EG UK Theory and Practice of Computer Graphics (2008)
Ik Soo Lim, Wen Tang (Editors)

Localised Topology Correction for Hypertextured Terrains

Manuel N. Gamito[†] and Steve C. Maddock

Department of Computer Science
The University of Sheffield

Abstract

Terrains for computer graphics have traditionally been modelled with height fields. In the case of procedurally defined terrains, the height field is generated by a displacement map $z = f(x,y)$. Displacement maps, however, do not allow the formation of such terrain features as overhangs, arches or caves. The most flexible approach to model procedural terrain is to consider it as an implicit surface, given by $f(x,y,z) = 0$. Hypertexturing can be used to add terrain features to an initially smooth implicit surface. The main drawback of modelling terrains as hypertextures is that terrain fragments can easily become disconnected.

A topology correction method is presented that is able to remove all topologically disconnected fragments of a hypertextured terrain, leaving only the main surface of the terrain. The method works in the context of a ray casting algorithm for implicit surfaces and for any given terrain point indicates whether it is connected or not. The method is localised because it only needs to examine the terrain inside a neighbourhood of sufficiently small size around the point that is being tested in order to determine its connectivity state. This localised connectivity test allows the modelling of procedural terrains over wide areas such as terrains defined over an infinite horizontal plane. Our method works with terrains that are C^2 continuous and uses Morse theory to find the critical points of the terrain. The method follows the separatrix lines joining the critical points to isolate disconnected fragments.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image Generation]: Viewing algorithms I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations I.3.7 [Three-Dimensional Graphics and Realism]: Raytracing

1. Introduction

This paper is concerned with the procedural generation of terrains containing features such as overhangs, caves and arches. Although triangle meshes or parametric patches can be used to that effect, they cannot generate terrain over arbitrarily large regions. Only a procedural modelling approach has the ability to generate terrain detail everywhere in space and at any level of detail. This gives the user complete freedom to roam around the terrain at various altitudes.

Hypertexturing is a procedural technique proposed by Perlin & Hoffert to add three-dimensional small-scale detail to the surface of smooth objects [PH89]. When applied to implicit surfaces, it can be used to create the type of terrain features that are addressed in this paper. In the original definition of hypertexture, an object is defined in three-dimensional space with an *object density function* $f_0 : \mathbb{R}^3 \to [0,1]$, which associates a density value $f_0(\mathbf{x})$ with every point $\mathbf{x}$ in space. A density of 0 means total transparency while a density of 1 means total opacity. The shape of the object can then be deformed through the composition of f_0 with one or more *density modulation functions* $f_i : [0,1] \times \mathbb{R}^3 \to [0,1]$, $i = 1, \ldots n$ such that the final object density f is given by:

$$f(f_0(\mathbf{x}),\mathbf{x}) = f_n(\ldots f_2(f_1(f_0(\mathbf{x}),\mathbf{x}),\mathbf{x}),\ldots,\mathbf{x}). \qquad (1)$$

It is possible to apply (1) similarly to terrains defined as implicit surfaces by considering $f_0 : \mathbb{R}^3 \to \mathbb{R}$ such that $f_0 = 0$ represents the initial surface of the terrain and $f_0 > 0$ signals the interior of this surface. After this change, $f(f_0(\mathbf{x}),\mathbf{x})$ becomes a function that generates a *hypertextured terrain*.

[†] Supported by grant SFRH/BD/16249/2004 from the Fundação para a Ciência e a Tecnologia, Portugal.

The restriction that density modulation functions return opacity values in the range $[0,1]$ is no longer necessary and we can define them as functions $f_i : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}$ for $i = 1, \ldots, n$. The initial surface of the terrain can be a horizontal plane, defined with $f_0(\mathbf{x}) = f_0(x_1, x_2, x_3) = -x_3$, or a sphere of unit radius, defined with $f_0(\mathbf{x}) = 1 - \|\mathbf{x}\|$. The latter case can be used to model procedurally defined planets [Mus03a]

Displacement mapping is the traditional modelling technique for procedural terrains [Coo84, Mus03b]. A generalisation of displacement mapping, that can add overhangs to procedural terrains, is called *flow mapping* [Ped94, GM01]. The terrain is immersed in a procedurally defined vector field and made to follow the streamlines of that field along some specified distance. This allows parts of the terrain to be pushed on top of other parts, thus creating overhangs. The vector field deformation establishes a diffeomorphism between the modified terrain and the original terrain, meaning that the two terrains are topologically equivalent. If the original terrain has no holes, for example, the modified terrain will not have any holes either. The flow mapping method, therefore, cannot generate arches because they imply a topology change in the terrain.

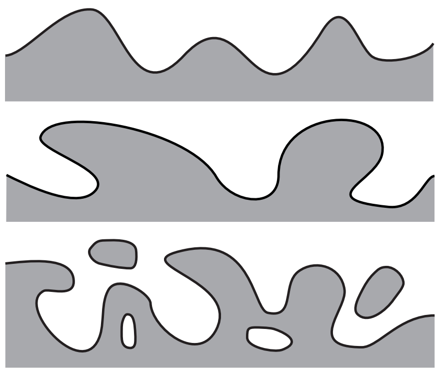

Figure 1: *From top to bottom, comparison between the displacement map, flow map and hypertexturing techniques when adding detail to a horizontal plane.*

Compared to displacement mapping or flow mapping, hypertexturing is a more flexible technique for adding geometric detail to an otherwise smooth terrain. Hypertextures can generate both overhangs and arches. Figure 1 illustrates the difference between the three techniques. Although it is not demonstrated here, any displacement map can also be expressed as a particular form of hypertexture. If f_0 generates a sphere of unit radius, for example, any hypertexture written as $f(f_0(\mathbf{x}), \mathbf{x}/\|\mathbf{x}\|)$ is equivalent to a displacement mapped sphere. Similarly, a flow mapped surface can also be expressed as a hypertexture although the formulation becomes more involved.

One drawback of the modelling flexibility provided by hypertextures is that a terrain can be split into several disconnected fragments, which leads to physically implausible results (this is illustrated in Figure 1). This paper presents a solution to the surface splitting effect of hypertextures that are defined with C^2 continuous functions. Disconnected terrain fragments are detected and removed during rendering. To achieve this goal we rely on Morse theory to analyse the topology of the terrain. The terrain connectivity can be completely determined by studying the critical points of the function f and the way they are linked to each other. We apply our technique as part of a ray casting algorithm for hypertextured terrains. Ray-terrain intersection points are queried for their connectivity state. If an intersection point along a ray is found to be part of a disconnected fragment it is ignored and another intersection is searched further along the same ray. We use a ray casting algorithm for hypertextured implicit surfaces that can find all intersection points between a ray and the terrain, sorted by distance along the ray [GM07a].

We previously presented a topology correction method, based on Morse theory, for hypertextured implicit surfaces [GM07b, GM08]. This method, however, is *global* in the sense that all the critical points of the surface need to be computed as a first step. The critical points are then linked during a second step to determine their connectivity state. Only after these two steps have been performed can the ray casting of the surface begin. This method is impractical for procedural terrains because the number of critical points can be quite large. In the case of a terrain defined over an infinite plane, in particular, the number of critical points is infinite.

Our new method, by contrast, is *local* and integrates the finding and linking of critical points into the ray casting procedure. Critical points are found on demand and only inside a small neighbourhood centred at the current ray-terrain intersection point. The size of the neighbourhood is progressively enlarged, and more critical points are located, until a definite answer can be given about the connectivity state of the intersection point. Critical points are then cached and reused for nearby ray intersection points on the terrain.

Section 2 presents an overview of the concepts from Morse theory that are used by our method, which is explained in Section 3. Section 4 discusses the results that have been obtained and finally Section 5 presents conclusions.

2. Morse Theory for Implicit Surfaces

Morse theory studies the behaviour of functions over a manifold [Mil63]. The theory was first introduced to computer graphics by Shinagawa et al. and was later shown by Hart to be relevant for the topological study of implicit surfaces [SKK91, Har98]. When the theory is applied to implicit surfaces, the manifold becomes the entire $\mathbb{R}^3$ space and the function defined over this space is our function f that generates the terrain. Central to the Morse theory is the notion of a *critical point* of f. A critical point $\mathbf{x}_C$ is such that:

$$\nabla f(f_0(\mathbf{x}_C), \mathbf{x}_C) = \mathbf{0}. \quad (2)$$

A critical point can be further classified by studying the eigenvalues of the Hessian matrix of f at $\mathbf{x}_C$. The Hessian matrix $\mathcal{H}\{f\}$ collects all the second partial derivatives of the function f:

$$\mathcal{H}\{f\} = \left[\frac{\partial f^2}{\partial x_i \partial x_j}\right]_{i,j\in\{1,2,3\}}. \tag{3}$$

If f is C^2 continuous then we have that $\partial f^2/\partial x_i \partial x_j = \partial f^2/\partial x_j \partial x_i$ and the Hessian is symmetric. The spectral theorem then guarantees that all three eigenvalues of $\mathcal{H}\{f\}$ will be real. Depending on the signs of the eigenvalues λ_1, λ_2 and λ_3, sorted in increasing order, a critical point can be classified as shown in Table 1. The type of a critical point gives an indication of the topology of the surface around that point. For example, the maxima occur near the local centroids of the surface while the 2-saddles occur at points where two surface fragments are joined together. In this paper, we only need to be concerned with the maxima and the 2-saddles in order to characterise the connectivity of the surface.

λ_1	λ_2	λ_3	**Type**
$-$	$-$	$-$	Maximum
$-$	$-$	$+$	2-saddle
$-$	$+$	$+$	1-saddle
$+$	$+$	$+$	Minimum

Table 1: *Distinct types of critical points are determined by combinations of the signs of the eigenvalues of the Hessian matrix* $\mathcal{H}\{f\}$.

The case where one or more of the eigenvalues is zero leads to a *degenerate critical point*. Morse theory breaks down in these circumstances. However, degenerate critical points are unstable and can easily be removed by introducing a small perturbation in the parameters defining the function. A function f that contains no degenerate critical points is then said to be a *Morse function*. Morse functions need to be C^2 continuous, considering that both first and second partial derivatives of f are required by the Morse theory. It is possible to relax this restriction and work with C^1 functions, provided that second derivatives are continuous at least over the critical points [HDA98]. Working with C^0 continuous functions, however, is not possible and this precludes the use of Morse theory to perform topology correction on terrains with ridges. A ridge is a part of the terrain where the gradient vector is not defined. This makes it impossible to apply the concept of critical point.

By taking the gradient ∇f, one obtains a vector flow field whose structure is intimately related to the topology of the implicit surface. From equation (2), the critical points of the surface are also the stagnation points of the flow field. A streamline of this field is a path that is obtained by following the local gradient vector, according to the ordinary differential equation:

$$\frac{d\mathbf{x}}{dt} = \nabla f(f_0(\mathbf{x}), \mathbf{x}). \tag{4}$$

All streamlines terminate at maxima of f. A streamline, starting at any point $\mathbf{x}$, can be followed by integrating (4) for $t \to +\infty$. A streamline is called a *separatrix* if it separates two regions of the flow with different characteristics [HH91]. Separatrices are important as they also give information about the topology of the surface. All the separatrices originate and terminate at maxima of f. For every separatrix there is always a 2-saddle somewhere along its path. The separatrix is locally tangent to the $\mathbf{v}_3$ eigenvector (associated with the λ_3 eigenvalue) at the 2-saddle.

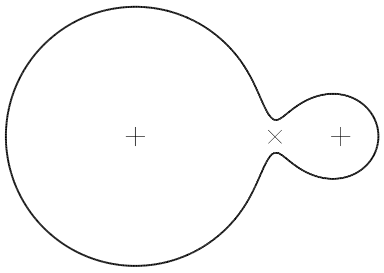

Figure 2: *An implicit surface formed from two blobs. The "+" signs mark the two maxima and the "×" sign marks the 2-saddle.*

Figure 2 shows a simple case of two implicit blobs connected as a single surface. There are two maxima close to the centroids of each blob and a 2-saddle at the junction of the two blobs. The separatrix, in this simple case, is a straight line segment linking the two maxima and passing through the 2-saddle. In a more general situation the separatrix would be curvilinear. Knowing the position $\mathbf{x}_S$ of the 2-saddle, it is possible to locate the two maxima linked with this critical point by integrating equation (4) backwards and forwards from $\mathbf{x}_S$, following a direction that is initially coincident with the $\mathbf{v}_3$ eigenvector of the 2-saddle. It is also possible to determine the connectivity of the two blobs by checking the sign of $f(f_0(\mathbf{x}_S), \mathbf{x}_S)$. If this sign is positive, the blobs are connected and the separatrix is known to travel exclusively through the interior of the surface. If the sign is negative, the two blobs are disconnected and the separatrix must exit and enter the surface again at some points.

The concepts from Morse theory can be applied to the case of terrains expressed as hypertextured implicit surfaces. To know if a terrain fragment is connected to the ground, one needs to find a path formed by interior separatrices that links a maximum of the fragment to another maximum that is part of the ground terrain. If, on the other hand, there is no link

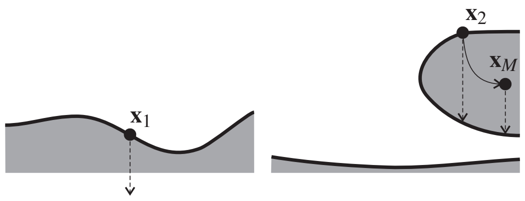

Figure 3: *A quick connectivity test. The point* $\mathbf{x}_1$ *is connected. The point* $\mathbf{x}_2$ *has unknown connectivity.*

between any of the maxima that are located inside the fragment and the ground then the fragment is deemed to be disconnected. The following section explains how this testing is accomplished on a point-wise basis for every ray-terrain intersection point of a ray casting algorithm.

3. Localised Connectivity Testing

The connectivity testing algorithm is a boolean function that receives an arbitrary point $\mathbf{x}$, assumed to be on the surface of the terrain, and returns a result of true if $\mathbf{x}$ is part of the ground or false if it is part of a disconnected fragment. The algorithm begins by performing a quick connectivity test, as illustrated in Figure 3, which consists of tracing a ray from $\mathbf{x}$ towards the interior of the terrain and checking if it results in any intersection. For a terrain defined over a plane, the ray follows a vertical direction and, for a terrain defined over a sphere, it follows the direction towards the centre of the sphere. If no intersection is found, the point $\mathbf{x}$ is known to be part of the ground and the algorithm returns, otherwise more expensive connectivity tests need to be performed. In preparation for those tests, the maximum $\mathbf{x}_M$ that is closest to $\mathbf{x}$ is found by integrating (4). A maximum is tagged as connected or disconnected by tracing another interior ray, as shown on the right side of Figure 3. If there is an intersection, the maximum is tentatively classified as disconnected although this status may change in the course of the algorithm, otherwise the maximum is connected and the algorithm again returns early with a connectivity status of true.

The three-dimensional space is decomposed into a partition of cubic voxels and the search for critical points is performed within each voxel, as explained in Section 3.1. The lateral size of the voxels is given by the characteristic length of the hypertexture. This characteristic length is the average distance between the smallest features of the hypertexture. The algorithm keeps a list of active voxels and a list of active maxima. At this point, the list of maxima is initialised to the single maximum $\mathbf{x}_M$ and the list of voxels is initialised to the voxel that contains this maximum. Figure 4 shows the main loop of the algorithm in pseudo-code.

The algorithm proceeds by removing one voxel at a time from the list of active voxels. The active voxels are kept sorted by their vertical coordinates, for planar terrains, or by

```
if $\mathbf{x}$ is connected return true;
find maximum $\mathbf{x}_M$ reached from $\mathbf{x}$;
if $\mathbf{x}_M$ is connected return true;
initialise list of active maxima $L_M$ with $\mathbf{x}_M$;
initialise list of active voxels $L_V$ with voxel $V(\mathbf{x}_M)$;
while $L_V$ not empty
  remove current voxel $V$ with the lowest altitude;
  find all maxima and 2-saddles of $V$;
  while any 2-saddles of $V$ link with active maxima
    remove a 2-saddle linking with an active maximum
    find maximum $\mathbf{x}_M$ at opposite end of link;
    if $\mathbf{x}_M \notin L_M$
      if $\mathbf{x}_M$ is connected return true;
      add $\mathbf{x}_M$ to $L_M$;
      add voxel $V(\mathbf{x}_M)$ to $L_V$ if $V(\mathbf{x}_M) \neq V_0$;
  for every voxel $V_i \notin L_V$ that is a neighbour of $V$
    if terrain fragment continues into $V_i$
      add $V_i$ to $L_V$;
return false;
```

Figure 4: *The connectivity testing algorithm. The input is an arbitrary point* $\mathbf{x}$ *on the surface of the terrain.*

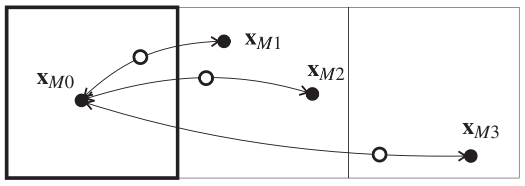

Figure 5: *Examples of linking. The current voxel is on the left. Black circles are maxima. White circles are 2-saddles.*

their distance to the centre of a spherical terrain, for planets. The voxel with the lowest altitude is removed at the start of each iteration and all the critical points contained in the voxel are located. The algorithm then iterates repeatedly over the 2-saddles found inside the voxel. When a 2-saddle links to an active maximum, the maximum at the opposite end of the link, if new, is added to the list of active maxima and the 2-saddle is discarded (this is illustrated in Figure 5 for maximum $\mathbf{x}_{M1}$). The voxel that contains the new maximum is added to the list of active voxels if it is different from the current voxel. The algorithm keeps iterating over the 2-saddles for the current voxel until no more 2-saddles can be linked. Any remaining 2-saddles of the voxel are then ignored since they do not provide any further links to active maxima. If, at any time, a 2-saddle links to a new maximum that is tagged as connected, the algorithm terminates immediately by classifying all the active maxima as connected and returning a connectivity status of true.

One difficulty of the localised topology correction algorithm is that a 2-saddle, linking an active maximum to a new maximum, may reside in a voxel different from the one that is currently being examined. This is illustrated in Figure 5 for the maxima $\mathbf{x}_{M2}$ and $\mathbf{x}_{M3}$. Iterating over the 2-saddles of the current voxel, as previously explained, will not be able to find these new maxima. If no other 2-saddles inside any of the active voxels link with $\mathbf{x}_{M2}$ and $\mathbf{x}_{M3}$ those critical points will not be considered, leading to possible incorrect results. The problem is solved by checking all the neighbours of the current voxel that are not already in the active list. If the terrain fragment that is being examined continues into any of the non-active neighbouring voxels (see Figure 6) those voxels are added to the list of active voxels. This is further explained in Section 3.2.

The search is exhausted when no more active voxels remain in the list. All the active maxima are then known to belong to a disconnected fragment and the algorithm finally returns with a connectivity status of false.

3.1. Locating Critical Points

Critical points inside the current voxel are located with recursive spatial subdivision of the voxel into progressively smaller voxels [SH97]. The existence of a critical point inside each subdivided voxel is tested with an interval arithmetic version of the condition (2) for a critical point [Sny92]. If $\mathbf{X}$ is the three-dimensional interval vector that spans the extent of a voxel, the interval condition is:

$$\nabla f(f_0(\mathbf{X}),\mathbf{X}) \ni \mathbf{0}. \tag{5}$$

We are only interested in the critical points that are located inside the terrain. The following interval condition signals subdivided voxels that lie completely outside the terrain, which can then be removed from the subdivision process:

$$f(f_0(\mathbf{X}),\mathbf{X}) < 0. \tag{6}$$

Subdivision stops once voxels reach a small enough size. A final three-dimensional Newton root finder then locates the exact position of the critical points inside the smallest voxels. More details can be found in the description of the global topology correction method [GM07b, GM08]. The eigenvalues of the Hessian matrix (3) are computed at the critical points and only the maxima and the 2-saddles are kept for further processing.

3.2. Checking Neighbouring Voxels

Before completing the processing of the current voxel in the algorithm of Figure 4, it is necessary to add those neighbouring voxels that share part of the terrain into the list of active voxels. It is only necessary to consider voxels that share the same terrain fragment of point $\mathbf{x}$, for which the connectivity test was invoked. Figure 6 shows an example – the voxel on the left needs to be added to the list but the voxel on the right needn't. One can detect if the terrain fragment continues into any of the 26 neighbouring voxels by checking for the existence of maxima of f restricted to the subdomains consisting of the 2D faces, 1D edges and 0D corner points of the current voxel.

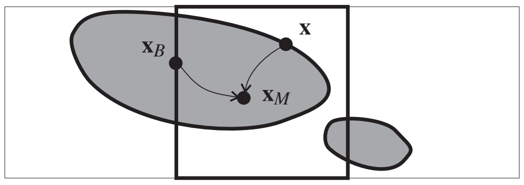

Figure 6: *Example of edge neighbour checking. The current voxel is in the centre. The algorithm is determining the connectivity of point* $\mathbf{x}$ *on the larger fragment.*

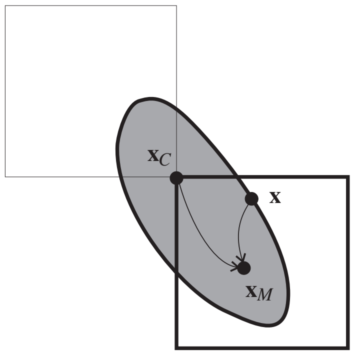

Figure 7: *Example of corner neighbour checking. The current voxel is in the lower right. The other voxels sharing the corner point* $\mathbf{x}_C$ *are treated as in Figure 6.*

In the case of face neighbours, a recursive subdivision of the boundary face is performed, followed by a two-dimensional Newton root finder to locate the maxima. This is analogous to what is described in Section 3.1 for the three-dimensional case. The Hessian matrix of f restricted to the face of a voxel has a dimension of 2×2 and a maximum is a critical point where the two eigenvalues λ_1 and λ_2 of the Hessian are both negative. The situation is similar for edge neighbours with subdivision and root finding working over one dimension. Along the edge of a voxel, the Hessian matrix degenerates into a single partial derivative $\partial^2 f/\partial x_i^2$ for i equal to 1, 2 or 3, depending on which coordinate axis the edge is parallel to. A maximum along the edge is then characterised by $\partial f/\partial x_i = 0$, which is detected by the Newton root finder, and $\partial^2 f/\partial x_i^2 < 0$. For corner neighbours one only needs to verify that the corner point $\mathbf{x}_C$ is inside the terrain by checking the condition $f(f_0(\mathbf{x}_C),\mathbf{x}_C) > 0$. This is illustrated in Figure 7. Corner neighbours whose corner point, relative to the current voxel, is outside the terrain are not considered.

For each maximum that is found along one of the borders between the current voxel and a neighbour, it is necessary to check if this boundary maximum belongs to the terrain fragment being examined. For that purpose, one tracks a streamline starting from the boundary maximum and checks if it converges towards one of the active maxima. Figure 6 shows an example where $\mathbf{x}_B$ is a boundary maximum relative to the left neighbour voxel and $\mathbf{x}_M$ is one of the active maxima for the terrain fragment. The path of the streamline is followed by integrating (4). If the streamline terminates at one of the active maxima of the fragment then the neighbour voxel also contains the fragment. At this point, no further maxima along the common border between the two voxels needs to be considered and the neighbour can be added to the list of active voxels. If no boundary maximum links with active maxima then the terrain that is shared between the two voxels is not part of the terrain fragment being examined for connectivity. The situation is similar for corner neighbours (Figure 7) where a streamline is followed from the corner point $\mathbf{x}_C$ to see if it terminates at an active maximum.

When going from the current voxel V to a neighbouring voxel V_i through the boundary maximum $\mathbf{x}_B$, it is necessary to record the triple $(V, V_i, \mathbf{x}_B)$. If V is revisited during a later iteration of the algorithm (by following new links that may have since been discovered), the transition to V_i via $\mathbf{x}_B$ will not be performed again if it has been done before. This is to prevent the possibility of the algorithm becoming locked in an infinite loop. A transition from V to V_i is still possible if done through some other boundary maximum different from $\mathbf{x}_B$. In the case of corner neighbours, it is only necessary to record the pair (V, V_i) since the corner point $\mathbf{x}_C$ is unique. The transition from a voxel to one of its corner neighbours, therefore, can only be done once for each invocation of the connectivity test algorithm.

3.3. Caching

Caching is essential for the efficiency of the algorithm. Without caching many of the operations previously described would be needlessly repeated for points $\mathbf{x}$ on the surface of the terrain that happened to be in close proximity. There are several levels of cache that can be implemented: caching of maxima, caching of critical points per voxel and caching of boundary maxima per face or edge.

Active maxima are sent to the cache whenever their connectivity status becomes known. This happens when the algorithm of Figure 4 returns with a connectivity result of either true or false. When the algorithm links to a new maximum, it always checks first to see if the new maximum is in the cache and returns early if true. The return value is the connectivity status of the new maximum in the cache.

All the voxels that have been processed in previous invocations of the algorithm are also kept in another cache together with their maxima and 2-saddles. This avoids having to locate again the critical points for those voxels with recursive subdivision. The algorithm merely retrieves from the cache the list of maxima and 2-saddles of a voxel that was previously computed as part of the connectivity test for a different point on the terrain. A similar caching mechanism is implemented for the boundary maxima of the faces and edges of the voxel space decomposition.

Caching is also useful for computer animations where the camera moves over the terrain. The caches can be kept when rendering the consecutive frames of an animation. To prevent the caches from growing without bounds, however, it is necessary to remove those cache elements that have not been used for some specified number of frames.

4. Results

Figure 8 is a rendering of the surface of a procedural planet with overhangs and arches, modelled as a hypertextured implicit surface. Although the terrain appears to be defined over a flat surface, it is actually a sphere seen from a very close range. The function defining the sphere is $f_0(\mathbf{x}) = 1 - \|\mathbf{x}\|$. The hypertexture combines two procedural noise functions:

$$f(f_0(\mathbf{x}), \mathbf{x}) = f_0(\mathbf{x}) + 2 \cdot 10^{-4} g(4 \cdot 10^4 \mathbf{x}) s^2(10^4 \mathbf{x}/\|\mathbf{x}\|). \quad (7)$$

A gradient noise function $g(\mathbf{x})$ provides the basic terrain pattern and is then modulated by a squared sparse convolution noise function $s(\mathbf{x})$ to create the appearance of rocky outcrops over an otherwise flat terrain [Per02, Lew89]. The modulating noise is expressed in the form $s(\mathbf{x}/\|\mathbf{x}\|)$ so that it is made to depend only on the position over the surface of the planet but not on the height above the same surface. In order to apply Morse theory to any hypertexturing function, expressions need to be available to evaluate the function's gradient vector and Hessian matrix. Such expressions for the two noise functions used can be found in [GM08].

The detection of surface connectivity is shown in the middle image of Figure 8 with the disconnected surface fragments coloured in green. The topology correction is then shown in the bottom image. It is possible to see that the shadows cast on the ground by disconnected fragments, which are visible in the lower left corner of the top and middle images, have disappeared in the bottom image due to those surface fragments having been removed. This effect is achieved by performing connectivity testing for shadow rays, similar to what is done for view rays. Disconnected fragments are ignored for shadow rays and a point is only in shadow if its shadow ray intersects with the ground terrain. The images in Figure 8 have a resolution of 1368×828 and took, from top to bottom, 354 minutes, 516 minutes and 685 minutes to render on a Pentium4 2.8GHz. These numbers give an overhead of 46% for connectivity testing relative to pure ray casting and a 94% overhead for full topology correction.

Figure 9 shows the same terrain of Figure 8 with the viewpoint placed at two successively higher altitudes. The atmosphere has been removed so as to see more of the terrain

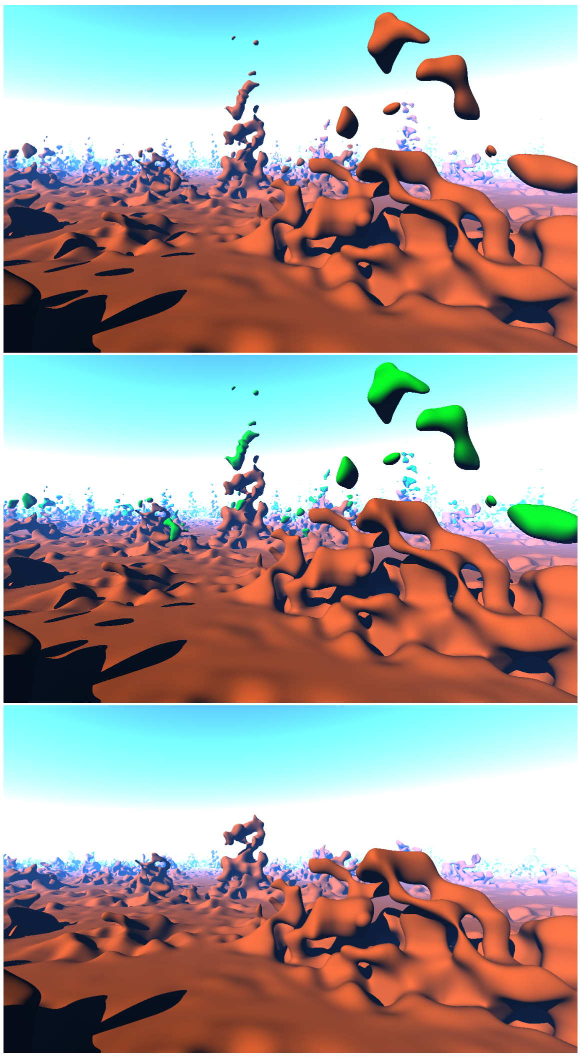
Figure 8: *A hypertextured planet featuring terrain overhangs and arches. The top image shows the original surface. The middle image shows disconnected components in green. The bottom image shows the topological correction.*

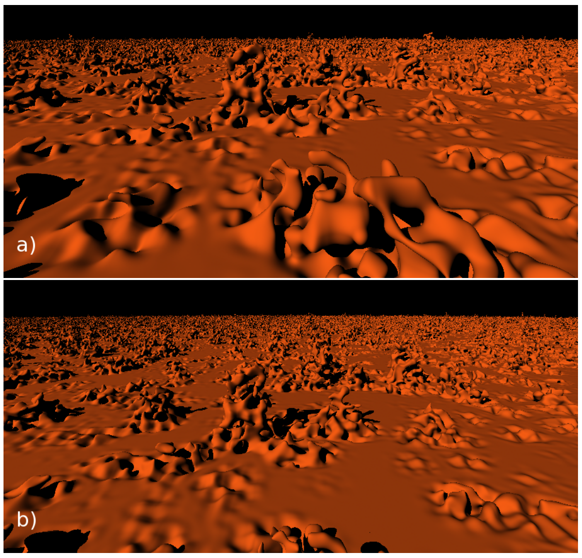

Figure 9: *The same terrain of Figure 8 without atmosphere and seen from two higher altitudes.*

over long distances. Table 2 gives some statistics relative to the images of Figures 8 and 9. The statistics shown are the number of maxima (# Maxima) and 2-saddles (# Saddles) located, the number of maxima placed in the cache (Cache Size), the cache hit rate for the maxima (Cache Hits), the percentage of success of the initial quick connectivity test (Quick Test), the percentage of time spent in ray tracing (Tracing), which includes the time spent in tracing shadow rays and connectivity test rays, the percentage of time spent in integrating streamlines (Streaming) and the percentage of time spent in locating critical points with recursive subdivision (Locating).

Table 2 shows that the statistics of the topology correction algorithm generally degrade as the altitude of the viewpoint increases and a larger area of the terrain becomes visible. The number of critical points located by the algorithm increases accordingly. The number of cached maxima is generally smaller than the number of located maxima because, when locating critical points inside a voxel, some maxima belong to terrain fragments that are never tested. These terrain fragments are missed by the ray caster due to the finite sampling density. The performance of the cache also decreases as more disconnected fragments need to be removed. The performance of the quick connectivity test, used at the beginning of the algorithm, is the only statistic that improves as there is more flat terrain visible on the lower part of the images with increasing altitude.

Even with the high hit rates for the cache, shown in Table 2, the location of critical points can easily become the bottleneck of the topology correction algorithm. As the number of critical points increases, the time spent in this operation becomes larger than the time spent in either tracing rays or following streamlines. Similarly to the other statistics, this imbalance becomes more pronounced as a larger area of the terrain becomes visible.

5. Conclusions

The topology correction method that has been presented detects and removes disconnected terrain fragments, turning any C^2 continuous hypertextured implicit surface into a topologically correct terrain. The method uses a localised connectivity test that is meant to be used in conjunction with a

	# Maxima	# Saddles	Cache Size	Cache Hits	Quick Test	Tracing	Streaming	Locating
Fig. 8	5901	5742	5423	98.67%	58.24%	38.12%	25.03%	36.85%
Fig. 9a)	20534	21242	18650	95.38%	59.30%	20.57%	14.02%	65.42%
Fig. 9b)	31607	33440	28611	90.57%	68.34%	14.57%	9.12%	76.31%

Table 2: *Several statistics for the images of Figures 8 and 9. Refer to the text for the meaning of these statistics.*

ray casting algorithm for rendering the terrain. The localised nature of the connectivity test only requires analysing the terrain inside a neighbourhood around each surface point.

Due to the use of Morse theory, the method is robust and can remove all disconnected fragments that are intersected by the ray casting algorithm, even when these fragments are very small or very close to the ground. The localised topology correction method has a higher implementation complexity than the global method that was previously developed. This higher complexity allows the handling of virtually infinite terrains, something that the global topology correction method could not do.

Finally we should note that although the method performs a topological analysis of the terrain features, it does not perform a stability analysis of those same features. A method that can analyse both issues of topology and stability under the action of gravity is a possible area of future research.

References

[Coo84] COOK R. L.: Shade trees. In *Computer Graphics (SIGGRAPH '84 Proceedings)* (1984), Christiansen H., (Ed.), vol. 18, ACM Press, pp. 223–231.

[GM01] GAMITO M. N., MUSGRAVE F. K.: Procedural terrains with overhangs. In *Proc. of the 10th Eurographics Portuguese Chapter Conference* (2001), Grupo Português de Computação Gráfica, pp. 33–42.

[GM07a] GAMITO M. N., MADDOCK S. C.: Ray casting implicit fractal surfaces with reduced affine arithmetic. *The Visual Computer 23*, 3 (Mar. 2007), 155–165. A correction to this paper has been made, which is available at http://www.dcs.shef.ac.uk/~mag/raycast.html.

[GM07b] GAMITO M. N., MADDOCK S. C.: Topological correction of hypertextured implicit surfaces for ray casting. In *IEEE International Conference on Shape Modelling and Applications (Proc. SMI '07)* (2007), Galin E., Pauly M.„ Wyvill B., (Eds.), IEEE Press, pp. 103–112.

[GM08] GAMITO M. N., MADDOCK S. C.: Topological correction of hypertextured implicit surfaces for ray casting. *The Visual Computer* (2008). Revised selected paper from the SMI '07 Conference. to appear.

[Har98] HART J. C.: Morse theory for implicit surface modeling. In *Mathematical Visualization*, Hege H.-C., Polthier K., (Eds.). Springer Verlag, Heidelberg, 1998, pp. 257–268.

[HDA98] HART J. C., DURR A., ARSCH D.: Critical points of polynomial metaballs. In *Proc. Workshop on Implicit Surfaces* (June 1998), Eurographics/SIGGRAPH, pp. 69–76.

[HH91] HELMAN J. L., HESSELINK L.: Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications 11*, 3 (May 1991), 36–46.

[Lew89] LEWIS J.-P.: Algorithms for solid noise synthesis. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (July 1989), Lane J., (Ed.), vol. 23, ACM Press, pp. 263–270.

[Mil63] MILNOR J.: *Morse Theory*, vol. 51 of *Annals of mathematics studies*. Princeton University Press, Princeton, 1963.

[Mus03a] MUSGRAVE F. K.: *Mojoworld: Building Procedural Planets*, 3rd ed. Morgan Kauffman Publishers Inc., 2003, ch. 20, pp. 565–615.

[Mus03b] MUSGRAVE F. K.: *Procedural Fractal Terrains*, 3rd ed. Morgan Kauffman Publishers Inc., 2003, ch. 16, pp. 489–506.

[Ped94] PEDERSEN H. K.: Displacement mapping using flow fields. In *Computer Graphics (SIGGRAPH '94 Proceedings)* (July 1994), Glassner A., (Ed.), vol. 28, ACM Press, pp. 279–286.

[Per02] PERLIN K.: Improving noise. *ACM Transactions on Graphics (SIGGRAPH '02 Proceedings) 21*, 3 (July 2002), 681–682.

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (July 1989), Lane J., (Ed.), vol. 23, ACM Press, pp. 253–262.

[SH97] STANDER B. T., HART J. C.: Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Computer Graphics (SIGGRAPH '97 Proceedings)* (Aug. 1997), Whitted T., (Ed.), vol. 31, ACM Press, pp. 279–286.

[SKK91] SHINAGAWA Y., KUNII T. L., KERGOSIEN Y. L.: Surface coding based on morse theory. *IEEE Computer Graphics and Applications 11*, 5 (Sept. 1991), 66–78.

[Sny92] SNYDER J. M.: Interval analysis for computer graphics. In *Computer Graphics (SIGGRAPH '92 Proceedings)* (July 1992), Catmull E. E., (Ed.), vol. 26, ACM Press, pp. 121–130.