Multiresolution control of curves and surfaces with a self-similar model

Houssam Hnaidi, Eric Guérin and Samir Akkouche

LIRIS - Université Claude Bernard Lyon 1

May 28, 2009

Abstract

This paper presents two self-similar models that allow the control of curves and surfaces. The first model is based on IFS (Iterated Function Systems) theory and the second on subdivision curve and surface theory. Both of these methods employ the detail concept as in the wavelet transform and allow the multiresolution control of objects with control points at any resolution level.

In the first model, the detail is inserted independently of control points, requiring it to be rotated when applying deformations. On the contrary, the second method describes details relative to control points, allowing free control point deformations.

Modelling examples of curves and surfaces are presented, showing manipulation facilities of the models.

1 Introduction

Controlling generated objects has always been important in computer graphics, to allow easy modelling and fast deformation processes. Free forms (Bézier curves, splines, NURBS) are pioneers in this domain, allowing the user to control them using control points.

To expand this concept at different scales, the concept of multiresolution has appeared. Subdivision surfaces [1] have successfully introduced a model capable of multiresolution editing. Based on this principle, a great deal of research has been conducted to model or control curves and surfaces with a compression goal, for instance.

In addition, wavelet theory has been introduced in signal processing, including for compression purposes. This theory uses the principle of multiresolution decomposition of a signal into a low-resolution version that combines it with detail information. This decomposition principle can be applied to subdivision theory for compression purposes but also for multiresolution editing.

Subdivision surfaces and wavelet representations are well adapted to smooth shapes. When natural and rough objects need to be represented, however, the use of fractal models is recommended. Unfortunately, fractal models are difficult to control. Some have attempted to transpose the free form concept to fractals [2]. To the authors' knowledge, no fractal model permits the multiresolution control of the generated object.

In this paper, two self-similar (intrinsically fractal) models that allow the multiresolution control of the generated curves and surfaces are presented. The first one is based on IFS (Iterated Function Systems) theory, in particular projected IFS. The second model employs the subdivision principle used in subdivision surfaces. In both cases, the detail concept similar to the concept employed in wavelet theory is used.

This article first presents a review of the related work, then we introduce our contribution and the results for the two models.

2 Related Work

Bézier curves, B-splines and NURBS are used intensively in computer graphics when global control over a curve or a surface is needed. These models cannot be directly applied if the user wishes to modify the shape at different scales.

Several models have been proposed to satisfy this need. Finkelstein *et al.* [3] introduced a multiresolution representation of curves based on wavelets. This representation can support several operations on curves: smoothing, detail-preserving deformation and tolerance-based approximation. Elber [4] introduced a system that allows the multiresolution control over a linearly constrained NURBS curve.

Biermann *et al.* [5] proposed a cut-and-paste method based on a multiresolution subdivision. It allows the partial transfer of one surface on another.

In [6], a method localising the refinement effect is presented. This method uses a controlled hierarchical subdivision and aims at manipulating a surface from editing points.

In [7], a method allowing the user to cut-and-paste one surface on another at any hierarchical level is presented, making it possible to add details in a surface without increasing its complexity.

[8] presents a new interactive mesh editing approach. Large meshes are first simplified, then subdivided into sub-meshes interactively. Each sub-mesh can be edited with geometrical transformations. Finally, the mesh is reconstructed entirely.

[9] introduces a curvature based multiresolution representation for 2-D polygonal curves. It represents all the detail coefficients by lengths and angles in order to preserve the orientation of the details during deformation.

All of these models have a common feature: they are designed to treat smooth objects. In nature, objects are not smooth. They can be represented by fractal models [10, 11], which are usually split into two categories: stochastic and deterministic models. Only the latter can be potentially used in a context where the user seeks con-

trol. L-systems [12] and IFS [13] are examples of such models.

IFS is a powerful tool for analysis and synthesis of fractal objects. Zaïr [2] introduced a variant of this model that could control curves and surfaces interactively, called projected IFS. This model is an adaptation of free forms. In classical free forms, blending functions are made of polynomials, whereas in projected IFS, they are computed with IFS. Guérin [14,15] employed this model to approximate curves and surfaces with a distance minimisation formalism.

Several studies have used constraints to control the generation of self-similar objects. Blanc-Talon [16] introduced an interpolation system with spline-based approximation. Coefficients of subdivision matrices are computed from global and local geometrical constraints. This process can generate fractal curves whose fractal dimension is reproduced from an initial given curve. Belhadj [17] introduced an algorithm based on fractals that can generate DEM (digital elevation maps) that conform to constraints. The user can choose global features and morphology, local details and can see the reconstruction interactively. Stachniak [18] presented a method that employs a stochastic search to identify local modifications. These modifications can deform the fractal terrain and conform to a set of constraints. The results show that the method can integrate multiple constraints while preserving the natural aspect of the terrain.

3 Contribution

The goal of this paper is to propose a model that can combine a fractal shape with a detail concept such as that used in wavelet transform [19,20]. To this end, we introduce a general formalism that leads to two approaches. The first approach is based on projected IFS, whereas the second uses the subdivision concept. The following presents this general formalism:

$$\mathcal{P}^{J+1} = \varphi(\mathcal{P}^J) \oplus \psi(\mathcal{P}^J, \delta\mathcal{P}^J) \tag{1}$$

where $\mathcal{P}^J$ represents the object at the resolution level J and $\delta\mathcal{P}^J$ is the detail associated with this level. This operation is called multiresolution synthesis (see Fig. 1). With this formalism, an object $\mathcal{P}^N$ can be represented by its version at a given resolution level J and associated details $\delta\mathcal{P}^J, \ldots, \delta\mathcal{P}^{N-1}$. When successively combined, the resolution N can be reconstructed.

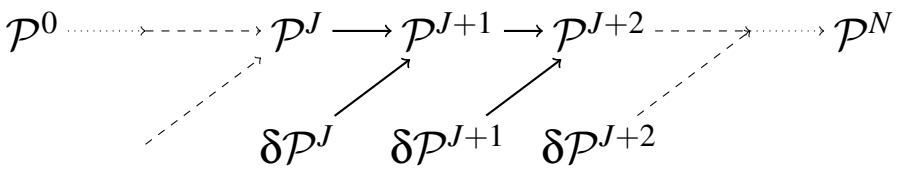


Figure 1: Multiresolution synthesis

Each term $\mathcal{P}^J$, $\delta\mathcal{P}^J$, $\varphi(\mathcal{P}^J)$, $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ and $\oplus$ will be defined in each approach.

In both approaches, we will introduce an invertible formula, able to perform the so-called analysis operation (see Fig. 2). Given an object at a resolution level N, analysis decomposes it into a multiresolution representation combining detail information $\delta\mathcal{P}^0, \ldots, \delta\mathcal{P}^{N-1}$ and the low- resolution version $\mathcal{P}^0$.

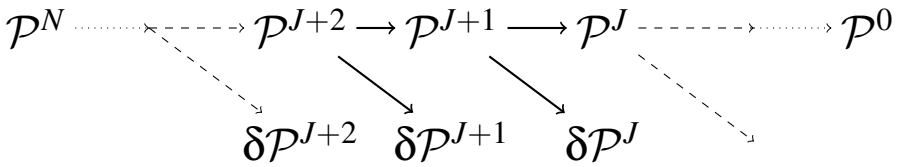


Figure 2: Multiresolution analysis

3.1 An Approach Based on Projected IFS

First, classical IFS theory definitions and projected IFS definitions will be reviewed.

3.1.1 IFS

IFS theory can be defined as follows.

Let $(\mathbb{X}, d)$ be a metric space. The function $f : \mathbb{X} \to \mathbb{X}$ is called contraction mapping if and only if there is some real number $0 < s < 1$ such that $\forall x, y \in \mathbb{X} : d(f(x), f(y)) < sd(x, y)$.

Let $(\mathbb{X}, d)$ be a complete metric space. An IFS is a finite set $\mathcal{T} = \{T_0, ..., T_{N-1}\}$ of contractions on $\mathbb{X}$ [13].

To each IFS $\mathcal{T}$ is associated a unique non-empty compact A of $(\mathbb{X}, d)$ such that:

$$\begin{aligned} A &= \mathcal{T}A \\ &= T_0A \cup \ldots \cup T_{N-1}A \end{aligned}$$

A is called the attractor of $\mathcal{T}$ and is denoted $A(\mathcal{T})$, T_i belongs to a contractive function class.

$\Sigma = \{0, ..., N-1\}$ is called the alphabet associated with $\mathcal{T}$.

Let Σ^* be the set of finite words of Σ and Σ^w the set of infinite words of Σ. We can state (see Barnsley [13]):

Let $\theta = \theta_1\theta_2\theta_3 \ldots \in \Sigma^w$. For all $\lambda \in \mathbb{X}$ the following limit:

$$\lim_{j\to\infty} T_{\theta_1} \ldots T_{\theta_j}\lambda$$

is defined and is independent of λ.

We can now define an address function ϕ that maps from an infinite word θ of Σ to a point of the attractor:

$$\begin{aligned} \phi : \Sigma^\omega &\to \mathbb{X} \\ \theta &\mapsto \phi(\theta) = \lim_{j\to\infty} T_{\theta_1} \ldots T_{\theta_j}\lambda \end{aligned}$$

where λ can be any value in $\mathbb{X}$ and $\theta = \theta_1 \ldots \theta_j \ldots,$.

An example of IFS is the Cantor set (Fig 3). We can generate this set by using two

contraction mappings $T_0 = \frac{1}{3}x$ and $T_1 = \frac{1}{3}x + \frac{2}{3}$ where the metric space is defined by $\mathbb{X} = [0,1]$ and d the euclidean distance.

Figure 3: Seven iterations of the Cantor set IFS.

3.1.2 Projected IFS

The projected IFS is a model derived from the free-form model (like Bézier curves) where both fundamental notions of control points and blending functions are present. To build fractal blending functions, barycentric space and coordinates are used.

Let $\mathbb{J}$ be an index set for example $\{0,1,2,3\}$. From now on the space $\mathbb{X}$ is a barycentric space associated with $\mathbb{J}$ and is defined by:

$$\left\{ (\lambda_j)_{j\in\mathbb{J}} | \sum_{j\in\mathbb{J}} \lambda_j = 1 \right\}$$

where $\lambda_j \in \mathbb{R}$. For example $\lambda = (0.5, 0.2, 0.2, 0.1)$ belongs to the barycentric space associated with $\mathbb{J} = \{0,1,2,3\}$.

We now have to set an iteration semi-group that operates on this barycentric space. The simplest solution consists in using barycentric column matrices. A matrix T has a barycentric columns if:

$$\sum_{j\in\mathbb{J}} T_{ij} = 1, \forall i \in \mathbb{J}$$

Let $\mathcal{P} = (p_j)_{j\in\mathbb{J}}$ be a control polygon. Fig (4) shows an example of such a control polygon with $\mathbb{J} = \{0,1,2,3\}$.

This choice makes it possible to project the IFS attractor $A(\mathcal{T})$ by means of the

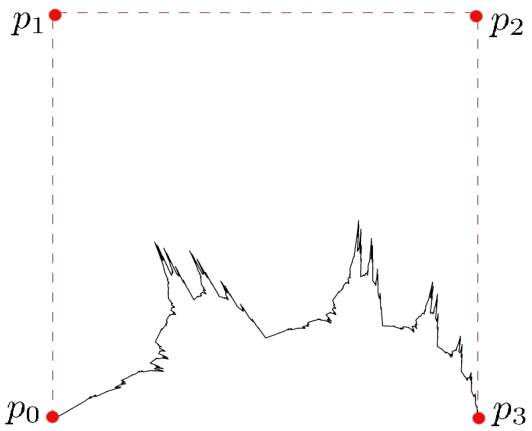


Figure 4: Projected IFS example with its control polygon.

control polygon:

$$\begin{aligned} \mathcal{PA}(\mathcal{T}) &= \{\mathcal{P}\lambda \,|\, \lambda \in \mathcal{A}(\mathcal{T})\} \\ &= \{\mathcal{P}\phi(\theta) \,|\, \theta \in \Sigma^{\omega}\} \end{aligned}$$

where $\mathcal{P}\lambda$ is the projection of a barycentric point λ through a control polygon $\mathcal{P}$:

$$\mathcal{P}\lambda = \sum_{j \in \mathbb{J}} \lambda_j p_j$$

.

3.1.3 Detail Insertion

In this section, we present a version of Eq. (1) that makes use of projected IFS.

Let us define the following series, which is used to construct and display projected

IFS:

$$(\mathcal{S}_n)_{n\in\mathbb{N}} = \begin{cases} \mathcal{S}_0 & = \{\mathcal{P}\} \\ \mathcal{S}_{n+1} & = \mathcal{S}_n\mathcal{T}, \forall n \in \mathbb{N} \end{cases}$$

$\mathcal{S}_n$ is a finite set of polygons that can be constructed recursively as a tree:

$$\mathcal{S}_n = \mathcal{P}\mathcal{T}^n = \left\{\mathcal{P}T_{\theta_1}\dots T_{\theta_n} \,|\, |\theta| = n\right\}$$

where $|\theta|$ is the length of the word θ.

Let us denote $T_\theta = T_{\theta_1}\dots T_{\theta_n}$ and $\mathcal{P}_\theta = \mathcal{P}T_\theta$. It can be stated that:

$$\begin{aligned} \mathcal{P}_{\theta i} &= \mathcal{P}T_\theta T_i \\ &= \mathcal{P}_\theta T_i \qquad \text{where} \qquad i \in \Sigma \end{aligned}$$

Inspired by the work of Tosan [21], a detail can be added to the formula $\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i$ such that:

$$\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i + \delta\mathcal{P}_\theta U_i \quad \text{where} \quad i \in \Sigma \tag{2}$$

where $\delta\mathcal{P}_\theta U_i$ is an ordered list of displacement vectors obtained by multiplying the detail vector $\delta\mathcal{P}_\theta$ with a matrix U_i that we call the detail displacement matrix.

We now need to define the addition $(+)$ between a polygon and an ordered list of vector.

Definition 1 ***(displacement of a polygon with an ordered list of vectors)*** *Let P be a polygon and Q be an ordered list of vectors such that the number of vertices n of P equals to the number of vectors of Q, let this number be n. $P+Q$ is defined as a new polygon having n vertices and calculated as follows:*

$$P + Q = \{P_i + Q_i\}_{i=0,\dots,n-1}$$

where $P_i + Q_i$ is a point whose coordinates are the sum of each coordinate of P_i and Q_i. (see Fig. 5)

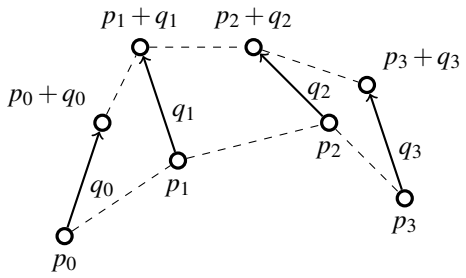


Figure 5: Displacement of a polygon.

We can now define $\mathcal{P}^{J+1}$, $\mathcal{P}^J$, $\varphi(\mathcal{P}^J)$, $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ and $\oplus$ in Eq. (1) for this model.

We set $\mathcal{P}^J = \{\mathcal{P}_\theta\}_{|\theta|=J,\, \theta\in\Sigma^\star}$, then $\mathcal{P}^{J+1} = \{\mathcal{P}_{\theta i}\}_{|\theta|=J,\, i\in\Sigma,\, \theta\in\Sigma^\star}$. Here we consider that $\mathcal{P}^J$ represents an ordered set of polygons, for instance if $J=2$ and $\Sigma=\{0,1\}$ then $\mathcal{P}^2 = (\mathcal{P}_\theta)_{\{|\theta|=2\}} = (\mathcal{P}_{00}, \mathcal{P}_{01}, \mathcal{P}_{10}, \mathcal{P}_{11})$.
$\varphi(\mathcal{P}^J)$ is defined by:

$$\varphi(\mathcal{P}^J) = \{\mathcal{P}_\theta T_i\}_{|\theta|=J,\, i\in\Sigma,\, \theta\in\Sigma^\star}$$

and we define $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ as:

$$\psi(\mathcal{P}^J, \delta\mathcal{P}^J) = \{\delta\mathcal{P}_\theta U_i\}_{|\theta|=J,\, i\in\Sigma,\, \theta\in\Sigma^\star}$$

we finally define $\oplus$ as:
Let $(P_i)_{i=0,\ldots,m-1}$ be an ordered set of m polygons and $(Q_i)_{i=0,\ldots,m-1}$ be an ordered set of m ordered lists of vectors. We define $\oplus$ by:

$$(P_i)_{i=0,\ldots,m-1} \oplus (Q_i)_{i=0,\ldots,m-1} = (P_i + Q_i)_{i=0,\ldots,m-1}$$

where $P_i + Q_i$ is the displacement of polygon P_i with an ordered list of vectors Q_i as defined in definition (1).

We can now write:

$$(\mathcal{P}_{\theta i}) = (\mathcal{P}_\theta T_i) + (\delta\mathcal{P}_\theta U_i) \text{ where } |\theta| = J,\ i \in \Sigma,\ \theta \in \Sigma^\star$$

If we consider a n vertices polygon, N transformations, and each vertex is defined in $\mathbb{R}^d$, then the dimension of matrix T_i is $n \times n$, U_i is $n.(N-1) \times n$, $\mathcal{P}_\theta$ is $d \times n$, and $\delta\mathcal{P}_\theta$ is $d \times n.(N-1)$.

The matrix dimensions are chosen such that an invertible system will be obtained.

When considering the formula $\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i + \delta\mathcal{P}_\theta U_i$ for all $i \in \Sigma$ we have:

$$\begin{array}{rcl} \mathcal{P}_{\theta 0} & = & \mathcal{P}_\theta T_0 + \delta\mathcal{P}_\theta U_0 \\ \vdots & \vdots & \vdots \\ \mathcal{P}_{\theta N-1} & = & \mathcal{P}_\theta T_{N-1} + \delta\mathcal{P}_\theta U_{N-1} \end{array}$$

We can now write the previous formulas in a matrix formalism:

$$(\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1}) = (\mathcal{P}_\theta|\delta\mathcal{P}_\theta) \left(\begin{array}{c|c|c} T_0 & \ldots & T_{N-1} \\ \hline U_0 & \ldots & U_{N-1} \end{array}\right)$$

if we set:

$$R = \left(\begin{array}{c|c|c} T_0 & \ldots & T_{N-1} \\ \hline U_0 & \ldots & U_{N-1} \end{array}\right)$$

then the formula becomes:

$$(\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1}) = (\mathcal{P}_\theta|\delta\mathcal{P}_\theta)\,R$$

The so-called inverse formula is:

$$(\mathcal{P}_\theta|\delta\mathcal{P}_\theta) = (\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1})\,R^{-1}$$

The matrix R can be compared to a synthesis filter and R^{-1} is similar to an analysis filter used in the wavelet transform (Figs. 1-2).

3.1.4 Results

We used this model to apply transformations on curves. To achieve this, the model (the R matrix) has been optimised with respect to the curve using a non-linear method (Levenberg-Marquardt [22]) to minimise the distance between the original curve and the curve generated with our model.

Figure (7) shows an example of the deformation of a curve with a four control points and two transformations model. The intermediate steps are:

- optimise matrices T_i and U_i ($i \in \{0,1\}$ in this case) with respect to the curve; (Fig. 6 shows the T_0,T_1,U_0 and U_1 optimisation result)
- analyse the curve: obtain the control polygon and detail coefficients;
- move a control point and compute the associated affine transformation $\mathcal{M}$;
- extract $\mathcal{L}(\mathcal{M})$, the linear part (rotation and scaling) of $\mathcal{M}$;
- apply $\mathcal{L}(\mathcal{M})$ on the detail vectors;
- display the deformed curve.

$$T_0 = \begin{pmatrix} +0.92 & +0.46 & +0.27 & +0.16 \\ +0.32 & +1.31 & +1.12 & +0.79 \\ -0.22 & -0.74 & -0.47 & +0.11 \\ -0.02 & -0.03 & +0.08 & -0.06 \end{pmatrix} \quad \text{(a)}$$

$$T_1 = \begin{pmatrix} -0.05 & -0.08 & -0.19 & -0.24 \\ +0.32 & -0.21 & -0.40 & +0.16 \\ +0.74 & +0.79 & +0.88 & -0.06 \\ -0.01 & +0.50 & +0.71 & +1.14 \end{pmatrix} \quad \text{(b)}$$

$$U_0 = \begin{pmatrix} +1.48 & -1.21 & +1.06 & -0.80 \\ -0.18 & -0.20 & +2.02 & -2.20 \\ -0.66 & +0.24 & +0.48 & -0.38 \\ +1.44 & -0.61 & +1.29 & +0.17 \end{pmatrix} \quad \text{(c)}$$

$$U_1 = \begin{pmatrix} +0.53 & -0.67 & -1.05 & +0.23 \\ -0.18 & -0.32 & +0.05 & -0.96 \\ +1.23 & -1.43 & +1.49 & -1.25 \\ +0.09 & -0.73 & +1.62 & -1.27 \end{pmatrix} \quad \text{(d)}$$

Figure 6: Transformation and detail displacement matrices.

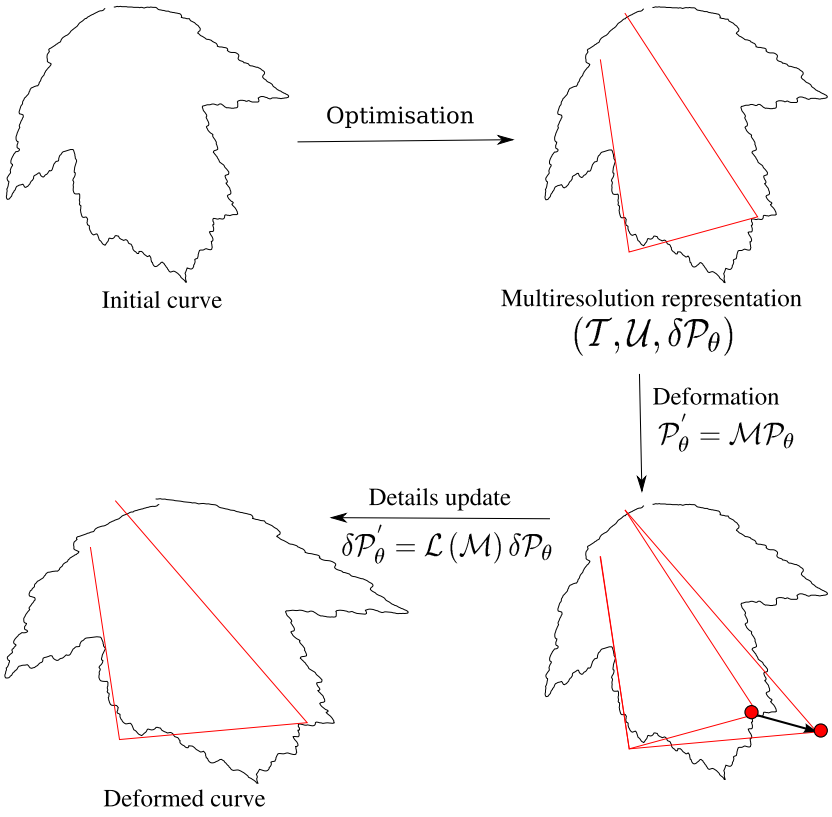


Figure 7: Global deformation steps.

This approach is a simple way to perform the analysis and synthesis but has the disadvantage of requiring the optimisation step. This step has a high computing cost, because of the non-linear minimisation. This cost is even higher when working with surfaces. This model presents a hierarchical representation of the object with details, but local deformations are not possible. Another approach that addresses these drawbacks was therefore used.

3.2 An Approach Based on Subdivision

The approach introduced in this section can perform local deformations with no optimisation cost. This approach is based on the subdivision surface principle [1, 23]. Subdivision surfaces are used in many research fields: surface editing [24], surface approximation [25], *etc.* Adding details to subdivision surfaces has already been addressed for compression purposes [26] and multiresolution editing [3, 4].

We use the subdivision surface with the detail principle for multiresolution editing of curves and surfaces. The novelty of this approach is that several masks are employed to gain in control over the curve or the surface: these masks can be fractal.

Here, two similar models are presented: one for curves and another for surfaces.

3.2.1 Curves

Let $\mathcal{P}^J$ be an ordered set of points $\mathcal{P}^J = (\mathcal{P}^J_i)_{i=0,\ldots,m_J-1}$ in $\mathbb{R}^2$, the minimum number of points allowed for our model is 4. If we have $m_J = 2^J + 3$ points at resolution J, then the rule for calculating the growth is: $m_{J+1} = 2^{J+1} + 3$ for resolution $J+1$.
The definition of the sum $\oplus$ is the same as in definition (1), one must simply consider that a polygon is an ordered set of points.
The function φ maps to an ordered set of points. For the sake of simplicity, we will set it as $\varphi = \varphi(\mathcal{P}^J)$. If $\mathcal{P}^J$ has m_J points, then φ has m_{J+1} components.

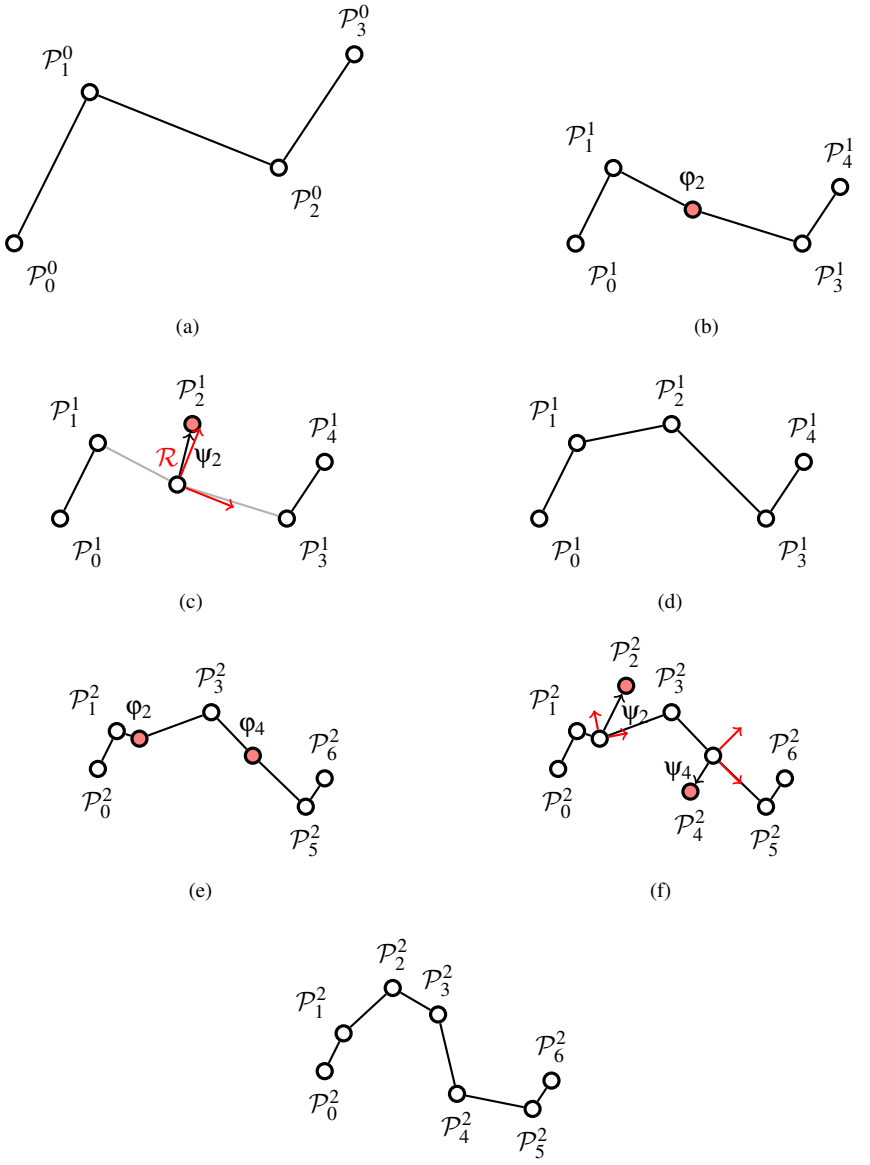


Figure 8: Multiresolution synthesis steps:
(a) Initial points at level $J = 0$.
(b) Computing φ_2 by applying the mask.
(c) Detail ψ_2 added to φ_2.
(d) New points at level $J = 1$.
(e) Computing φ_2 and φ_4 by applying the mask.
(f) Details ψ_2, ψ_4 added to φ_2, φ_4 respectively.
(g) New points at level $J = 2$.

$\varphi(\mathcal{P}^J)$ is defined by:

$$\varphi(\mathcal{P}^J) = (\varphi_k)_{k=0,\ldots,m_{J+1}-1}$$

and we have $\mathcal{P}^J = (\mathcal{P}_i^J)_{i=0,\ldots,m_J-1}$. Then each component φ_k is defined by:

$$\varphi_k = \begin{cases} \frac{1}{2}(\mathcal{P}_i^J + \mathcal{P}_{i+1}^J) & \text{if } k = i = 0 \vee (i = m_J - 2 \wedge k = 2i) \\ \mathcal{P}_i^J & \text{if } k = 2i-1 \wedge 0 < i < m_J - 1 \\ \mathcal{M}(\mathcal{P}_i^J, \ldots, \mathcal{P}_{i+3}^J) & \text{if } k = 2i+2 \wedge i < m_J - 3 \end{cases}$$

where $\mathcal{M}$ is a mask containing a single column and four rows in this case. This mask plays an important role because it determines the nature of the deformation applied over a curve. For instance, this mask can imply fractal or smooth deformations.

Before the function ψ can be defined, we have to determine $\delta\mathcal{P}^J$. An important point here is the size of $\delta\mathcal{P}^J$: we know that $\mathcal{P}^J$ has $m_J = 2^J + 3$ points; then $\delta\mathcal{P}^J$ has $m_J - 3 = 2^J$ vectors.

Now, if we denote ψ in the same way as φ, we have $\psi = \psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ and $\psi = (\psi_k)_{k=0,\ldots,m_{J+1}-1}$ and each component ψ_k is defined by:

$$\psi_k = \begin{cases} 0 & \text{if } k = 0 \vee k = 2(m_J - 1) - 2 \\ 0 & \text{if } k = 2i-1 \wedge 0 < i < m_J - 1 \\ \mathcal{R}(\mathcal{P}_{i+1}^J, \mathcal{P}_{i+2}^J)\delta\mathcal{P}_i^J & \text{if } k = 2i+2 \wedge i < m_J - 3 \end{cases}$$

where $\mathcal{R}(\mathcal{P}_{i+1}^J, \mathcal{P}_{i+2}^J)$ is a frame similar to the one used by Forsy et al. [6], constructed using the two points P_{i+1}^J and P_{i+2}^J as follows:

Let P and Q be two points, the frame $\mathcal{R}(P,Q)$ is defined by:

$$\mathcal{R}(P,Q) = (\vec{u}, \vec{v}) \quad \text{where } \vec{u} = \frac{\overrightarrow{PQ}}{\|PQ\|} \text{ and } \vec{u} \perp \vec{v}$$

This representation has the important advantage of being able to retain the object's

shape when applying rotations or translations to control points.

The steps needed to compute a new point are detailed in (Fig. 8). The analysis step is rather simple to perform. It consists in keeping points having an odd index and using those with an even index to compute the detail using a local frame inversion:

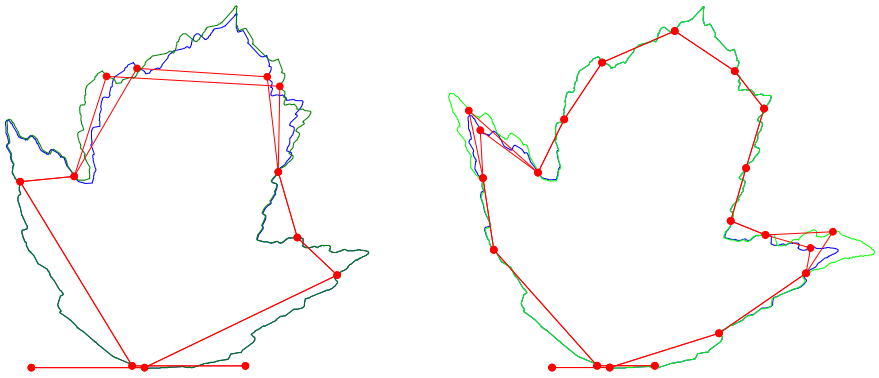
$$\delta\mathcal{P}_i^J = (\mathcal{R}(\mathcal{P}_{i+1}^J, \mathcal{P}_{i+2}^J))^{-1}(\mathcal{P}_k^{J+1} - \mathcal{M}(\mathcal{P}_i^J, \ldots, \mathcal{P}_{i+3}^J)) \quad \text{with } k = 2i+2 \wedge i < m_J - 3$$

To compute extreme points, the formula is inverted as follows:

$$\begin{cases} \mathcal{P}_0^J = 2 * \mathcal{P}_0^{J+1} - \mathcal{P}_1^J \\ \mathcal{P}_{m_j-1}^J = 2 * \mathcal{P}_{2*(m_J-1)-2}^{J+1} - \mathcal{P}_{m_J-2}^J \end{cases}$$

Results

Fig. 9 shows how our model is used to locally perform deformation on a curve. A preliminary analysis step is applied on the curve to obtain detail vectors. Then the curve can be deformed at any resolution level.



(a) (b)

Figure 9: Multiresolution editing. (a) With 11 control points. (b) With 19 control points (the level immediately after 11 control points).

Fig. 10 shows how our model can keep local specificities of the curve using the local frame representation of the detail.

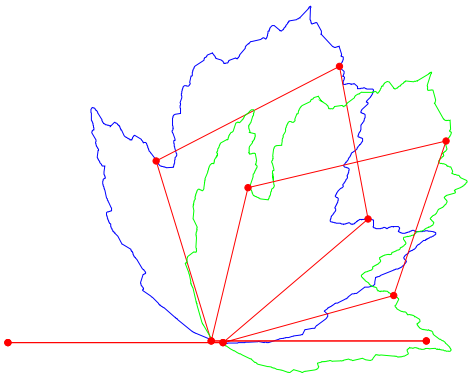


Figure 10: Conservation of local specificities.

3.2.2 Surfaces

In this section, two cases will be presented. The first treats height field surfaces and the second treats three-dimensional surfaces. The processing of these two cases is actually the same. The difference is only in detail processing: whereas the details in the first case are represented by scalar values, they are represented by three-component vectors in the second case.

Height Field Surface

Here $\mathcal{P}^J$ is an ordered set (grid) of scalars $\mathcal{P}^J = (\mathcal{P}^J_{ij})_{i=0,\ldots,m-1,j=0,\ldots,n-1}$ where $m = 2^U + 3$ and $n = 2^V + 3$ and $(J = min(U,V))$. The minimum grid size accepted by our model is a grid with four rows and four columns of scalars. The rule that increases the number of data in each row and each column of the grid, develops as follows: If a grid has $(2^U + 3) \times (2^V + 3)$ scalars in the Jth level, then a grid with $(2^{U+1} + 3) \times (2^{V+1} + 3)$ scalars in the $J + 1$th level will result.

The sum $\oplus$ is defined as follows:

Let $P = (P_{ij})_{i=0,\ldots,m-1,j=0,\ldots,n-1}$ and $Q = (Q_{ij})_{i=0,\ldots,m-1,j=0,\ldots,n-1}$ two ordered grids of scalars of the same size $m \times n$, we define $P \oplus Q$ to be a new ordered grid having $m \times n$

scalars. It is computed as follows:

$$P \oplus Q = (P_{ij} + Q_{ij})_{i=0,\ldots,m-1,j=0,\ldots,n-1}$$

Because the formulas that define the functions φ and ψ are long and complex to write, they will be explained using Figs. 11(a)-11(b)-11(c). (For formulas see Appendix A).

The result of φ and ψ functions is a scalar grid. If $\mathcal{P}^J$ has $(2^U+3)\times(2^V+3)$ scalars then each function of φ and ψ has $(2^{U+1}+3)\times(2^{V+1}+3)$ scalars where ($J = min(U,V)$). Indeed, we use the fact that each one of φ and ψ is a grid only in the theoretical definitions; in the implementation a more efficient data structure is used to improve storage and computation time.

We use several masks to calculate the $J+1$th level from the Jth level. The advantage is to have greater control on the form generated with these masks. Another advantage with our model is that the analysis (from the $J+1$th level to the Jth level) is carried out swiftly by removing data from the grid and then calculating the details associated with these data.

Figure (11(a)) shows how to calculate new scalars in even columns and even rows of the grid at the $J+1$th level except for the first and last column/row; the black balls represent the places of scalars in the grid at the Jth level used to calculate the new scalar. In this case, we employ a (4×4) mask. The new scalar will be the result of convolution between the mask and the scalars represented by the black balls added to the associated details. The placement of the new scalar compared to those above is shown in Fig. 11(a) by red ball.

Figure (11(b)) explains how we calculate the scalars in even columns and odd rows. We employ a (3×4) mask in this case. As in the previous case, the new scalar will be the result of convolution between the mask and the scalars represented by the black balls added to the associated details. The placement of the new scalar compared to former scalars is shown in Fig. 11(b) by a blue ball. A similar method is used to

calculate the scalars that existed in the odd columns and even rows. The difference is that the mask size becomes (4×3) and that the scalars used in convolution are the same size as the mask (four rows and three columns).

Figure (11(c)) uses the same method to calculate the scalars in the first or the last row and even columns. The new scalar is represented by a green ball. The mask size used in this case is (2×4). It will be a (4×2) mask to calculate the scalars in the first or the last column and even rows.

All the other scalars have no associated detail, and they are calculated quickly by using fewer data than in the previous cases. For example, in Fig. 12, which represents a completed transition from a (4×7) grid to a (5×11) grid, all the black balls of the right grid have the same scalar values as the corresponding balls in the left grid.

Results

Figure (13) shows how this model can be used for multiresolution modelling of a terrain. We initialise the terrain as a height field surface having a null altitude throughout. Then we analyze it using our masks to find the associated multiresolution model (control points and detail). In (13(b)) a gorge is created at a resolution level and a mountain is added at another resolution level (13(c)). Fractal masks has been used such that deformations are fractal too.

Figure (14) shows the rendering of a height field terrain modelled by our method.

Three-dimensional Surface:

This section discusses the three-dimensional surface, which is defined by a grid of control points (as the B-Spline surfaces).

The handling steps of this surface type are the same as in the example of a height field surface, with the only difference being that in the previous case details are displacements on the $\mathbb{Z}$ axis, represented by scalar values, whereas in this case the details are three-dimensional vectors represented according to a local frame $\mathcal{R}$ (as in the

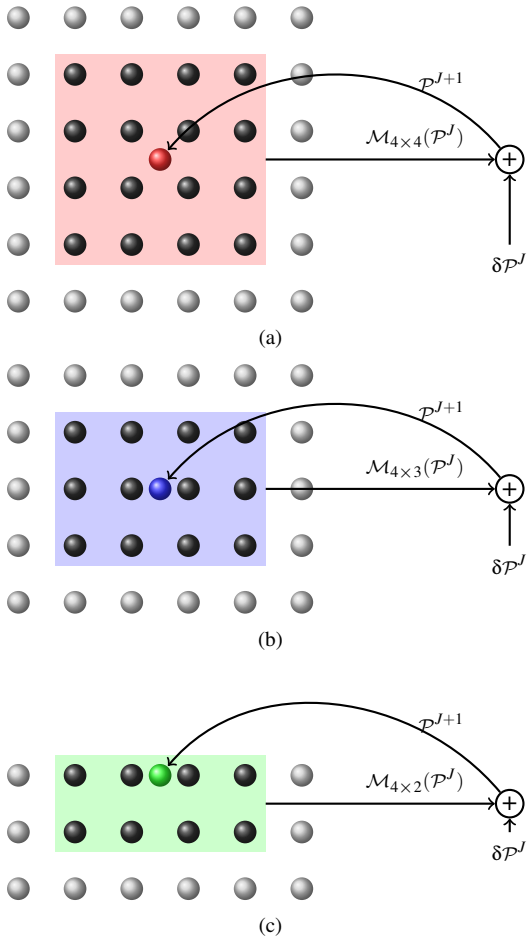


Figure 11: **Principle of multiresolution synthesis for surfaces (transition from the J level to the $J+1$ level).**

(a) A (4×4) mask is used to compute new scalars (even columns and rows).

(b) A (3×4) mask is used to compute new scalars (even columns, odd rows).

(c) A (2×4) mask is used to compute new scalars (even columns, first row).

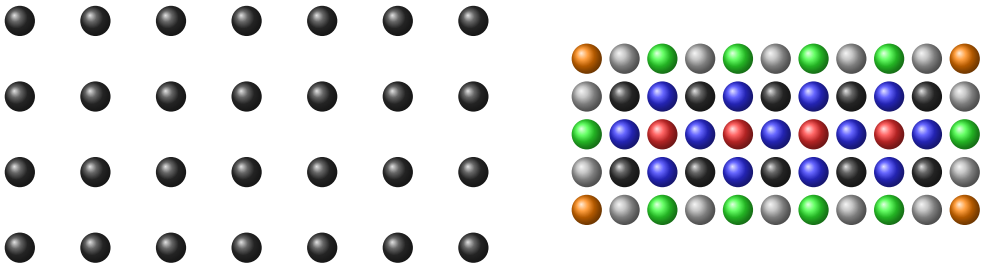


Figure 12: Complete transition from level J to $J+1$.

curve). We define $\mathcal{R}$ by using the control grid points (for greater detail see Appendix B).

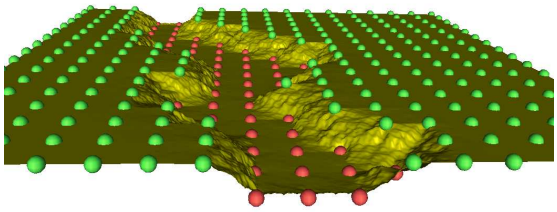
Results

Figure (15) shows the rendering of a three-dimensional terrain modelled by our method.

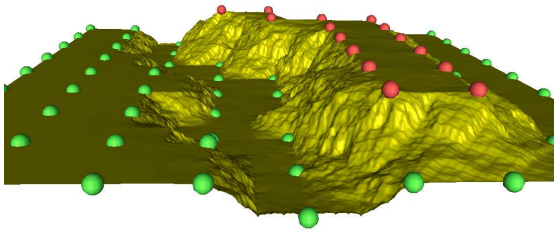
4 Conclusion

This article presented two self-similar models to perform multiresolution editing on curves or surfaces. The first model was based on projected IFS: it is a fractal model integrating the detail concept. Although this model is effective for curves, it increases in complexity for surfaces. For this reason we introduced a second model, which combines the principle of subdivision surface and the principle of detail. It can handle smooth shapes as well as rough ones depending on the masks used. To increase its capacity to control shapes, this model employs several masks to perform each synthesis or analysis step.

(a) Initialisation



(b) Creation of a gorge



(c) Creation of a mountain

Figure 13: Terrain modelling steps.

Figure 14: Rendering of a height field terrain.

Figure 15: Rendering of a 3D terrain.

References

[1] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, Septemper 1978.

[2] Chems Eddine Zaïr and Eric Tosan. Fractal modeling using free form techniques. *Comput. Graph. Forum*, 15(3):269–278, 1996.

[3] Adam Finkelstein and David Salesin. Multiresolution curves. In *SIGGRAPH*, pages 261–268, 1994.

[4] Gershon Elber. Multiresolution curve editing with linear constraints. *J. Comput. Inf. Sci. Eng.*, 1(4):347–355, 2001.

[5] Henning Biermann, Ioana M. Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Trans. Graph.*, 21(3):312–321, 2002.

[6] David R. Forsey and Richard H. Bartels. Hierarchical b-spline refinement. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 205–212, New York, NY, USA, 1988. ACM.

[7] Marryat Ma and Stephen Mann. Multiresolution editing of pasted surfaces. In *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 273–282, Nashville, TN, USA, 2001. Vanderbilt University.

[8] Frutuoso G. M. Silva and Abel J. P. Gomes. Interactive editing of multiresolution meshes. In *SIBGRAPI*, pages 202–209, 2004.

[9] Stefanie Hahmann, Georges-Pierre Bonneau, Baptiste Caramiaux, and M. Cornillac. Multiresolution morphing for planar curves. *Computing*, 79(2-4):197–209, 2007.

[10] B.B. Mandelbrot J.W. Van Ness. Fractal brownian motions, fractal noises, and applications. *SIAM Review*, (10):422–437, 1968.

[11] Przemyslaw Prusinkiewicz and James Hanan. Lindenmayer systems, fractals, and plants. *Lecture Notes in Biomathematics*, 75, 1989.

[12] Aristid Lindenmayer. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, March 1968.

[13] Michael Barnsley. *Fractals everywhere*. Academic Press, 1988.

[14] Eric Guérin, Eric Tosan, and Atilla Baskurt. A fractal approximation of curves. *Fractals*, 9(1):95–103, mar 2001.

[15] Eric Guérin, Eric Tosan, and Atilla Baskurt. Modeling and approximation of fractal surfaces with projected IFS attractors. In M. M. Novak, editor, *Emergent Nature, Fractal 2002 proceedings*, pages 293–303. World Scientific, mar 2002.

[16] Jacques Blanc-Talon. Self-controlled fractal splines for terrain reconstruction. In *CTME/GIP*, Arcueil, 1997.

[17] Farès Belhadj. Terrain modeling: a constrained fractal model. In *Afrigraph*, pages 197–204, 2007.

[18] Szymon Stachniak and Wolfgang Stuerzlinger. An algorithm for automated fractal terrain deformation. In *Computer Graphics and Artificial Intelligence*, May 2005.

[19] Y. Meyer. Les ondelettes. In *Contributions to nonlinear partial differential equations, Vol. II (Paris, 1985)*, volume 155 of *Pitman Res. Notes Math. Ser.*, pages 158–171. Longman Sci. Tech., Harlow, 1987.

[20] Stéphane Mallat. A theory for multiresolution signal decomposition: The wavelet-trepresentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989.

[21] E. Tosan, I. Bailly-Salins, I. Stotz, G. Gouaty, and Y. Weinand. Modelisation iterative de courbes et surfaces : aspect multiresolution. In *Groupe de travail en Modelisation Geometrique journee de Valenciennes*, pages 55–69, mars 2007.

[22] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*, chapter Nonlinear Models. Cambridge University Press, 1993.

[23] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6), 1978.

[24] Andrei Khodakovsky and Peter Schröder. Fine level feature editing for subdivision surfaces. In *Symposium on Solid Modeling and Applications*, pages 203–211, 1999.

[25] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 35–44, New York, NY, USA, 1993. ACM.

[26] Martin Bertram, Mark A. Duchaineau, Bernd Hamann, and Kenneth I. Joy. Generalized b-spline subdivision-surface wavelets for geometry compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):326–338, 2004.

Appendix A: Formulas for a height field surface

In this case, the function φ is defined as follows:

$$\varphi_{lk} = \begin{cases} \mathcal{M}_{22} * \begin{pmatrix} \mathcal{P}^J_{ij} & \mathcal{P}^J_{ij+1} \\ \mathcal{P}^J_{i+1j} & \mathcal{P}^J_{i+1j+1} \end{pmatrix} & \text{If } \begin{pmatrix} ((i=0 & \wedge & j=0) & \vee \\ (i=0 & \wedge & j=n-2) & \vee \\ (i=m-2 & \wedge & j=0) & \vee \\ (i=m-2 & \wedge & j=n-2)) & \wedge \\ (l=2i & \wedge & k=2j) & \end{pmatrix} \\ \mathcal{M}_{21} * \begin{pmatrix} \mathcal{P}^J_{ij} \\ \mathcal{P}^J_{i+1j} \end{pmatrix} & \text{If } \begin{pmatrix} ((i=0 & \wedge & 0<j<n-1) & \vee \\ (i=m-2 & \wedge & 0<j<n-1)) & \wedge \\ (l=2i & \wedge & k=2j-1) & \end{pmatrix} \\ \mathcal{M}_{12} * \begin{pmatrix} \mathcal{P}^J_{ij} & \mathcal{P}^J_{ij+1} \end{pmatrix} & \text{If } \begin{pmatrix} ((j=0 & \wedge & 0<i<m-1) & \vee \\ (j=n-2 & \wedge & 0<i<m-1)) & \wedge \\ (l=2i-1 & \wedge & k=2j) & \end{pmatrix} \\ \mathcal{P}^J_{ij} & \text{If } \begin{pmatrix} (0<i<m-1 & \wedge & 0<j<n-1) & \wedge \\ (l=2i-1 & \wedge & k=2j-1) & \end{pmatrix} \\ \mathcal{M}_{24} * \begin{pmatrix} \mathcal{P}^J_{ij} & \cdots & \mathcal{P}^J_{ij+3} \\ \mathcal{P}^J_{i+1j} & \cdots & \mathcal{P}^J_{i+1j+3} \end{pmatrix} & \text{If } \begin{pmatrix} ((i=0 & \wedge & j<n-3) & \vee \\ (i=m-2 & \wedge & j<n-3)) & \wedge \\ (l=2i & \wedge & k=2j+2) & \end{pmatrix} \\ \mathcal{M}_{42} * \begin{pmatrix} \mathcal{P}^J_{ij} & \mathcal{P}^J_{ij+1} \\ \vdots & \vdots \\ \mathcal{P}^J_{i+3j} & \mathcal{P}^J_{i+3j+1} \end{pmatrix} & \text{If } \begin{pmatrix} ((j=0 & \wedge & i<m-3) & \vee \\ (j=n-2 & \wedge & i<m-3)) & \wedge \\ (l=2i+2 & \wedge & k=2j) & \end{pmatrix} \\ \mathcal{M}_{34} * \begin{pmatrix} \mathcal{P}^J_{i-1j} & \cdots & \mathcal{P}^J_{i-1j+3} \\ \mathcal{P}^J_{ij} & \cdots & \mathcal{P}^J_{ij+3} \\ \mathcal{P}^J_{i+1j} & \cdots & \mathcal{P}^J_{i+1j+3} \end{pmatrix} & \text{If } \begin{pmatrix} ((0<i<m-1 & \wedge & j<n-3) \\ (l=2i-1 & \wedge & k=2j+2) \end{pmatrix} \\ \mathcal{M}_{43} * \begin{pmatrix} \mathcal{P}^J_{ij-1} & \mathcal{P}^J_{ij} & \mathcal{P}^J_{ij+1} \\ \vdots & \vdots & \vdots \\ \mathcal{P}^J_{i+3j-1} & \mathcal{P}^J_{i+3j} & \mathcal{P}^J_{i+3j+1} \end{pmatrix} & \text{If } \begin{pmatrix} ((0<j<n-1 & \wedge & i<m-3) & \wedge \\ (l=2i+2 & \wedge & k=2j-1) & \end{pmatrix} \\ \mathcal{M}_{44} * \begin{pmatrix} \mathcal{P}^J_{ij} & \cdots & \mathcal{P}^J_{ij+3} \\ \vdots & \ddots & \vdots \\ \mathcal{P}^J_{i+3j} & \cdots & \mathcal{P}^J_{i+3j+3} \end{pmatrix} & \text{If } \begin{pmatrix} ((i<m-3 & \wedge & j<n-3) & \wedge \\ (l=2i+2 & \wedge & k=2j+2)) & \end{pmatrix} \end{cases}$$

where $\mathcal{P}^J_{ij}$ is a scalar value, and $(\mathcal{M}_{22}, \mathcal{M}_{21}, \mathcal{M}_{12}, \mathcal{M}_{34}, \mathcal{M}_{43}, \mathcal{M}_{44})$ are the different masks used in the model. The function ψ is defined as follows:

$$
\psi_{lk} = \begin{cases}
0 & \text{If} \begin{pmatrix} ((i=0 & \wedge & j=0) & \vee \\ (i=0 & \wedge & j=n-2) & \vee \\ (i=m-2 & \wedge & j=0) & \vee \\ (i=m-2 & \wedge & j=n-2)) & \wedge \\ (l=2i & \wedge & k=2j) & \end{pmatrix} \\
0 & \text{If} \begin{pmatrix} ((i=0 & \wedge & 0<j<n-1) & \vee \\ (i=m-2 & \wedge & 0<j<n-1)) & \wedge \\ (l=2i & \wedge & k=2j-1) & \end{pmatrix} \\
0 & \text{If} \begin{pmatrix} ((j=0 & \wedge & 0<i<m-1) & \vee \\ (j=n-2 & \wedge & 0<i<m-1)) & \wedge \\ (l=2i-1 & \wedge & k=2j) & \end{pmatrix} \\
0 & \text{If} \begin{pmatrix} (0<i<m-1 & \wedge & 0<j<n-1) & \wedge \\ (l=2i-1 & \wedge & k=2j-1) & \end{pmatrix} \\
v\delta\mathcal{P}_{ij}^J & \text{If} \begin{pmatrix} ((i=0 & \wedge & j<n-3) & \vee \\ (i=m-2 & \wedge & j<n-3)) & \wedge \\ (l=2i & \wedge & k=2j+2) & \end{pmatrix} \\
h\delta\mathcal{P}_{ij}^J & \text{If} \begin{pmatrix} ((j=0 & \wedge & i<m-3) & \vee \\ (j=n-2 & \wedge & i<m-3)) & \wedge \\ (l=2i+2 & \wedge & k=2j) & \end{pmatrix} \\
v\delta\mathcal{P}_{ij}^J & \text{If} \begin{pmatrix} (0<i<m-1 & \wedge & j<n-3) & \wedge \\ (l=2i-1 & \wedge & k=2j+2) & \end{pmatrix} \\
h\delta\mathcal{P}_{ij}^J & \text{If} \begin{pmatrix} (0<j<n-1 & \wedge & i<m-3) & \wedge \\ (l=2i+2 & \wedge & k=2j-1) & \end{pmatrix} \\
d\delta\mathcal{P}_{ij}^J & \text{If} \begin{pmatrix} (i<m-3 & \wedge & j<n-3) & \wedge \\ (l=2i+2 & \wedge & k=2j+2) & \end{pmatrix}
\end{cases}
$$

where $(h\delta\mathcal{P}_{ij}^J, v\delta\mathcal{P}_{ij}^J, d\delta\mathcal{P}_{ij}^J)$ are scalar values and they represent vertical, horizontal and diagonal details, respectively.

Appendix B: Formulas for a three-dimensional surface

In this case, the function φ is defined as in the height field surface (Appendix A). However, here $\mathcal{P}_{ij}^J$ is a point with three coordinates. Like the curve, the function ψ is defined as follows:

$$\psi_{lk} = \begin{cases} 0 & \text{If } \begin{pmatrix} ((i=0 & \wedge & j=0) & \vee \\ (i=0 & \wedge & j=n-2) & \vee \\ (i=m-2 & \wedge & j=0) & \vee \\ (i=m-2 & \wedge & j=n-2)) & \wedge \\ (l=2i & \wedge & k=2j) & \end{pmatrix} \\ 0 & \text{If } \begin{pmatrix} ((i=0 & \wedge & 0<j<n-1) & \vee \\ (i=m-2 & \wedge & 0<j<n-1)) & \wedge \\ (l=2i & \wedge & k=2j-1) & \end{pmatrix} \\ 0 & \text{If } \begin{pmatrix} ((j=0 & \wedge & 0<i<m-1) & \vee \\ (j=n-2 & \wedge & 0<i<m-1)) & \wedge \\ (l=2i-1 & \wedge & k=2j) & \end{pmatrix} \\ 0 & \text{If } \begin{pmatrix} (0<i<m-1 & \wedge & 0<j<n-1) & \wedge \\ (l=2i-1 & \wedge & k=2j-1) & \end{pmatrix} \\ \mathcal{R}_4 \begin{pmatrix} \mathcal{P}^J_{ij+1} & \mathcal{P}^J_{ij+2} \\ \mathcal{P}^J_{i+1j+1} & \mathcal{P}^J_{i+1j+2} \end{pmatrix} v\delta\mathcal{P}^J_{ij} & \text{If } \begin{pmatrix} ((i=0 & \wedge & j<n-3) & \vee \\ (i=m-2 & \wedge & j<n-3)) & \wedge \\ (l=2i & \wedge & k=2j+2) & \end{pmatrix} \\ \mathcal{R}_4 \begin{pmatrix} \mathcal{P}^J_{i+1j} & \mathcal{P}^J_{i+1j+1} \\ \mathcal{P}^J_{i+2j} & \mathcal{P}^J_{i+2j+1} \end{pmatrix} h\delta\mathcal{P}^J_{ij} & \text{If } \begin{pmatrix} ((j=0 & \wedge & i<m-3) & \vee \\ (j=n-2 & \wedge & i<m-3)) & \wedge \\ (l=2i+2 & \wedge & k=2j) & \end{pmatrix} \\ \mathcal{R}_6 \begin{pmatrix} \mathcal{P}^J_{i-1j+1} & \mathcal{P}^J_{i-1j+2} \\ \mathcal{P}^J_{ij+1} & \mathcal{P}^J_{ij+2} \\ \mathcal{P}^J_{i+1j+1} & \mathcal{P}^J_{i+1j+2} \end{pmatrix} v\delta\mathcal{P}^J_{ij} & \text{If } \begin{pmatrix} (0<i<m-1 & \wedge & j<n-3) & \wedge \\ (l=2i-1 & \wedge & k=2j+2) & \end{pmatrix} \\ \mathcal{R}_6 \begin{pmatrix} \mathcal{P}^J_{i+1j-1} & \mathcal{P}^J_{i+1j} & \mathcal{P}^J_{i+1j+1} \\ \mathcal{P}^J_{i+2j-1} & \mathcal{P}^J_{i+2j} & \mathcal{P}^J_{i+2j+1} \end{pmatrix} h\delta\mathcal{P}^J_{ij} & \text{If } \begin{pmatrix} (0<j<n-1 & \wedge & i<m-3) & \wedge \\ (l=2i+2 & \wedge & k=2j-1) & \end{pmatrix} \\ \mathcal{R}_4 \begin{pmatrix} \mathcal{P}^J_{i+1j+1} & \mathcal{P}^J_{i+1j+2} \\ \mathcal{P}^J_{i+2j+1} & \mathcal{P}^J_{i+2j+2} \end{pmatrix} d\delta\mathcal{P}^J_{ij} & \text{If } \begin{pmatrix} (i<m-3 & \wedge & j<n-3) & \wedge \\ (l=2i+2 & \wedge & k=2j+2) & \end{pmatrix} \end{cases}$$

where $(h\delta\mathcal{P}^J_{ij}, v\delta\mathcal{P}^J_{ij}, d\delta\mathcal{P}^J_{ij})$ are vectors with three coordinates and they represent vertical, horizontal and diagonal details, respectively, and

$$\mathcal{R}_4 : (\mathbb{R}^3)^4 \longrightarrow (\mathbb{R}^3)^3$$

and

$$\mathcal{R}_6 : (\mathbb{R}^3)^6 \longrightarrow (\mathbb{R}^3)^3$$

are functions that automate the computation of local frames by using 4 or 6 points. The frame is composed of three perpendicular vectors: two of them are in an approximate plane that minimizes the distance to the 4 or 6 given points, and the third is the normal of this plane.