Modeling Rocky Scenery using Implicit Blocks

Axel Paris, Adrien Peytavie, Eric Guérin, Jean-Michel Dischler, Eric Galin

Modeling Rocky Scenery using Implicit Blocks

Axel Paris, Adrien Peytavie, Eric Guérin, Jean-Michel Dischler, Eric Galin

Modeling Rocky Scenery using Implicit Blocks

A. Paris[1] · **A. Peytavie**[1] · **E. Guérin**[1] · **J-M. Dischler**[2] · **E. Galin**[1]

Abstract We present a novel geologically-based method to generate vertical walls of rocky cliffs, crags or promontories. Our method procedurally generates a distribution of fractures in the bedrock to create a set of tiling blocks defined as implicit volumetric primitives. Blocks are in turn implicitly replicated over the vertical parts of the terrain and combined together to obtain a consistent volumetric representation of the fractured bedrock patterns using generalized union and blending operators. Our framework provides multiple levels of control: in addition to automatically generated blocks, the geometry of specific ones can be prescribed by the user using implicit primitives or construction trees, the shape of the blocks can be controlled by several parameters, and the placement rules may adapt according to the underlying geological strata and geometry of the terrain.

Keywords Implicit surfaces · Terrain Synthesis · Procedural Modeling

1 Introduction

Three-dimensional and vertical landforms such as cliffs, steep-walled canyons, crags, promontories, or overhangs are fundamental visual elements of scenic terrains. Despite the wide application of artificial terrains in the entertainment industry as well as simulation, and extensive research in this area, modeling truly 3D landforms with a high level of detail such as eroded karst tunnel networks, rocky cliff overhangs and arches with bare rock strata remains an unsolved problem. The vast majority of existing techniques addresses only $2\frac{1}{2}$D heightfield terrains which do not allow for an accurate representation of vertical landforms. Even though recent advanced techniques for modeling truly 3D terrains have been proposed [22,3,18], most of them have a limited resolution and only address the generation of large-scale features. Hyper-textures [20] can synthesize fractal mesoscale and microscale details over the surface of a coarse terrain, however, the self-similar appearance of the resulting bedrock, often corresponding to sandstone, lacks structure. Generating the mesoscale block structures and small-scale patterns of bedrock that appear on bare rocky terrain and the vertical walls of canyons, steep-walled cliffs or promontories, has received little attention.

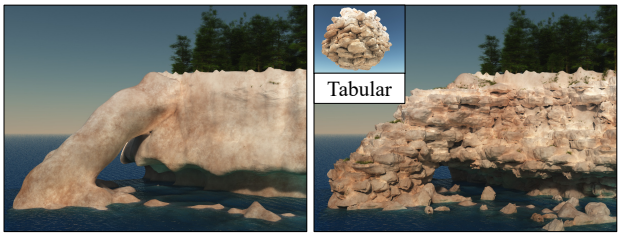

Fig. 1 Given an low resolution input terrain $\mathcal{T}$, our method generates fractured mesoscale blocks featuring small-scale details inside a cubic tile, and replicates them realistically in the scene according to the geology of the different strata.

The challenge stems from the fact that the geometry of rocky surfaces results from different physical processes (including fracturing, percolation, and erosion), depends on the materials involved (such as limestone, dolomite, sandstone or basalt), and shows in a variety of forms (from regular hexagonal prisms to seemingly chaotic polyhedral shapes) and scales (from a few decimeters to meters).

In this paper, we propose an amplification method that enhances an otherwise smooth input terrain by au-

[1] CNRS, Université de Lyon, LIRIS, France
[2] ICube, Université de Strasbourg, CNRS, France

tomatically carving geomorphologically consistent volumetric details over cliffs and overhangs. Our method consists in reproducing the fractures that exist in the bedrock to generate blocks that will be visible on the vertical walls of cliffs (Figure 1).

More precisely, the main contributions of our work are as follows: 1) we present an original geologically-based fracturing process to generate realistic blocks of rock separated by fractures tiling space; 2) we introduce a novel gradient-based warping operator for adding surface details to implicit primitives which allows us to carve small-scale rock patterns from synthetic or real images over blocks; 3) we define a field function node compatible with any hierarchical implicit surface modeling framework for replicating the field functions characterizing the blocks, therefore implicitly replicating them over a volumetric terrain or a heightfield. Altogether, we provide a unified implicit framework for the representation of mesoscale ($\approx 1\,\mathrm{m}$) and small-scale ($\approx 1\,\mathrm{cm}$) bedrock details allowing to reproduce complex bedrock patterns and shapes. Moreover, our method provides multiple levels of control, allowing the user to author blocks and tune the placement rules.

2 Related work

Our method relates both to 3D terrain modeling and to hyper-textures or amplification techniques that enhance an existing coarse terrain representation with smaller-scale details. For a more complete coverage of terrain modeling techniques, the reader is referred to the review in [7].

Voxel representations were first employed for modeling volumetric rocky landforms such as cliffs [12] by simulating fracturing between neighboring cells along rock joints. Stability analysis is then applied to remove disconnected cells. Besides memory and computational costs, the method does not scale to large terrains and does not address the distribution of joints crucial for synthesizing patterns observed in nature. Voxels were also used for simulating spheroidal and cavernous erosion producing volumetric structures such as hoodoos and goblins [2,13]. Limiting the range to small features allows to reproduce small-scale details on such specific landforms, but does not lend itself to simulating erosion on large terrains.

Implicit surfaces provide an alternative function-oriented mathematical framework for modeling 3D terrains. Peytavie *et al.* extend the concept of stacked layers, first introduced for erosion simulation, by incorporating water and air layers and thereby enabling the construction of caves and overhangs [22]. The final smooth surface is defined as a convolution of these layers. Although more compact than voxels, material stacks need to be stored explicitly and are limited in terms of precision. Another approach consists in combining voxels and feature curves [3] for modeling arches and overhangs. Again the method explicitly stores the voxels of the terrain making it memory demanding. Recently, a hierarchical implicit construction tree combining feature primitives [18] was proposed for modeling a vast variety of landforms including arches, caves, karsts, sea cliffs, Goblins by simulating erosion processes by an invasion percolation algorithm.

While effective for modeling general landforms, most existing terrain modeling techniques successfully generate smooth large-scale landforms but suffer from a lack of precision and fail at representing mesoscale and small-scale details. While procedural sum of scales-noise functions [6] can in theory compute an infinite amount of details, the self-similar geometries resulting from the fractal process do not capture the characteristic structures and patterns observed on real terrains. In contrast, our method generates an implicit tiling function combined with small scale gradient-based warping for representing the mesoscale structure and small-scale details conforming to observations in geomorphology.

A common approach often used in the entertainment industry consists in decorating a heightfield by distributing 3D meshes taken from an atlas of characteristic landforms over the vertical parts on the terrain. A specific technique for generating rock piles was proposed in [21]. It consists in creating aperiodic rock tiles to avoid computationally intensive physical simulation for stabilizing rocks. Rock shapes are generated by computing the Voronoi cells of a set of seed points using a parametrized anisotropic distance. While this approach can reproduce rock piles found at the bottom of cliffs, it does neither accounts for geological correctness nor follows a joint model, and therefore fails at reproducing certain types of blocks. *G*host Tiles introduced in [11] use a densely populated tile with pre-computed intersections to synthesize the multitude of entangled objects fallen to the ground such as rocks, branches or leaves. Those methods do not address the synthesis of geologically consistent fractures and do not create realistic block structures.

Finally, textures without uv-maps can be used to add details terrains by displacement mapping [28], but only few texturing methods can be applied to introduce real 3D features. Hyper-textures [20,6] often based on noise [19] allow the synthesis of a theoretically infinite amount of volumetric details. Hyper-textures perturb an initial smooth volumetric model by adding a fractal density function or warping space. Although this method generates fractal surfaces with self-similar fea-

tures such as powdery rocks or eroded limestone, it fails at reproducing block structures such as fractured sandstone or columnar basaltic shapes. Cellular textures [25] can be easily adapted to generalized Voronoi diagrams and synthesize volumetric blocks patterns. Discrete element textures [15] synthesize distributions of objects using a small input examplar, however, there is no guarantee to obtain geologically correct block structures. Texel mapping enhances the surface of an object with details stored in voxel grids and replicated around the surface of the input object [16]. This method produces 3D features on a thick layer only, requires an explicit parametrization of the surface, but fails at generating geomorphologically coherent mesoscale block structures, which is the focus of our work. An important aspect of our method is that it is compatible with these texturing techniques, which can be invoked to synthesize small and micro-scale details.

Contrary to previous approaches, our method is capable of reproducing geologically correct blocks enhanced with small-scale details and implicitly places and combines them over the original terrain to produce a detailed representation featuring the complex patterns observed in real landscapes.

3 Overview

Real terrains often feature complex rock formations in areas such as cliffs or caves. These landforms are the result of complex, interconnected physical processes that include glacial erosion, catastrophic rock collapses due to instability, or aeolian erosion. To avoid computationally intensive simulations, we propose a procedural approach to create block formations based on classifications used in geomorphology.

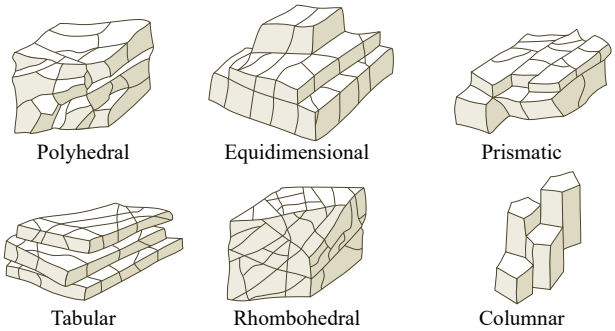

Fig. 2 Categories of blocks identified in geomorphology corresponding to various fracture distributions.

Our work comes from the observation that fractures in bedrock form blocks exhibiting a variety of shapes of different sizes. Archetype structures such as prismatic, equidimensional or rhombohedral (Figure 2) have been identified in geomorphology [17, 5]. A fracture is a break of continuity in the body of rock whose origin is natural. It most frequently occurs as *fracture sets* that are defined as a family of parallel, evenly spaced broken fractures that can be identified by analyzing their orientations, spacing, and physical properties. Regular and periodic distributions of fractures result in regular columnar blocks, whereas distributions with three major orthogonal directions result in equidimensional blocks; randomly distributed fractures produce polyhedral type.

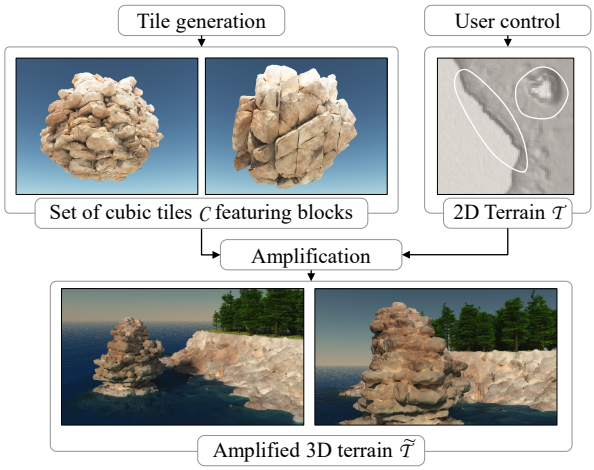

Fig. 3 Overview of the algorithm: during a pre-processing step, we generate cubic tiles containing 3D blocks of distinct types according to different fracture distributions, then given an initial coarse heightfield, strata definition and control regions, we automatically generate a 3D implicit model defined as a field function replicating the blocks for creating mesoscale and small-scale details.

Fracturing the entire input terrain $\mathcal{T}$ would be computationally and memory intensive. Therefore, we propose a procedural tiling method. The goal is to compute cubic tiles containing geologically correct blocks that will be placed over the vertical parts of bare bedrock.

The overall terrain amplification process is composed of two steps (Figure 3). We first generate a set of blocks $\mathcal{B}_i$ organized into a cubic tile denoted as $\mathcal{C}$ using a procedural fracturing approach based on a geomorphological classification (Section 4). These blocks $\mathcal{B}_i$ are implicitly defined by scalar functions $b_i : \mathbb{R}^3 \to \mathbb{R}$, which allows to obtain a 3D volumetric representation of the mesoscale patterns and small-scale details using a new gradient-based warping operator.

The second step consists in amplifying a smooth input terrain $\mathcal{T}$ by replicating blocks over the bare vertical parts of the bedrock (Section 5). The input terrain can be any kind of $2\frac{1}{2}$D heightfield (resulting from edit-

ing, simulation, procedural generation or synthesized from examples, we refer to [7]), or a 3D terrain model (see Section 2). The new 3D detailed terrain model $\widetilde{\mathcal{T}}$ is defined as an implicit surface whose scalar field function $\widetilde{f}$ is a construction tree combining the field function of the initial terrain f and the scalar functions of the blocks b_i.

The cubic tile $\mathcal{C}$ is virtually tiling $\mathbb{R}^3$, and only the blocks $\mathcal{B}_i$ that straddle the vertical parts of the terrain are replicated. To avoid explicit instantiation, we propose an original selective field function replication method inspired from [23] that allows to virtually replicate the field functions b_i of the blocks only over the vertical parts of the input terrain.

4 Block tile generation

The key process in the formation of blocks is *fracturing*. In geomorphology, fractures are represented and simulated using 3D discs that define the formation of blocks by breaking the continuity of the bedrock.

From this observation, we propose a procedural and controllable method to simulate the different types of block formations as found in nature. We address the generation of blocks tiling space. In the following section, we consider periodic tiling out of clarity, the generation of aperiodic tiling using Wang Cubes [4] or Corner Cubes [14, 21] is a direct technical generalization of this work. Therefore, we address the generation of blocks in a cubic tile $\mathcal{C}$ of size s (in our implementation we use a cube with size $s \approx 20\,\mathrm{m}$). The algorithm proceeds in two steps as depicted in Figure 4.

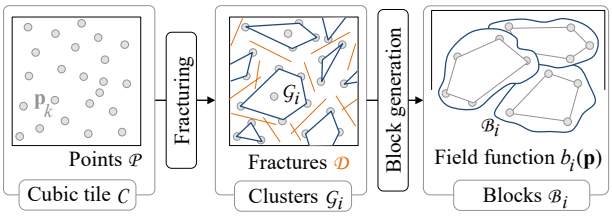

Fig. 4 Blocks generation pipeline inside a cubic tile $\mathcal{C}$. We first compute a nearest neighbor graph $\mathcal{G}$ over a set of sample points $\mathcal{P}$ inside the tile. Fracturing discs $\mathcal{D}$ remove edges crossing fractures. We extract clusters $\mathcal{G}_i$ as the set of disconnected sub-graphs, and generate implicit primitives $\mathcal{B}_i$ for each cluster. For the sake of clarity, the method is depicted in 2D and edges of $\mathcal{G}$ are not represented.

Fracturing. Starting from an initial set of points $\mathcal{P} = \{\mathbf{p}_k\}$ sampling the cubic tile, we compute the geometric nearest neighbor graph $\mathcal{G}$ over this set. A set of fractures $\mathcal{D} = \{\mathcal{D}_i\}$ where $\mathcal{D}_i$ are discs is then generated using procedural rules. These fractures cut edges from the graph, thus creating connected sub-graphs called clusters and denoted as $\mathcal{G}_i$ that will finally form the blocks.

Implicit primitive generation. The coarse geometry of the blocks is defined as the convex hull of the point sets for every cluster $\mathcal{G}_i$. For every block, we define a scalar function b_i that computes a signed distance bound to the surface of the block. Its corresponding construction tree combines the planes of the convex hull using a smooth intersection operator. The convex blocks are responsible for the mesoscale details of the bare rock. We finally apply a new gradient-based warping operator for generating the small-scale volumetric details.

4.1 Fracturing

The fracturing process starts by generating a set of sample points $\mathbf{p}_k$ inside the cube, using a Poisson sphere distribution. Experiments demonstrated that a regular sampling leads to 3D aliasing with unnatural axis-aligned fractured shapes.

The Poisson radius r influences the size and shape of the blocks. Higher radius values yield tiles with fewer points inside, which in turn leads to blocks with coarser convex shapes and fewer polygonal faces after the fracturing process. For a cubic tile size of $s \approx 20\,\mathrm{m}$, we generate points with a Poisson radius $r \approx 10\,\mathrm{cm}$ which leads to $\approx 14\,000$ points. After fracturing, each block contains between 80 and 150 sample points (see Table 1). We then compute the nearest neighbor graph $\mathcal{G}$ connecting the set of points $\mathcal{P}$ using r as the neighboring distance threshold, *i.e.* an edge between points $\mathbf{p}_i$ and $\mathbf{p}_j$ exists if $\|\mathbf{p}_i - \mathbf{p}_j\| < r$.

Fractures $\mathcal{D}_i(\mathbf{c}_i, r_i, \mathbf{n}_i)$ are defined as discs in space characterized by their centers $\mathbf{c}_i$, radii r_i and normal orientation $\mathbf{n}_i$. Geomorphological types are distinguished by the way fractures are distributed in the cube, *i.e.* by the following parameters: number of fractures, average radius size, distribution of disc centers, and relative orientations. Without loss of generality, discs centers $\mathbf{c}_i$ are randomly generated in the cube using a Poisson sphere sampling with an average radius of $\approx 2\,\mathrm{m}$. Both r_i and $\mathbf{n}_i$ distributions are tuned according to the block type (see Figure 5).

The user may control the fracturing process either by tuning the disc distribution parameters, or by placing specific fractures in the cubic tile, or by directly placing several authored blocks in the tile (see Figure 6), and the system will adapt. It is also possible to increase the overall amount of fracture by decreasing the Poisson radius of the fracture centers distribu-

tion. Other disc sampling strategies could be used, however we found Poisson sampling practical for obtaining regular-pattern-free distributions with a practical control over the disc centers spacing.

The following paragraphs explain the different fracture distributions, as described in [17] and their control parameters.

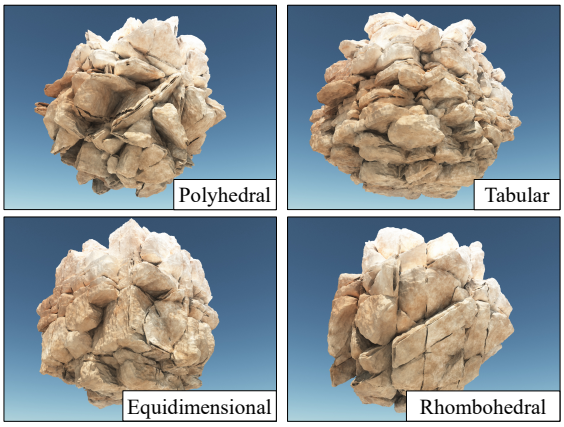

Fig. 5 Different types of blocks generated with different disc distributions.

Equidimensional block type has three dominant sets of fractures, approximately orthogonal, with occasional irregular fractures, giving almost cubic-shaped blocks. Prismatic blocks are similar in shape but are formed under slightly more irregular fracture distributions. For these types, normals $\mathbf{n}_i$ are determined using a random axis-aligned direction. Radii r_i are randomly chosen in an interval given by a user parameter. In our experiments, we set the radii to ≈ 5m.

Polyhedral type shows irregular small blocks without explicit arrangement into distinct sets. This type requires more fractures due to the completely stochastic essence of the distribution, therefore we set the radius of the fracture centers distribution to ≈ 1 m and create more fractures.

Rhombohedral blocks have three (or more) dominant mutually oblique sets of fractures, giving oblique-shaped blocks. The parameters are the same as for equidimensional blocks, but with tilted axis-aligned directions to obtain diagonal orientations.

Tabular block type has one dominant set of parallel fractures orthogonal to a dominant axis direction, for example bedding planes, with other non-persistent fractures; thickness of blocks is much lower than length or width. Discs orthogonal to the dominant axis orthogonal have a large radius equal to the size s of the cubic tile $\mathcal{C}$, whereas the other discs parallel to the two orthogonal axes occur less frequently, and have with small radii $r_i \approx s/10$.

Columnar type is composed of several (usually more than three) sets of continuous, parallel fractures. The length is much greater than other dimensions.

Type	$\#\mathcal{P}$	$\#\mathcal{D}$	$\#\mathcal{B}$	$\#\overline{\mathcal{P}}_i$	Time
Equidimensional	14 524	15	125	110	6.9
Rhombohedral	14 589	24	78	161	12.0
Polyhedral	14 532	44	100	140	25.4
Tabular	14 554	30	82	143	16.3

Table 1 Statistics for the generation tiles: number of points $\#\mathcal{P}$, number of fracture discs $\#\mathcal{D}$, number of blocks $\#\mathcal{B}$, average number of points $\#\overline{\mathcal{P}}_i$ inside a block $\mathcal{B}_i$, and generation time (in seconds).

The next step consists in removing edges cut by a disc $\mathcal{D}_i$ and then create clusters $\mathcal{G}_i$ of connected nodes $\mathbf{p}_k$. This is performed by using a greedy algorithm: starting from a random point, aggregation propagates through the graph until no more points can be connected. Formally, an edge $\mathbf{p}_j\mathbf{p}_k$ is kept if it does not intersect any fracture:

$$\forall\, \mathcal{D}_i \in \mathcal{D},\ \mathcal{D}_i \cap \mathbf{p}_j\mathbf{p}_k = \varnothing$$

Finally, we check that all the points in a given cluster are visible to each other, *i.e.* that fractures do not cut an edge connecting any pair of points in the cluster. This guarantees that clusters should not spread around fracture discs, which would create blocks that do respect the constraints.

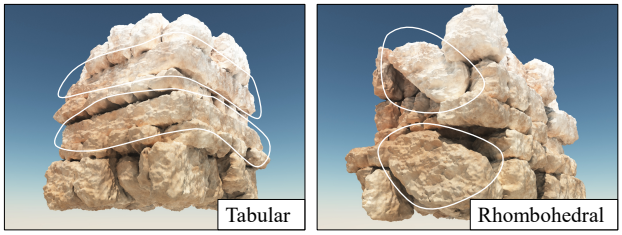

Fig. 6 Example of user-control during the fracturing step: the user authored specific tabular blocks (left), and rhombohedral blocks (right) inside an equidimensional block tile.

The complexity of the fracturing step depends on the number of fractures $\#\mathcal{D}$: a highly fractured cube will require more intersection tests between edges of

the graph and the discs. Table 1 reports some statistics for the different types illustrated in Figure 5.

4.2 Implicit block generation

Recall that we aim at generating an amplified terrain model $\widetilde{\mathcal{T}}$ as an implicit surface. Implicit surfaces allow us to create a unified volumetric representation for both the base terrain and the mesoscale and small-scale details of the bedrock. We create the scalar field b_i from the previously computed clusters $\mathcal{G}_i$ as a construction tree defined as the smooth-intersection of half-spaces forming a base convex shape, and then deformed using a gradient-based warping operator (Figure 7).

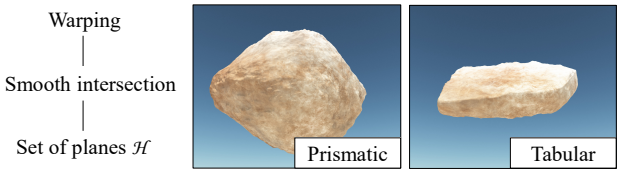

Fig. 7 Simplified hierarchical representation of blocks $\mathcal{B}_i$: the base convex shape is defined as the smooth intersection of a set of plane primitives $\mathcal{H}$, and procedurally warped.

We first compute the convex hull of each cluster and extract a corresponding polygonal mesh $\mathcal{M}_i$. We compute the planes $\mathcal{H}_k = (\mathbf{o}_k, \mathbf{u}_k)$ associated to every polygon of $\mathcal{M}_i$ such that their normal $\mathbf{u}_k$ should be oriented towards the exterior of $\mathcal{M}_i$. The corresponding scalar function of the convex base c_i is defined as the smooth intersection of the half-spaces functions h_k associated to planes $\mathcal{H}_k$. Every half-space is defined by the signed distance to the plane $h_k(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_k) \cdot \mathbf{u}_k$. The final implicit convex blocks $\mathcal{B}_i$ are finally defined as:

$$b_i(\mathbf{p}) = c_i \circ \omega^{-1}(\mathbf{p})$$

where $c_i(\mathbf{p})$ denotes the function associated to the block $\mathcal{B}_i$, and ω^{-1} denotes a gradient-based warping operator modeling small-scale details. Should the points of the cluster $\mathcal{G}_i$ be coplanar, or contain less than 3 points, we define the base geometry as a thick polygon, line segment or point primitive.

Smooth convex base Using a traditional intersection operator [26] defined as $f_{A\cap B} = \max(\mathrm{f_A}, \mathrm{f_B})$ creates gradient discontinuities (Figure 8) which prevents the use of gradient-based warping as the resulting field function would no longer be continuous. Therefore, we use the smooth intersection operator introduced in [1] over the n planes of the block $\mathcal{B}_i$. Without loss of generality, we consider the intersection in the case of two half-spaces A and B. The field function of the smooth

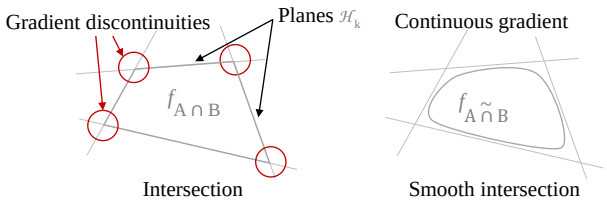

Fig. 8 Smooth intersection generates a convex block shape with rounded edges and preserves the continuity of the gradient ∇c_i, therefore allowing a correct gradient-based warping.

intersection $A\widetilde{\cap}B$ is parameterized by a radius R and defined as:

$$f_{A\widetilde{\cap}B} = \max(f_A, f_B) - R\,k(f_A, f_B)^3/6$$
$$k(f_A, f_B) = \max(1 - |f_A - f_B|/R, 0)$$

Higher values for R lead to smoother shapes, whereas smaller values preserve sharp features. In our method, we set $R = 25\,\mathrm{cm}$. Since the smooth intersection operator is not associative, the order in which intersections are performed may influence the final scalar field. In practice, the impact of the ordering is limited and appears not to have any visual impact over the block shape. Different operators [10] could also be used, as long as they do not introduce gradient discontinuities.

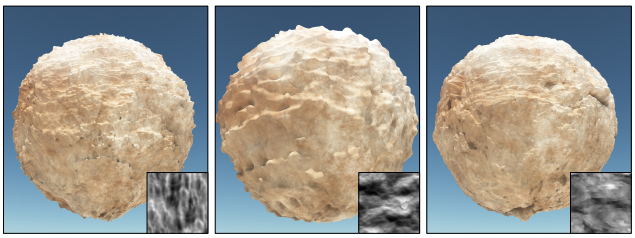

Fig. 9 Examples of gradient-based warping with different reliefs applied to a sphere primitive.

Surface details Adding surface details to implicit surfaces is a challenging problem as implicit surfaces do not provide an explicit parameterization. Existing methods rely either on interactive authoring [24] or require an explicit parameterization of the implicit surface [29], and do not lend themselves for generating the surface details of blocks. We introduce a new gradient-based warping operator for adding details to any implicit surfaces, taking inspiration from tri-planar projection [9]. This warping, applied for every block, does not require an explicit parameterization of the surface, and introduces small-scale volumetric details as displacements that enhance the model. Figure 9 shows the effect of warping with different relief functions d encoded as images. The operator is defined as:

$$\omega^{-1}(\mathbf{p}) = \mathbf{p} + \delta(\mathbf{p})$$

Let $g(\mathbf{p}) = \nabla b(\mathbf{p})/\|\nabla b(\mathbf{p})\|$ denote the normalized gradient. Let $\gamma_i(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^2$ denote the projection of $\mathbf{p}$ on the i-th plane, namely xy, xz and yz. The function $\delta(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^3$ computes the 3D warping of $\mathbf{p}$ according to a 2D displacement function $d : \mathbb{R}^2 \to \mathbb{R}$:

$$\delta(\mathbf{p}) = g(\mathbf{p}) \sum_{i=0}^{3} \alpha_i \circ g(\mathbf{p})) \cdot d \circ \gamma_i(\mathbf{p})$$

The weighting function $\alpha_i : \mathbb{R}^3 \to \mathbb{R}$ weights the contributions of the three displacements $d \circ \gamma_i(\mathbf{p})$ according to the scalar product between the normalized gradient and the unit axis-aligned vectors $\mathbf{u}_i$: $\alpha_i(\mathbf{p}) = |g(\mathbf{p}) \cdot \mathbf{u}_i|$.

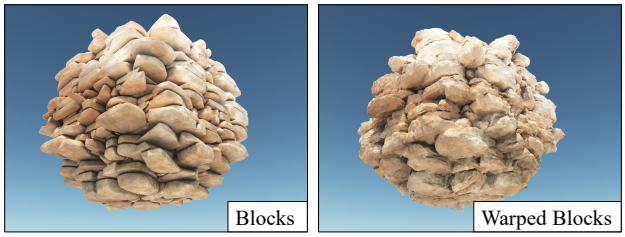

Fig. 10 Tabular block tile without warping (left) and after gradient-based warping (right): this new operator allows to create highly detailed blocks without holes.

The function $d : \mathbb{R}^2 \to \mathbb{R}$ computes the displacement distance and can be effectively defined either as a procedurally defined turbulence, or from real displacement images. It is computed for every projection of the point $\mathbf{p}$, thus three times. Figure 10 shows the effect of gradient warping for the tabular type.

5 Terrain amplification

The amplification process aims at generating a modified terrain $\widetilde{\mathcal{T}} = \mathcal{T} \cup \mathcal{R}$ defined as the union of $\mathcal{T}$ and the replication $\mathcal{R}$ of some of the blocks $\mathcal{B}_i$ in $\mathcal{C}$ over the bare vertical walls of the cliffs. We propose a new replication operator transforming the initial scalar field f into an amplified model $\widetilde{f}$.

Infinite replication of a scalar field b_i over the entire space, first addressed in [23], can be obtained by directly computing b_i with the modulo between the argument point $\mathbf{p}$ and the size s of the tiling cube: $b_i(\mathbf{p} \bmod s)$ with $\mathbf{p} \bmod s = \mathbf{p} - s \lfloor \mathbf{p}/s \rfloor$. In our case, the challenge consists of computing the scalar fields b_i only for the blocks $\mathcal{B}_i$ that straddle the terrain at certain positions, therefore according to the field function representing the terrain f. We employ a presence function $e : \mathbb{R}^3 \to \{0, 1\}$ that evaluates whether a block $\mathcal{B}_i$ should be replicated or not.

For every block $\mathcal{B}_i$, we define a corresponding anchor point $\mathbf{a}_i$ that will be used to evaluate its presence. Here we present the concept with only one anchor point per block out of clarity, but the method can be easily generalized for a set of anchor points. Recall that $\lfloor \mathbf{p}/s \rfloor$ denote the integer coordinates of the virtual cell containing $\mathbf{p}$. Blocks are selectively replicated by computing the presence e function at the virtual anchor point in world space $\mathbf{a}_i + s \lfloor \mathbf{p}/s \rfloor$. The replication operator is a function t which defines its field function as the union of all block field function b_i times their presence function e_i.

$$t(\mathbf{p}) = \max_i b_i(\mathbf{p} \bmod s)\, e_i(\mathbf{a}_i + s \lfloor \mathbf{p}/s \rfloor)$$

The final function $\tilde{f}$ of the terrain is defined as $\tilde{f} = \max(f, t)$.

More precisely, the evaluation of the replication operator at a point $\mathbf{p}$ is performed as follows. We evaluate in which cell of the grid lies $\mathbf{p}$, using a modulo operation on the floating point coordinates of $\mathbf{p}$. We compute the contribution of each block function b_i, virtually translated in the cell for the point $\mathbf{p}$.

If the distance from $\mathbf{p}$ to a border of a cell is less than a given threshold we compute the contribution in neighboring cells to account for blocks straddling $\mathcal{C}$. Therefore, we compute the total contribution as the union between blocks in the current cell and 2, 4 or 8 neighboring cells.

In the case of a set $\mathcal{A} = \{\mathbf{a}_k\}$ of several anchor points, the presence function e_i computes the percentage of anchor points $\mathbf{a}_i$ of $\mathcal{B}_i$ which satisfy geometric criteria. In our implementation, we replicate a block if more than half, *i.e.* $\#\mathcal{A}/2$, of the anchor points satisfy $e(\mathbf{a}_k) = 1$.

The presence function can be any combination of a variety of criteria; here we briefly review some important ones that allow for the automatic placement of blocks over the vertical walls, and that provide control by prescribing a geological strata definition as presented in [18].

We first define criteria based on the distance to the surface: in our model, a block can be replicated only if enough anchor points $\mathbf{a}_k$ are in a given distance range $[v_a, v_b]$ to the surface, i.e $f(\mathbf{a}_k) \in [v_a, v_b]$. To only replicate blocks on steep slopes and vertical walls, we compare the direction of the gradient with the up direction u_z; recall that g denotes the normalized gradient of the terrain, we set $e(\mathbf{a}_k) = 1$ if $|g(\mathbf{p}) \cdot u_z| < \varepsilon$, with ε the maximum slope parameter. Finally, we extend on the implicit geological function γ defined in [18] and define different bedrock material at different location in the scene. Therefore, an anchor point $\mathbf{a}_k$ satisfies the repli-

Fig. 11 A canyon amplified with equidimensional blocks located at the bottom of the canyon, and tabular blocks placed on the higher parts of the cliff walls.

cation criteria if the material $\gamma(\mathbf{a}_k)$ corresponds to the material of the underlying block $\mathcal{B}_i$.

Note that the presence function is generic and can be easily extended to account for other criteria such as the presence of a water level, trees and obstacles.

6 Results

We implemented our method in C++ and all the scenes were generated on a desktop computer equipped with Intel® Core *i*7, clocked at 4 GHz with 16GB of RAM, and an NVidia GTX 970 graphics card. The implicit surface representing the amplified terrain was polygonized [27] and the resulting mesh directly streamed into Vue Xstream® to produce the final images (Figures 5, 6, 9, 10, 11, 12, 13, 14).

Scene	Figure	Size	$\#\mathcal{R}$	Memory
Sea cliff	1, 14	100×100	959	0.57
Canyon	11	100×50	1041	0.54
Sea spire	3	150×150	1680	0.53

Table 2 Statistics for the scenes: size (in meters), number of replicated blocks $\#\mathcal{R}$, and amount of memory (in megabytes) needed to store the field functions b_i representing the blocks $\mathcal{B}_i$ in the cubic tiles $\mathcal{C}$.

Table 2 reports some statistics about the different scenes. Contrary to noise-based hyper-textures that only require a few dozens of parameters, our method needs to store the hierarchical implicit models of the different blocks; the required amount of memory remains small (less than a megabyte).

6.1 Control

Figure 5 a variety of types of blocks: different fractures distributions were prescribed (Section 4) and lead to different shapes such as polyhedral, rhombohedral or tabular blocks. The user can tune the parameters of different fractures within a tile, and our method automatically computes block shapes that adapt to these constraints; it is also possible to sculpt specific blocks and let the system automatically adapt and generate the remaining ones, as demonstrated by Figure 6, where the user placed specific tabular or rhombohedral blocks at the desired locations in the tiles.

Figure 11 illustrates user-control over the block placement: an equidimensional bedrock strata was defined by the user at the bottom of the cliff resulting in regular cliff walls. A second strata with tabular type was prescribed above the previous one and tabular blocks were automatically added to account for the specified material. In practice, it takes a few iterations for the user to tune the placement rules of the different block tiles for a specific scene; the main limitation remains the computationally intensive visualization of the final implicit surface (see Section 6.4).

Figure 11 also demonstrates the effectiveness of control based on the computation of the gradient ∇f for detecting the vertical parts of the terrain: no blocks were generated on the top of the plateau, thus keeping the ground flat where trees appear. Figure 14 shows different styles of blocks applied to a sea cliff scene, completely changing the overall mesoscale geometry of the final landscape.

Finally, micro-scale details are defined by using the gradient-based warping operator combined with different displacement maps as showcased in Figure 9. The relief function d can be painted by the user; procedurally defined by combining noise functions in construction tree of scalar primitives or scanned from real rock surfaces.

6.2 Comparison with other methods

While the sum of scaled-noise functions [6] can, in theory, produce an infinite amount of details, the self-similar geometries resulting from the fractal process do not capture the characteristic structures and patterns observed on real terrains. Figure 12 shows a comparison

of different methods used to add details onto a sphere. Hyper-textures [20,6] may generate holes and disconnected parts, star-shaped primitives with a radial turbulence [18] avoids artifacts but both methods lack geological structure. In contrast, fracturing generates geomorphologically consistent blocks, and gradient-based warping allows for the generation of small-scale details captured from real rocks. Hyper-textures [20] could also be used to add small-scale details, however gradient-based warping allows for better control over the displacement by using reliefs from real rocks.

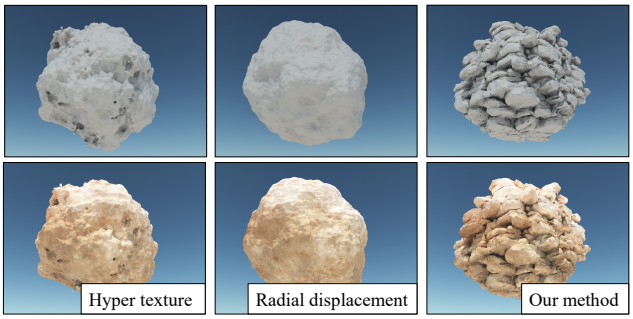

Fig. 12 Comparison of different methods used to add details onto an implicit sphere, with an ambient shading (top) and a high resolution texture (bottom).

In terms of mesoscale details, our method improves existing volumetric terrain models. Global warping operators [8] do not provide sufficient accuracy to reproduce the complex geometry of the vertical parts of the bedrock. Voxels and features curves-based approaches [3] are limited by the grid resolution and generate smooth large-scale terrains.

While the system described in [18] can generate large scale landforms such as arches, overhangs, and simulate large scale erosion effects, the vertical surface of cliffs lacks geometrical patterns and bedrock details. Although primitive-based implicit surfaces can theoretically reproduce small-scale details, their modeling comes at the price of defining a huge number of small primitives in the construction tree, a memory-intensive process. In contrast, our method defines a memory-efficient tiling and replication function that implicitly tiles space with a union of blocks to amplify and add details to cliffs.

In spirit, generating and replicating instances resembles the Ghost Tiling approach [11], and the rock pile generation based on aperiodic tiling [21] that instantiate rock meshes. Our approach differs in the sense that blocks are defined as implicit primitives and virtually replicated by using a replication operator combined with a presence function, which allows us to combine them to produce a consistent volumetric model. Moreover, these methods would need to be improved to account for fracture distributions to guarantee the replicability of certain block types - which is one primary focus of our work.

6.3 Compatibility with other techniques

An important aspect of this work is that it is compatible with other terrain modeling techniques. Our method can benefit to implicit surface-based terrain representations such as the hybrid layer-stack convolution-surface framework described in [22] or the primitive-based system [18]. Those models, designed for modeling large scale terrains, can be amplified with mesoscale details as demonstrated in Figure 13. Moreover, our method lends itself to amplifying heightmaps or procedurally defined elevations since a three-dimensional scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ can be directly derived from the elevation function $h : \mathbb{R}^2 \rightarrow \mathbb{R}$ by defining $f(\mathbf{p}) = \mathbf{p}_z - h(\mathbf{p}_{xy})$.

Fig. 13 Side by side comparison showing a canyon with smooth overhangs produced by the layer-stack model, and the rocky amplified vertical-wall generated by blocks.

Finally, although we used implicit surfaces for modeling the complex mesoscale 3D features of terrains, the block generation algorithm (Section 4) can also produce a mesh representation $\mathcal{M}_k$ for every block $\mathcal{B}_k$ in the tile. In this context, our method can be streamed in production environments using mesh representations and is compatible with standard instantiation techniques such as the ones described in [21] or [11] to distribute block meshes over the vertical parts of the input terrain.

6.4 Limitations

The block tile generation process (Section 4) can take up to 30s because of the many intersection tests between segments (linking two points in the tile) and fractures discs. However, time was not spent on optimizing

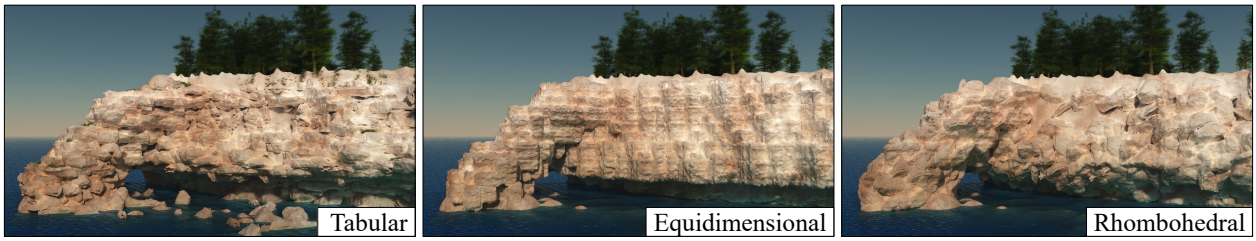

Fig. 14 Different styles of blocks generated on a cliff and arches. From left to right, tabular block style, equidimensional blocks and finally rhombohedral block style.

this step of our pipeline, since it is done once in pre processing. A more limiting property of our method is the fact that our block primitives can only model convex shapes. This could be resolved by developing primitives more suited for non-convex shapes, but it is beyond the scope of this paper and is left as future work.

Although implicit surfaces provide a powerful, sparse, and compact framework for generating complex volumetric structures, their visualization remains computationally intensive. In particular, the computation of the field function b_i for one single block $\mathcal{B}_i$ requires many half space distance computations, combined with a warping operator: blocks could be optimized with a limited number of primitives to reduce the overall number of field function queries.

Using a limited set of block tiles sometimes yields visible repetitions when applied to flat cliff walls parallel to the tiling directions. Aperiodic tiling using Wang cubes [4] or Corner Cubes [14,22] would reduce visible artifacts, yet at the price of a larger number of precomputed tiles and, therefore, a more memory demanding implementation.

Finally, the procedural fracturing tool only provides an indirect control over the distribution of fractures, and thus the shape of the blocks: a better interactive material editing tool would allow the user to author a wider range of shapes conforming to his intent.

7 Conclusion

We have introduced an implicit representation of blocks for automatically generating mesoscale and microscale details on the vertical walls of rocky cliffs, crags, or promontories. Our method relies on the decomposition of a bedrock tile into blocks using a fracturing approach based on classifications in geomorphology. The implicit surface description allows generating realistic shapes enhanced with details organized into patterns and structures as observed in geomorphology. Our method is compatible with other terrain modeling techniques: the block generation algorithm can additionally produce triangle meshes, which in turn can be instantiated to enhance heightfields. We also introduced a warping operator that can be applied to any implicit surface model to add small-scale details.

A direct extension of this work would consist of generalizing with different types of blocks, in particular prismatic and columnar configurations that require specific fractures distributions. Another avenue of research worth investigating would be to generate large-scale fractures spreading over hundreds of meters and shaping the vertical walls of cliffs by taking into account the underlying strata of the bedrock.

Conflict of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work was part of the project HDW ANR-16-CE33-0001, supported by Agence Nationale de la Recherche.

References

1. Barthe, L., Gaildrat, V., Caubet, R.: Combining implicit surfaces with soft blending in a CSG tree. CSG Conference Series pp. 17–31 (1998)
2. Beardall, M., Farley, M., Ouderkirk, D., Reimschussel, C., Smith, J., Jones, M., Egbert, P.: Goblins by spheroidal weathering. In: Proceedings of Third Eurographics Conference on Natural Phenomena, pp. 7–14 (2007)
3. Becher, M., Krone, M., Reina, G., Ertl, T.: Feature-based volumetric terrain generation and decoration. IEEE Transactions on Visualization and Computer Graphics **25**(2), 1283–1296 (2019)
4. Culk, K.: An aperiodic set of 13 wang tiles. Discrete Mathematics **160**(1-3), 245–251 (1996)
5. Dearman, W.: Engineering geological mapping. Butterworths advanced series in geotechnical engineering. Butterworth-Heinemann (1991)
6. Ebert, D.S., Musgrave, F.K., Peachey, D., Perlin, K., Worley, S.: Texturing and Modeling: A Procedural Approach, 3rd edn. The Morgan Kaufmann Series in Computer Graphics. Elsevier (1998)

7. Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M.P., Benes, B., Gain, J.: A review of digital terrain modeling. Computer Graphics Forum (proceedings of Eurographics 2019 STAR) **38**(2), 553–577 (2019)
8. Gamito, M.N., Musgrave, K.F.: Procedural landscapes with overhangs. In: Proc. of the Portuguese Computer Graphics Meeting. Lisbon, Portugal (2001)
9. Geiss, R.: Generating complex procedural terrains using the GPU. In: GPU Gems 3, chap. 1, pp. 7–37. Addison-Wesley (2008)
10. Gourmel, O., Barthe, L., Cani, M.P., Wyvill, B., Bernhardt, A., Paulin, M., Grasberger, H.: A gradient-based implicit blend. ACM Transactions on Graphics **32**(2) (2013)
11. Guérin, E., Galin, E., Grosbellet, F., Peytavie, A., Génevaux, J.D.: Efficient modeling of entangled details for natural scenes. Computer Graphics Forum (Proceedings of Pacific Graphics 2016) **35**(7), 257–267 (2016)
12. Ito, T., Fujimoto, T., Muraoka, K., Chiba, N.: Modeling rocky scenery taking into account joints. In: Proceedings of Computer Graphics International, pp. 244–247. IEEE, Tokyo, Japan (2003)
13. Jones, M., Farlay, M., Butler, M., Beardall, M.: Directable weathering of concave rock using curvature estimation. IEEE Transactions on Visualization and Computer Graphics **16**(1), 81–97 (2010)
14. Lagae, A., Dutré, P.: An alternative for Wang tiles: Colored edges versus colored corners. ACM Transactions on Graphics **25**(4), 1442–1459 (2006)
15. Ma, C., Wei, L.Y., Tong, X.: Discrete element textures. ACM Trans. Graph. **30**(4) (2011)
16. Neyret, F.: Modeling animating and rendering complex scenes using volumetric textures. IEEE Transactions on Visualization and Computer Graphics **4**(1), 55–70 (1998)
17. Palmstrom, A.: Measurement and characterization of rock mass jointing, pp. 49–97 (2001)
18. Paris, A., Galin, E., Peytavie, A., Guérin, E., Gain, J.: Terrain amplification with implicit 3D features. ACM Transactions on Graphics **38**(5), 147:1–147:15 (2019)
19. Perlin, K.: An image synthesizer. SIGGRAPH Computer Graphics **19**(3), 287–296 (1985)
20. Perlin, K., Hoffert, E.M.: Hypertexture. SIGGRAPH Computer Graphics **23**(3), 253–262 (1989)
21. Peytavie, A., Galin, E., Grosjean, J., Mérillou, S.: Procedural Generation of Rock Piles Using Aperiodic Tiling. Computer Graphics Forum **28**, 1801–1809 (2009)
22. Peytavie, A., Galin, E., Mérillou, S., Grosjean, J.: Arches: A framework for modeling complex terrains. Computer Graphics Forum **28**(2), 457–467 (2009)
23. Stolte, N.: Infinite implicit replication: Case study for voxelizing and representing cyclical parametric surfaces implicitly. Shape Modeling International'02, pp. 105–111 (2002)
24. Sugihara, M., Wyvill, B., Schmidt, R.: Warpcurves: A tool for explicit manipulation of implicit surfaces. Computers & Graphics **34**(3), 282–291 (2010)
25. Worley, S.: A cellular texture basis function. SIGGRAPH 96, pp. 291–294 (1996)
26. Wyvill, B., Guy, A., Galin, E.: Extending the CSG tree – Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. Computer Graphics Forum **18**(2), 149–158 (1999)
27. Wyvill, G., McPheeters, C., Wyvill, B.: Data structure for soft objects. The Visual Computer **2**, 227–234 (1986)
28. Yuksel, C., Lefebvre, S., Tarini, M.: Rethinking texture mapping. Computer Graphics Forum **38**(2), 535–551 (2019)
29. Zanni, C., Bares, P., Lagae, A., Quiblier, M., Cani, M.P.: Geometric Details on Skeleton-based Implicit Surfaces. In: Eurographics Short Papers, pp. 49–52. Cagliari, Italy (2012)