceedings of Pacific Graphics) 42*, 7 (2023). 1, 2, 7

[PPB*23] PERCHE S., PEYTAVIE A., BENES B., GALIN E., GUÉRIN E.: authoring terrains with spatialised style. *Computer Graphics Forum (Proceedings of Pacific Graphics) 42*, 7 (2023). 1, 2, 9

[SGG*24] SCHOTT H., GALIN E., GUÉRIN E., PEYTAVIE A., PARIS A.: Terrain amplification using multi-scale erosion. *ACM Transactions on Graphics 43*, 4 (2024). 1, 2, 3, 9

[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), 1–15. 3

[TGSC24] TZATHAS P., GAILLETON B., STEER P., CORDONNIER G.: Physically-based analytical erosion for fast terrain generation. *Computer Graphics Forum 43*, 2 (2024), e15033. 2

[WLB95] WANG C. Y., LIU P.-L., BASSINGTHWAIGHTE J.: Off-lattice eden-c cluster growth model. *Journal of physics A: Mathematical and general 28* (1995), 2141–2148. 3, 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). 2

[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. 1, 2

Appendix A: Displacement function properties

We aim to construct a smooth warping function ω, defined as a composition of compactly supported functions f_i:

$$f_i(\mathbf{p}) = \begin{cases} \mathbf{p} + \delta(u)(\mathbf{b}-\mathbf{a}) & \text{if } \mathbf{p} \in \mathcal{B}(\mathbf{v}_i, r) \\ \mathbf{p} & \text{otherwise} \end{cases}$$

where u, $\mathbf{a}$ and $\mathbf{b}$ are defined in Section 5.2.

We want to define δ such that each f_i describes a smooth displacement inside the disc $\mathcal{B}(\mathbf{v}_i, r)$ in the direction $\mathbf{d}_i$. Below, we

summarize the key properties we believe a good displacement function should satisfy.

Boundary conditions Since ω must be continuous and bijective to avoid self-intersecting iso-contours, each f_i must also satisfy these properties. Because f_i is the identity outside $\mathcal{B}(\mathbf{v}_i, r)$, the displacement function δ must be continuous and satisfy the following constraints to guarantee a smooth transition at the boundaries of the local support:

$$\delta(0) = \delta(1) = 0$$

Bijectivity For f_i to be invertible, the derivative of δ must satisfy the constraint:

$$\delta' \geq -1$$

To see why this condition ensures bijectivity, we can rewrite f_i for $\mathbf{p} \in \mathcal{B}(\mathbf{v}_i, r)$ as: $f_i(\mathbf{p}) = \mathbf{a} + (\delta(u) + u)(\mathbf{b} - \mathbf{a})$. From the boundary conditions, we have: $\delta(0) + 0 = 0$ and $\delta(1) + 1 = 1$. For δ to be bijective, this mapping must be strictly monotonic over $[0, 1]$, which is equivalent to $(\delta(x) + x)' \geq 0$, therefore $\delta' \geq -1$.

Positivity and maximum at center To ensure that the deformation always moves points forward along the direction $\mathbf{d}_i$, we require: $\forall x \in [0, 1], \delta(x) \geq 0$.

Intuitively, the largest displacement should occur at the center of the support, corresponding to the vertex $\mathbf{v}_i$ on the user-drawn curve. Therefore, we impose: $\text{argmax}(\delta) = 0.5$.

Center to center When the deformation strength is set to its maximum, the center of the support should be displaced exactly from $\mathbf{v}_i$ to $\mathbf{v}_{i+1}$. This implies:

$$f_i(\mathbf{v}_i) = \mathbf{v}_{i+1}$$

This property ensures that consecutive centers along the user-drawn path map directly to one another, preserving the intended trajectory of the deformation.

Derivability For the deformation to look smooth and natural, ω should be at least twice continuously differentiable. This is guaranteed if δ is $\mathcal{C}^2$ and:

$$\delta'(0) = \delta'(1) = 0$$

Appendix B: Displacement function creation

While we have provided a specific choice for δ, any continuous function $\delta_0 : [0, 1] \to [0, 1]$ can be adapted to meet the requirements of a valid displacement function, provided it satisfies the following boundary conditions: $\delta_0(0) = \delta_0(1) = 0$.

Matching the center displacement requirement. To produce the desired deformation strength, we require:

$$f_i(\mathbf{v}_i) = \mathbf{v}_{i+1} = \mathbf{v}_i + \varepsilon \mathbf{d}_i \tag{1}$$

where ε is the distance between two consecutive samples along Γ (the centers of the local supports), r is the radius of the compact support, and $\mathbf{d}_i$ is the direction of the i^{th} segment.

At $\mathbf{p} = \mathbf{v}_i$, we have $u = 0.5$ and $\mathbf{b} - \mathbf{a} = 2r\mathbf{d}_i$. Substituting into the definition of f_i given in Section 5.2, yields:

$$f_i(\mathbf{v}_i) = \mathbf{v}_i + \delta(0.5)\, 2r\mathbf{d}_i$$

To match Equation 1, we require:

$$\delta(0.5) = \frac{\varepsilon}{2r}$$

We can satisfy this requirement by scaling δ_0 by a constant factor:

$$\delta(x) = \frac{\varepsilon}{2r \cdot \delta_0(0.5)} \delta_0(x)$$

This preserves the shape of the original displacement profile while adjusting its magnitude to match the desired center value $\delta(0.5) = \varepsilon/(2r)$.

Bijectivity constraint. Scaling does not affect the boundary conditions, but it can compromise the invertibility of f_i if not done carefully. As we have shown, bijectivity requires: $\delta' \geq -1$. Substituting our scaled definition of δ yields:

$$\frac{\varepsilon}{2r \cdot \delta_0(0.5)} \delta_0{}' \geq -1$$

Let $\lambda_{\delta_0} = \min(\delta_0{}')$ over the domain. Since δ_0 is continuous and non-constant, it can be shown that $\lambda_{\delta_0} < 0$. To guarantee invertibility for any radius $r > 0$, we must restrict the segment length ε to:

$$\varepsilon \leq \frac{2r\,\delta_0(0.5)}{-\lambda_{\delta_0}}$$

This condition only makes sense if $\delta_0(0.5) > 0$, which should always be true – otherwise, the displacement at the center would point in the direction $-\mathbf{d}_i$, contradicting the intended deformation.

Example. The displacement function defined in Section 5.2 follows this construction with:

$$\delta_0(x) = (x(1-x))^k$$

Appendix C: No perfect maximal segment size

We show here that no universal value of τ can guarantee the absence of segment crossings after applying ω, if the spacing between iso-contours is unknown.

Consider a local function f_i. For any $\tau > 0$, it is always possible to find a segment $\mathbf{ab}$ of length τ, perpendicular to the direction $\mathbf{d}_i$, such that both endpoints $\mathbf{a}$ and $\mathbf{b}$ lie outside the support $\mathcal{B}(\mathbf{v}_i, r)$ while the segment itself intersects the support.

Inside the support, all points are displaced in the direction $\mathbf{d}_i$ after applying in f_i. Therefore, we can choose another segment $\mathbf{cd}$ of length at most τ, located below $\mathbf{ab}$, and entirely inside the support, such that $f_i(\mathbf{c})$ ends up above $\mathbf{ab}$ while $f_i(\mathbf{c})$ remains below it. This displacement produces an intersection between $f_i(\mathbf{cd})$ and $f_i(\mathbf{ab}) = \mathbf{ab}$. This configuration can always be constructed for any $\tau > 0$, it follows that no single value of τ can universally prevent crossings without knowledge of the local iso-contour spacing.