Computers & Graphics: X 2 (2019) 100006

Contents lists available at ScienceDirect

Computers & Graphics: X

journal homepage: www.elsevier.com/locate/cagx

Special Section on SMI 2019

Representation of NURBS surfaces by Controlled Iterated Functions System automata

Lucas Morlet*, Christian Gentil*, Sandrine Lanquetin, Marc Neveu, Jean-Luc Baril

Laboratoire Informatique de Bourgogne (LIB), Univ. Bourgogne Franche - Comté, 8 avenue Alain Savary, 21000 Dijon, France

ARTICLE INFO

Article history:
Received 9 March 2019
Revised 24 May 2019
Accepted 24 May 2019
Available online 7 June 2019

Keywords:
Geometric modeling
Iterative modeling
Non-Uniform Rational B-Splines
Iterated Functions Systems
CIFS automata

ABSTRACT

Iterated Function Systems (IFS) are a standard tool to generate fractal shapes. In a more general way, they can represent most of standard surfaces like Bézier or B-Spline surfaces known as self-similar surfaces. Controlled Iterated Function Systems (CIFS) are an extension of IFS based on automata. CIFS are basically multi-states IFS, they can handle all IFS shapes but can also manage multi self-similar shapes. For example CIFS can describe subdivision surfaces around extraordinary vertices whereas IFS cannot. Having a common CIFS formalism facilitates the development of generic methods to manage interactions (junctions, differences...) between objects of different natures.

This work focuses on a CIFS approach of Non-Uniform Rational B-Splines (NURBS) which are the main used representation of surfaces in CAGD systems. By analyzing the recursive generating process of basis functions, we prove the stationarity of NURBS computation. This implies that NURBS can be represented as a finite automaton: a CIFS. Subdivision transformations implied in the generating process are directly deduced from blossoming formulation and are expressed as a function of the initial nodal vector. We provide a method to construct the CIFS automata for NURBS of any-degree. Then NURBS-surfaces automata are deduced using a "tensor-product" of NURBS automata. This new representation of NURBS allows us to build a bridge between them and other surfaces already represented in CIFS formalism: fractals and subdivision surfaces.

1. Introduction

Iterative processes based on fractal geometry is a common way to produce complex objects like terrains, clouds, trees, textures [1,2]. Generally, these processes used stochastic parameters to model natural objects with non-strict repetition of the same pattern. These algorithms are widely used in Computer Graphics but few in Computer Aided Geometric Design (CAGD), even if resulting shapes present interesting properties like, for example, rough surfaces to improve heat exchange, porous volumes needing less material or simply for their aestetic value. The purpose of our research is to develop a generic iterative geometric modeler for industrial applications associated to additive manufacturing. A first application, related to energy saving which is a crucial issue for industry, consists of designing lighter objects while maintaining high mechanical properties. To address this challenge we suggest to consider multiscale controlled lacunary structures which fill a given geometric hull, with imposed functional surface. A second application is tree-like structures support design for additive manufacturing or for reinforcing a given functional surfaces (see Fig. 1). In both cases, imposed geometric inputs, geometric hulls or functional surfaces, come from a CAGD software and are generally represented by Non-Uniform Rational B-Splines (NURBS). From the past decades, NURBS imposed themselves into CAGD. They are a powerful tool to construct smooth surfaces while respecting usual constraints of industrial conception. Our iterative modeler have to interact with CAGD software and needs to integrate NURBS representation.

From a theoretical point of view, it could be interesting to have a common representation for fractals, subdivision surfaces and NURBS to identify possible interactions. From a practical point of view, we could take advantage of each representation, without conversions and approximations.

Few generic representations were proposed for iterated processes. Most of the time, an iterative process is translated in a specific algorithm with a specific data structure and implemented in a specific software for a specific application. The two main generic representations are L-system [3] and Iterated Function

* Corresponding authors.
E-mail addresses: lucas.morlet@u-bourgogne.fr (L. Morlet), christian.gentil@u-bourgogne.fr (C. Gentil).

https://doi.org/10.1016/j.cagx.2019.100006

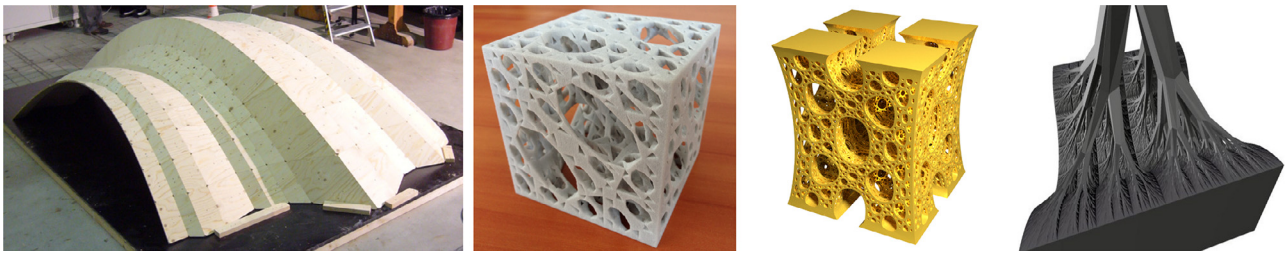

Fig. 1. Original creations of CIFS-based CAGD-modeling. From left to right: a wooden overlapping structure which is smooth in one direction and fractal in the second direction (©IBOIS, laboratory of EPLF), a lacunar structure build by 3D printing, a lacunar structure whose edges are B-Splines using topological optimization (≈88% of material saved compared to a filled structure) and an arborescent fractal structure that support a bi-quadratic Bézier surface.

System (IFS) [4] which are quasi-equivalent [5]. L-system representation is based on rewriting rules while IFS representation is based on a self-similarity property. For our purpose we believe that IFS are more suitable according to their formulation which is easier for formal manipulation associated to CAGD. Iterated Function Systems (IFS) have been introduced by Hutchinson [4] and developed by Barnsley [6] to study self-similar shapes. Even if they are initially created for fractals, they can represent other types of objects: Goldman [7] and Schaeffer et al. [8] used this formalism to respectively generate Bèzier and B-Spline tensor-surfaces. In previous works, Zaïr and Tosan [9] introduce fractal modeling using free-form techniques which includes Bèzier and uniform B-spline curves and surfaces. Due to the presence of irregular vertices, subdivision surfaces are not self-similar and cannot be handled by IFS. To overpass this constraint, Morlet et al. [10] used Controlled Iterated Function Systems (CIFS) to generate low-degree uniform subdivision surfaces in an iterative way. CIFS are an extension of IFS based on automata whose states can represent different objects, self-similarities are no longer required. Every IFS can be described as a one-state CIFS-automaton.

In addition, to bring together usual CAGD representations in a common formalism, CIFS-representation provides specific tools. For instance, Podkorytov et al. [11] proved that interactions between two objects of different natures can be performed since they are both represented as CIFS-automata.

At this point, our CIFS-based CAGD-modeler does not handle NURBS, and this is the main purpose of this paper. We prove that NURBS tensor-surfaces of any degree can be represented as CIFS-automata and so can be integrated in our iterative modeler. Then we avail this integration for new applications.

2. Overview

First of all, we introduce in Section 3 all the notations and mathematical background that will be used in the paper. We explain two usually separate research fields: IFS and their extension CIFS in a first part and NURBS and their blossoming representation in a second one. Our work is placed at the cross of these two research fields.

In Section 4, we present how to create NURBS CIFS-automata. The automata for quadratic and cubic cases are fully described and a method to generate any-degree automaton is proposed.

Furthermore we explain how to generate "tensor-products" of complete automata in Section 5. Following this method we create bi-quadratic and bi-cubic NURBS tensor-surfaces CIFS-automata. Then we prove that any NURBS tensor-surfaces can be represented as a CIFS automaton.

In Section 6, we present several solutions that our formalism brings to CAGD-industry usual challenges. First, we use the formalism of Section 5 to generate blending tensor-surfaces (*i.e.* the two generative curves are of different natures). Then we show how to create a junction that respects constraints between a NURBS tensor-surface and a surface already defined as a CIFS-automaton. To finish we present the application of this formalism to the topological optimization field: we create a lacunar NURBS tensor-surface and an arborescent structure that support a NURBS tensor-surfaces.

To conclude, all contributions are summarized and research tracks for extension to any-degree non-tensor NURBS surface are proposed in Section 7.

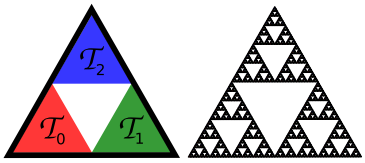

Fig. 2. IFS of Sierpinski triangle and its attractor. Transformations $\mathcal{T}_0, \mathcal{T}_1$, and $\mathcal{T}_2$ associate the black triangle to one of the three sub-triangles (respectively red, green, and blue). These transformations are homotheties of scale factor 0.5 where the center is one of the vertices of the initial triangle.

3. Background

3.1. Iterated Functions Systems (IFS)

According to Huchinson [4], an IFS is a set of n contracting transformations of a complete metric space $(\mathcal{X}, d)$: $\{\mathcal{T}_i\}_{i\in\Sigma}$ with $\Sigma = \{0, 1, \ldots, n-1\}$. Let $\mathcal{K}$ be a compact (*i.e.* a subset of an Euclidean space which is both closed and bounded), the Hutchinson operator $\mathcal{H}$ is defined by:

$$\mathcal{H}(\mathcal{K}) = \bigcup_{i=0}^{n-1} \mathcal{T}_i(\mathcal{K})$$

In a same way as a contractive transformation owns a unique fixed-point, there is a unique compact, $\mathcal{A}$, referred as the attractor of the IFS, which is defined as:

$$\mathcal{H}(\mathcal{A}) = \mathcal{A} \tag{1}$$

Due to the contractivity of the Huchinson operator, the attractor can be computed as the result of an infinite application of Hutchinson operator on any starting compact $\mathcal{K}$.

$$\mathcal{A} = \underbrace{\mathcal{H} \circ \ldots \circ \mathcal{H}}_{\infty}(\mathcal{K})$$

An usual example of IFS is the Sierpinski triangle presented in Fig. 2.

Every point belonging to the attractor can be indexed by at least one infinite word $\sigma = \sigma_0\sigma_1\cdots\sigma_\ell\cdots$ of Σ ($\sigma \in \Sigma^\infty$) such that:

$$\lim_{\ell\longrightarrow\infty} \mathcal{T}_{\sigma_0} \circ \mathcal{T}_{\sigma_1} \circ \cdots \circ \mathcal{T}_{\sigma_\ell}(p) = p_\sigma, \ \forall p \in \mathcal{X}.$$

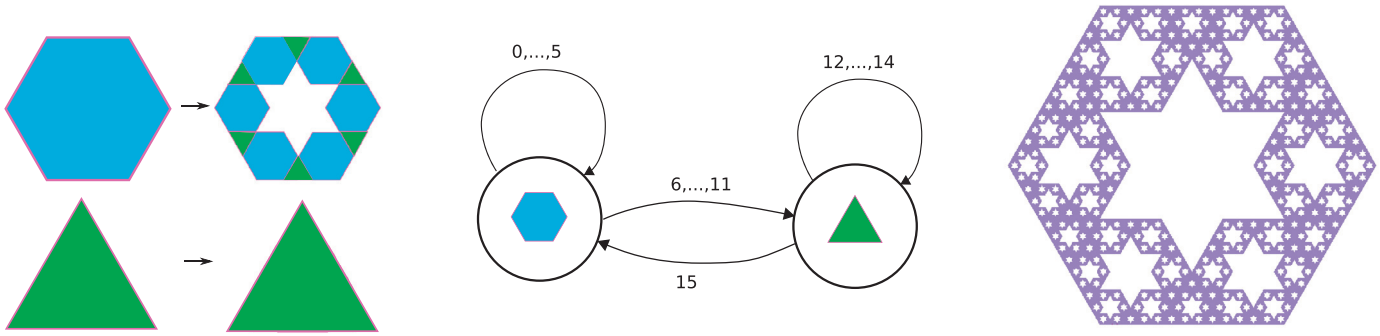

Fig. 3. Example of a CIFS automaton composed of two states: "blue hexagon" and "green triangle". On the left, the decomposition rules of each state. On the center, the CIFS automaton deduced from decomposition rules. On the right, the resulting attractor.

In other words, the infinite composition of the corresponding transformations leads to the point p_σ. σ is referred as an address of the point p_σ.

To any infinite word $\sigma \in \Sigma^\infty$ corresponds a point of $\mathcal{A}$. In particular, the address i^ω (composed of an infinite number of repetition of i), corresponding to the fixed point of $\mathcal{T}_i$ belongs to $\mathcal{A}$, $p_{i^\omega} \in \mathcal{A}$. This property is important, because if $\mathcal{T}_i$ are affine or linear transformations (which will be the case in the following) we can directly compute a point belonging to $\mathcal{A}$ in a finite time by calculating the fixed point of $\mathcal{T}_i$ by eigen-analysis. Furthermore, according to Eq. (1), $\mathcal{T}_j p_{i^\omega}$ belongs to $\mathcal{A}$ too. This means that we can compute a tesselation of $\mathcal{A}$ with all vertices belonging to $\mathcal{A}$ in finite time. The tesselation is calculated from a finite number of iterations of the Hutchinson operator. The starting compact $\mathcal{K}$ is set as a list of vertices of a 1D, 2D or 3D mesh, each vertex being a fixed points of a tranformation $\mathcal{T}_i, i \in \Sigma$. This approach is equivalent to Halstead et al. [12] and Stam [13] methods which directly compute a piece of subdivision surface from a given patch of control mesh.

3.2. Controlled Iterated Functions Systems (CIFS)

CIFS are an extension of IFS that provides more control. Instead of defining the attractor as all points P_σ, $\sigma \in \Sigma^\omega$, the attractor is restricted to points whose addresses are accepted by a given automaton.

A CIFS is defined by:

- an automaton $(\Sigma, Q, \delta, \natural)$, where Σ is an alphabet, Q a set of states, δ a transition function (δ: $Q \times \Sigma \to Q$), $\natural$ the initial state and all states of Q are final states;
- a set of spaces associated to the states : $(\mathcal{X}^q)_{q\in Q}$;
- a set of transformations associated to the transitions T_i^x : $E^{\delta(x,i)} \to E^x$.

To each state is associated an attractor $\mathcal{A}^q$ defined by:

$$\mathcal{A}^q = \bigcup_{i\in\Sigma^q} T_i^q(\mathcal{A}^{\delta(q,i)}) \tag{2}$$

where $\Sigma^q = \{i \in \Sigma, \delta(q,i) \in Q\}$.

The attractor of the CIFS is the attractor of the initial state, i.e. $\mathcal{A} = \mathcal{A}^\natural$.

From Eq. (2), we understand that a transformation associated to a transition copy the attractor of the destination state into the attractor of the starting state. Then interpretation and design of an automaton is intuitive. An example can be found in Fig. 3.

The last step to complete the needed material, is the introduction of free-form shapes modeling with CIFS. It is straightforward by building the attractors in barycentric space and then project it into the modeling space according to a set of control points. The final modeled shape is $F = \mathbf{P}\mathcal{A}$, where $\mathbf{P} = [p_0 p_1 \dots p_k]$ is a line vector composed of the list of control points. The projection in the modeling space is included in the automaton by adding an initial state and a transition with $\mathbf{P}$ as the associated transformation. For example a quadratic uniform spline is defined by the automaton of Fig. 4, where $\mathbf{P} = [p_0 p_1 p_2]$ (the three control points), T_0^q and T_1^q represent the two Chaikin matrices and K is composed of the fixed points of T_0^q and T_1^q:

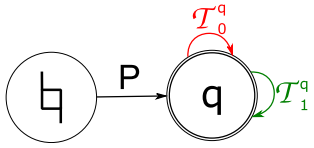

Fig. 4. CIFS automaton corresponding to Chainkin algorithm.

$$K = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}.$$

The attractor associated to state q is the quadratic uniform spline basic functions and the Spline curve is $\mathcal{A}^\natural = \mathbf{P}\mathcal{A}^q$. A tesselation of the curve is obtained by computing, for a given level of iteration (approximation) ℓ: $\{\mathbf{P}\mathcal{T}_{\sigma_0} \circ \dots \circ \mathcal{T}_{\sigma_\ell} K, \sigma_i \in \Sigma\}$.

By using different states, CIFS provides more control than IFS. For example, B-Spline tensor surfaces can easily be represented by an IFS [8] whereas subdivision surfaces, around extraordinary vertices, cannot. Indeed, management of extraordinary patches is impossible with IFS formalism because an irregular patch is subdivided into several regular and irregular patches. At least two states are necessary: regular and irregular. More generally, transformations that change the nature of the object can only exist in CIFS formalism because different objects are simply represented by different states. Morlet et al. [10] prove that any low-degree uniform subdivision schemes can be represented by a CIFS automaton instead of a set of rules. They propose a generic efficient visualization of subdivision surfaces on GPU, based on tesselation of attractors.

3.3. Non-Uniform Rational B-Splines

B-Splines are a user-friendly tool to define smooth parametric curves by a list of points called control points that form the control polygon. A B-Spline curve $\mathcal{C}$ of degree d is defined from a control polygon composed of $d+1$ points $\mathbf{P} = [p_0 \dots p_d]$.

A B-Spline curve can also be defined by pieces: a curve composed of m pieces is defined by $\mathbf{P} = [p_0 \dots p_{d+m-1}]$ where every sub-polygon $[p_i \dots p_{i+d-1}]$ defines a B-Spline of degree d. Even if B-Splines are built to be defined by pieces, each piece is independently computed; so only one-piece curves are treated in this article.

B-Splines basis functions are computed given a knot vector. For a one-piece B-Spline curve of degree d, the knot vector is $\mathbf{T} = [t_0 = 0 \leq \cdots \leq t_{(2d-1)} = 1]$. Every point of the curve $\mathcal{C}(t)$ is computed by the well-known Cox–De-Boor formula [14].

What is important for curves computation is not the knot values but differences between two consecutive knots and the ratio between these differences. Thus, the nodal vector $\mathbf{T}$ is often replaced by the inter-nodal vector $\mathbf{U} = [u_0 \dots u_{(2d-2)}]$ where $u_i = t_{i+1} - t_i$. In this article, the most convenient is used according to each situation.

Uniform B-Splines are often used (in this case it is convenient to set $\mathbf{T} = [t_0 = 0, t_1 = 1, < \cdots < t_{(2d-1)} = 2d-1]$ and then $u_0 = \cdots = u_{(2d-2)} = 1$).

Non-Uniform B-Splines (NUBS) offer more control over the curve. In this case u_i is computed according to user defined constraints (centripetal, chord length, ...)

The main problem NUBS have to face is their impossibility to handle conic shapes (*i.e.* circles, ellipses...). To overpass this issue a new concept is added: the rationality. NUBS then become Non-Uniform Rational B-Splines (NURBS) and a new coordinate is added to every control points. These coordinates increase or decrease points influence in a way similar to weight points. This rationality modifies the shape of the curves by adding an extra step in the computation which is independent from the non-uniformity. The rationality can be achieved simply by using control points with homogeneous coordinates. But it is not out purpose here, our work focuses only on the non-uniformity step and the term NURBS is used in a generic way.

3.4. Blossoming

NURBS are usually computed with the Cox-De-Boor formula [14] but Ramshaw [15] introduced an other way to understand NURBS called blossoming. A blossom of a curve of degree d is represented by a label composed of d arguments: $\{t_i \dots t_{i+d-1}\}$. Blossoming is completely characterized by three properties: symmetry, diagonal, and multi-affinity. From the three previous properties, another one is deduced: the consecutivity.

Symmetry:

The order of arguments in a label have no consequence on blossoms. Thus, they can and will be always written in ascending order:

$$\{\dots t_i \dots t_j \dots\} = \{\dots t_j \dots t_i \dots\} \qquad (j < i)$$

Diagonal:

Every point of the limit curve is defined by a label composed of identical arguments. This repeated argument is also the curve parameter.

$$\mathcal{C}(t) = \{t \dots t\}$$

Multi-affinity:

Every label can be defined as an affine combination of two others if these three labels only differ by one argument. Weights of the labels combination correspond to weights of the arguments affine combination:

$$\{\dots t \dots\} = \frac{b-t}{b-a}\{\dots a \dots\} + \frac{t-a}{b-a}\{\dots b \dots\}$$

Consecutivity:

A label whose arguments are consecutive nodes of the knot-vector corresponds to a control point; its index is the first index of arguments:

$$p_i : \{t_i, t_{i+1} \dots t_{i+d-1}\}$$

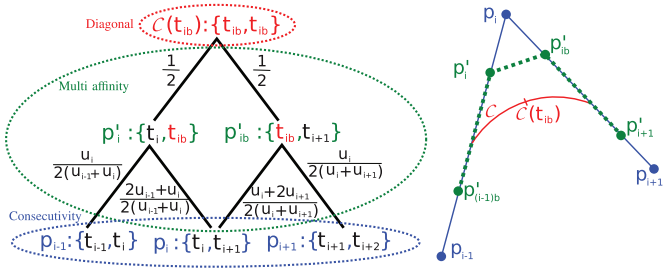

Fig. 5. On the left, the blossoming computation of the curve point $\mathcal{C}(t_{ib})$ where $t_{ib} = (t_i + t_{i+1})/2$. From this blossoming representation the influence of control points p_{i-1}, p_i, p_{i+1} are deduced: $\frac{1}{2} \cdot \frac{u_i}{2(u_{i-1}+u_i)}$, $\frac{1}{2} \cdot \frac{2u_{i-1}+u_i}{2(u_{i-1}+u_i)} + \frac{1}{2} \cdot \frac{u_i+2u_{i+1}}{2(u_i+u_{i+1})}$, and $\frac{1}{2} \cdot \frac{u_i}{2(u_i+u_{i+1})}$. On the right, the application of this result to compute this point of a quadratic NURBS (red) from a control polygon (blue). An intermediate polygon (green) that defines the same NURBS as the original one can also be computed according to the coefficients of the bottom branches of the blossoming. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Following these four properties, we can compute every point of the curve as a function of the control polygon. First, the diagonal property is used to create the label corresponding to the point of the curve and this label is placed as the root of the blossoming. Each label that does not respect the consecutive property have to be split into two new labels with respect to multi-affinity. The weights of affine combination are written on the branches that link the parent label to its two sons. A consecutive label is a leaf of the tree: it corresponds to a point of the control polygon. The weight of a path that links the root to a leaf is the product of branches weight it passes through. The influence of a control polygon point (leaf) on the curve point we compute (root) is the sum of paths weights joining the root to the leaf. An example of blossoming where a point of quadratic NURBS is computed is given in Fig. 5.

4. Representation of NURBS by CIFS-automata

In this section, we prove that NURBS with any-degree can be represented as a CIFS-automaton. The number of states is directly related to the degree of the NURBS. First the simplest cases (quadratic and cubic) are presented, then the generalization to any degree is explained and mathematically proven by generating words provided grammar rules.

4.1. NURBS and self-similarity

Let a NURBS be defined by a pair composed of a control polygon $\mathbf{P}$ and a knot-vector $\mathbf{T}$. If a new knot is inserted into $\mathbf{T}$, the new knot-vector $\mathbf{T}'$ can be associated to a new control polygon $\mathbf{P}'$ which defines the same NURBS as $(\mathbf{P}; \mathbf{T})$. An example can be found in Fig. 5 where the same curve is defined by the two pairs:

$$\begin{cases} [p_{i-1}, p_i, p_{i+1}] \\ [t_{i-1}, t_i, t_{i+1}, t_{i+2}] \end{cases} \text{ and } \begin{cases} [p_{i-1}, p'_i, p'_{ib}, p_{i+1}] \\ [t_{i-1}, t_i, t_{ib}, t_{i+1}, t_{i+2}] \end{cases}$$

The usual way to exploit this property is the mid-knot insertion strategy [16,17]: a new knot is inserted in the middle of each consecutive knots of $\mathbf{T}$. Then all knots of $\mathbf{T}'$ are multiplied by two because there are not the knots but the ratios between them that influences the shape of the curve. To finish, extreme knots are dropped because they do not influence the curve anymore. The mid-knot insertion strategy is equivalent to inter-node doubling: each inter-node of $\mathbf{U}$ is written twice in a row. Due to modification on knot/inter-node vector, the control polygon $\mathbf{P}$ has to be modified to $\mathbf{P}'$. For instance:

$$\mathbf{T} = [a, b, c, d] \rightarrow \mathbf{T}' = \left[a, \frac{a+b}{2}, b, \frac{b+c}{2}, c, \frac{c+d}{2}, d\right]$$

$\rightarrow \mathbf{T}' = [2a, a+b, 2b, b+c, 2c, c+d, 2d]$

$\rightarrow \mathbf{T}' = [a+b, 2b, b+c, 2c, c+d]$

$\mathbf{U} = [b-a, c-b, d-c] \rightarrow \mathbf{U}' = [b-a, c-b, c-b, d-c]$

$\mathbf{P} = [p_0, p_1, p_2] \rightarrow \mathbf{P}' = [p'_0, p'_1, p'_2, p'_3]$

This new pair ($\mathbf{P}'$; $\mathbf{U}'$) defines the same NURBS as ($\mathbf{P}$; $\mathbf{U}$) but the curve is no longer a one-piece curve: it is now composed of two pieces. These two pieces are both one-piece curves with the same degree as the original one: there is a self-similarity. To avail this self-similarity, we define two transformations, $\mathcal{L}$ and $\mathcal{R}$, which associate the original NURBS to the left and the right part of the new NURBS. These transformations map a NURBS to another one. This implies they transform both the control polygon and the inter-nodal vector into new ones.

$\mathcal{L} : (\mathbf{P}\ ;\ \mathbf{U}) \mapsto (\mathbf{P}_{\mathcal{L}}\ ;\ \mathbf{U}_{\mathcal{L}})$

$\mathcal{R} : (\mathbf{P}\ ;\ \mathbf{U}) \mapsto (\mathbf{P}_{\mathcal{R}}\ ;\ \mathbf{U}_{\mathcal{R}})$

4.2. Quadratic NURBS CIFS-automaton

The length of the control polygon and the inter-nodal necessary to define a one-piece NURBS vector depends on the degree and so $\mathcal{L}$ and $\mathcal{R}$ have to be defined for every degree. For instance, in quadratic case, they are defined by this way:

$\mathcal{L} : (\mathbf{P}\ ;\ [a,b,c]) \mapsto (\mathbf{P}\mathcal{M}_{\mathcal{L}}(a,b,c)\ ;\ [a,b,b])$

$\mathcal{R} : (\mathbf{P}\ ;\ [a,b,c]) \mapsto (\mathbf{P}\mathcal{M}_{\mathcal{R}}(a,b,c)\ ;\ [b,b,c])$

where $M_{\mathcal{L}}$ and $M_{\mathcal{R}}$ are the matrices defined below. Starting from an inter-node vector $\mathbf{U}$ = [v, w, x], the two transformations $\mathcal{L}$ and $\mathcal{R}$ create two new inter-node vector configurations [v, w, w] and [w, w, x] which are both centered on the same inter-node value (the center value of the parent configuration: w). Each new configuration can be associated with a piece of the two-pieces NURBS, or for self-similarity reason two one-piece NURBS. Both new configurations produce two configurations: themselves (that already exist) and another new uniform configuration: [w, w, w]. At this point, no more configuration can be found: every built configuration is transposed as a state of the quadratic NURBS CIFS-automaton (see Fig. 6).

Transformations are always labeled in a same way but the coefficients of the associated matrices are not always the same: they are parameterized by the three inter-nodes of $\mathbf{U}$ relatively to the starting state of the transition.

These matrices are directly deduced from the weights of branches between the two bottom rows of Fig. 5:

$$\mathcal{M}_{\mathcal{L}}(u_1,u_2,u_3) = \begin{pmatrix} \frac{u_1+2u_2}{2(u_1+u_2)} & \frac{u_2}{2(u_1+u_2)} & 0 \\ \frac{u_1}{2(u_1+u_2)} & \frac{2u_1+u_2}{2(u_1+u_2)} & \frac{u_2+2u_3}{2(u_2+u_3)} \\ 0 & 0 & \frac{u_2}{2(u_2+u_3)} \end{pmatrix}$$

$$\mathcal{M}_{\mathcal{R}}(u_1,u_2,u_3) = \begin{pmatrix} \frac{u_2}{2(u_1+u_2)} & 0 & 0 \\ \frac{2u_1+u_2}{2(u_1+u_2)} & \frac{u_2+2u_3}{2(u_2+u_3)} & \frac{u_3}{2(u_2+u_3)} \\ 0 & \frac{u_2}{2(u_2+u_3)} & \frac{2u_2+u_3}{2(u_2+u_3)} \end{pmatrix}$$

Notice the two matrices have two rows in common. This ensures the continuity of the junction between two pieces of quadratic NURBS.

From both matrices, we can directly compute the position of the fixed-point and the tangent at this point using the following barycentric combinations $\mathcal{B}$ and τ:

$$\begin{cases} \mathcal{B}_{\mathcal{L}} = \left(\frac{u_2}{u_1+u_2}\ ;\ \frac{u_1}{u_1+u_2}\ ;\ 0\right)^{\top} \\ \tau_{\mathcal{L}} = (-1\ ;\ 1\ ;\ 0)^{\top} \end{cases}$$

$$\begin{cases} \mathcal{B}_{\mathcal{R}} = \left(0\ ;\ \frac{u_3}{u_2+u_3}\ ;\ \frac{u_2}{u_2+u_3}\right)^{\top} \\ \tau_{\mathcal{R}} = (0\ ;\ -1\ ;\ 1)^{\top} \end{cases}$$

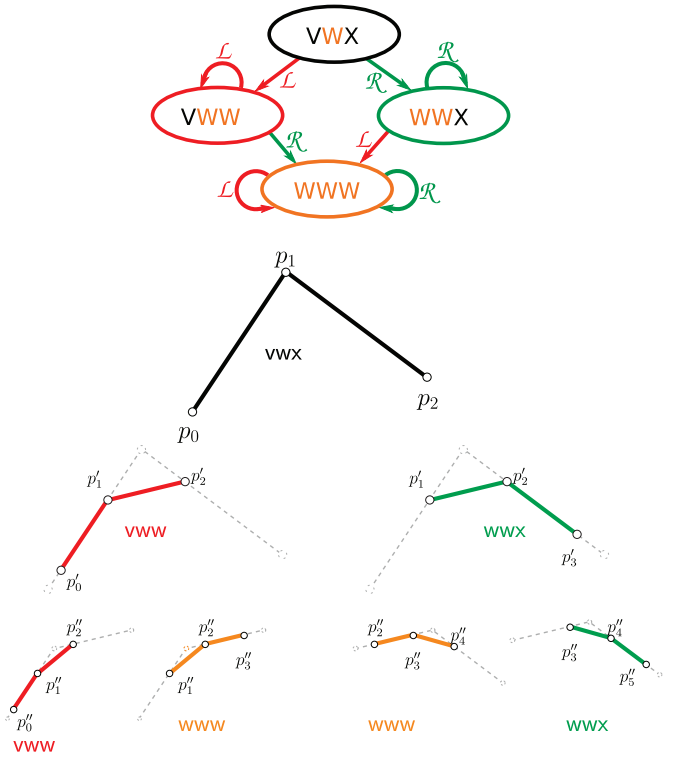

Fig. 6. The quadratic NURBS CIFS-automaton whose starting inter-node vector is [v,w,x]. The states are labeled as states of the corresponding configuration of inter-node vector. Below, the control polygon $\mathbf{P}$ is colored in black. After one iteration, the left and the right parts of the resulting polygons are respectively colored in red for the state vww and in green for the state wwx. For the second iteration, the two middle parts colored in brown define uniform B-Splines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

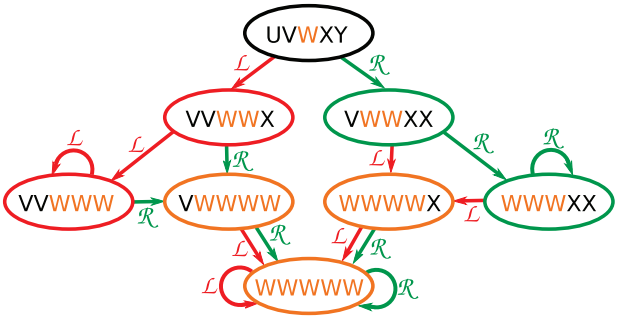

Fig. 7. The automaton used to generate cubic NURBS. Starting internode vector is [u,v,w,x,y] and states are labeled as states of the corresponding internode vector.

4.3. Cubic NURBS CIFS-automaton

Cubic NURBS are handled quite in the same way. They are at least defined by $\mathbf{P} = [p_0 \dots p_3]$ and $\mathbf{T} = [t_0 \dots t_5]$ or $\mathbf{U} = [u_0 \dots u_4]$. The automaton used to generate cubic NURBS is presented in Fig. 7. The transformation matrices, which are given in Appendix B are deduced from the blossoming form in Fig. 8.

4.4. High-degree NURBS CIFS-automaton

In CAGD-engineering, surfaces can be subject to constraints such as interpolating some points, continuity of junctions between surfaces *etc.* An usual way to solve these constraints for NURBS is degree elevation. This elevation involves a management of NURBS

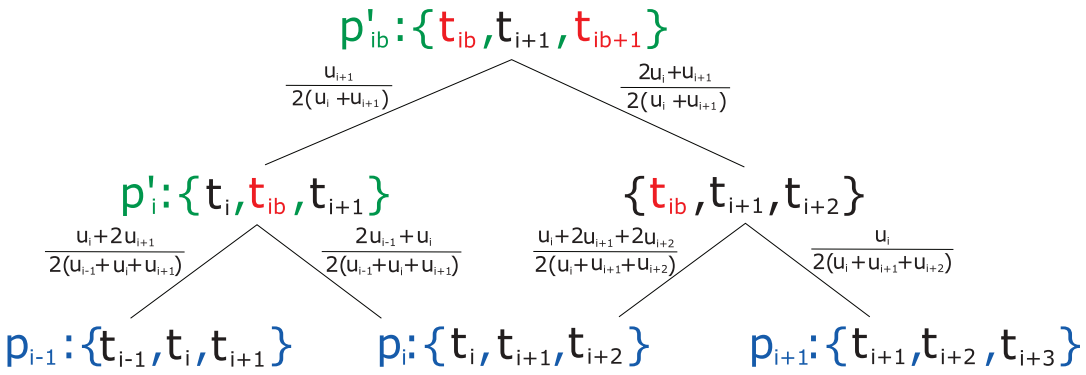

Fig. 8. The computation by blossoming of $\mathbf{P}'$ as a function of $\mathbf{P}$. The convention is $t_{jb} = (t_j + t_{j+1})/2$.

of any-degree, in particular of high-degree because a surface can be subject to many constraints.

A CIFS-automaton is available if and only if its number of states is finite. For NURBS CIFS-automata, each state corresponds to a possible configuration of inter-nodal vector. Before any iteration, this inter-nodal vector is composed of any values. Every iteration duplicates all inter-nodes and then deletes the extreme ones. Every sub-sequence of values (of a fixed length) is a possible state of the inter-nodal vectors. After some iterations, there only remains sub-sequences composed of at most two different values repeated several times. At this point, no more additional states can be found. High-degree automata are directly generated according to this algorithm. Mathematical proofs about the number of states based on a combinatorial approach of the generated words can be found in Appendix A.

5. Extension to NURBS tensor-surfaces

NURBS tensor-surfaces are created by a tensor-product of NURBS. They lie on a regular control grid and own two "orthogonal" inter-nodal vectors **U** and **V**. The automaton used to generate these surfaces can be viewed as a "tensor-product" of two curves-automata. In this section, the tensor-product of automata is explained and then used to generate NURBS tensor-surfaces CIFS automata.

5.1. Tensor-product of finite automata

Let **A** and **B** two finite automata whose states are respectively $\{\alpha_0 \dots \alpha_m\}$ and $\{\beta_0 \dots \beta_n\}$ and whose transitions are respectively $\{a_0 \dots a_p\}$ and $\{b_0 \dots b_q\}$. Let **C** be the automaton which is the "tensor-product" of **A** and **B**. The initial state of **C** is defined as the composition of the initial states of **A** (α_0) and **B** (β_0) and is labeled (α_0; β_0). For every pair of transitions a_i (that links α_0 to α_k) and b_j (β_0 to β_l), a new transition of **C**, $c_{(i,j)}$ is created between the initial state (α_0; β_0) to a potentially new state (α_k; β_l). If this state already exists, the transition is just added to the automaton; otherwise this new state is added to the automaton and its transitions are studied by the same process as the initial state.

From this generation, two properties are straightforward:

- **C** is finite and its numbers of states and transitions are respectively upper-bounded by $(m.n)$ and $(k.l)$.
- If **A** and **B** are complete (*i.e.* every transition is defined for every state), **C** is also complete.

5.2. NURBS-tensor surfaces CIFS automata

In the case where **A** and **B** are CIFS-automata that generate a curve and **C** an automaton of tensor-surface, the term "tensor-product" makes all its sense. Indeed, in this particular case, transformation matrices are tensor-products $\mathcal{M}_{c_{(i,j)}} = \mathcal{M}_{a_i} \otimes \mathcal{M}_{b_j}$. Furthermore, the left-eigenvector of matrices $\mathcal{M}_{c_{(i,j)}}$ are also tensor-product of left-eigenvectors of $\mathcal{M}_{a_i}$ and $\mathcal{M}_{b_j}$. In particular, the fixed point and its associated main-tangents and normal of a transformation γ defined as the tensor-product of two transformations α and β are:

$$\mathcal{B}_\gamma = \mathcal{B}_\alpha \otimes \mathcal{B}_\beta$$
$$\tau_\gamma = \mathcal{B}_\alpha \otimes \tau_\beta$$
$$\tau'_\gamma = \mathcal{B}_\beta \otimes \tau_\alpha$$

Let **A** be an "horizontal" NURBS CIFS-automaton whose transformations are $\mathcal{L}$ and $\mathcal{R}$ and **B** a "vertical" NURBS automaton whose transformations are respectively changed to $\mathcal{D}$ and $\mathcal{U}$. Four transformations are defined by the tensor-product of **A** and **B**:

$\otimes$	$\mathcal{L}$	$\mathcal{R}$
$\mathcal{U}$	$\mathcal{T}_3 = \mathcal{U} \otimes \mathcal{L}$	$\mathcal{T}_2 = \mathcal{U} \otimes \mathcal{R}$
$\mathcal{D}$	$\mathcal{T}_0 = \mathcal{D} \otimes \mathcal{L}$	$\mathcal{T}_1 = \mathcal{D} \otimes \mathcal{R}$

By using the tensor-product of automata presented in the previous sub-section, the bi-quadratic and bi-cubic automata are generated (see Fig. 9). Some examples of NURBS tensor-surfaces generated from this CIFS automata are given in Fig. 10.

Even if only low degree automata are presented, this method applies to any degree NURBS tensor-surfaces. The only pitfall to face is the explosion of the number of states that comes when the degree increases. This issue is inherent to NURBS: the recursive process of generation always implies difficulties related to the degree.

By generating several tensor automata from curves-automata of different degree, some properties appear. The first one is once a state is leaved, it is impossible to turn back to it. The second one is the number of stationary states (*i.e.* states that own at least one transition to themselves) is always 3 for curves-automata (labeled as left, right, and uniform); other states are transitory: they are active only once. The number of stationary states of a tensor-automaton is the product of the number of stationary states of the two original automata: it is 3 for curves, 9 for surfaces, 27 for volumes. Whatever the number of stationary states, there is always only one state, called the uniform state, which loops to itself for all transformations. This uniform state is the IFS that generates uniform B-Splines shapes (curve, surface, volume).

6. Applications

The representation of NURBS as CIFS-automata permits to push further some previous works. In this section, several CIFS-based

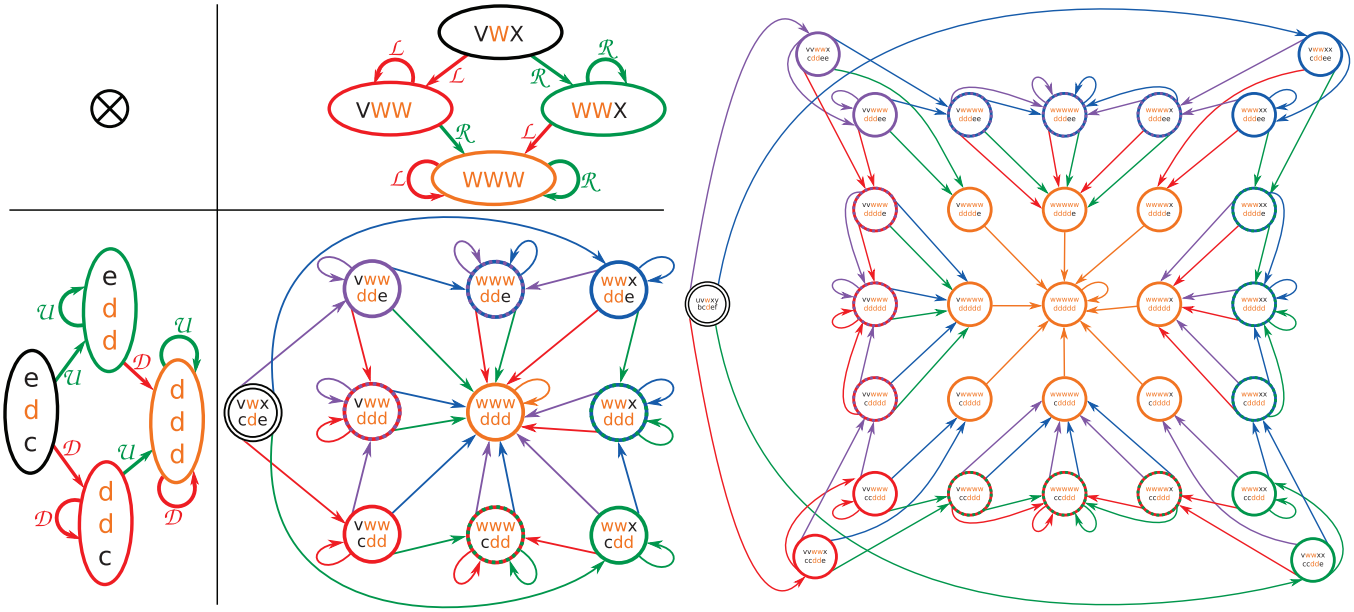

Fig. 9. On the left, the construction of bi-quadratic NURBS CIFS-automaton by tensor product of two quadratic NURBS CIFS-automata **A** and **B**. On the right the bi-cubic NURBS CIFS-automaton (without the input curve automata). Transformations $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ are respectively represented by red, green, blue, and purple arrows. Brown arrows represent any of these transformations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

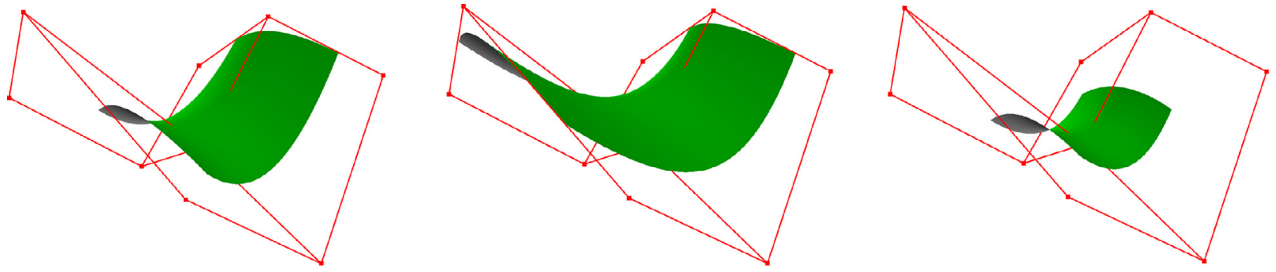

Fig. 10. Three examples of bi-quadratic NURBS tensor-surfaces with the same control grid but with different associated inter-nodal vectors. From left to right: $(1, 2, 3) \times (0, 3, 2)$, $(0, 1, 0) \times (1, 3, 8)$ and a uniform one $(1, 1, 1) \times (1, 1, 1)$.

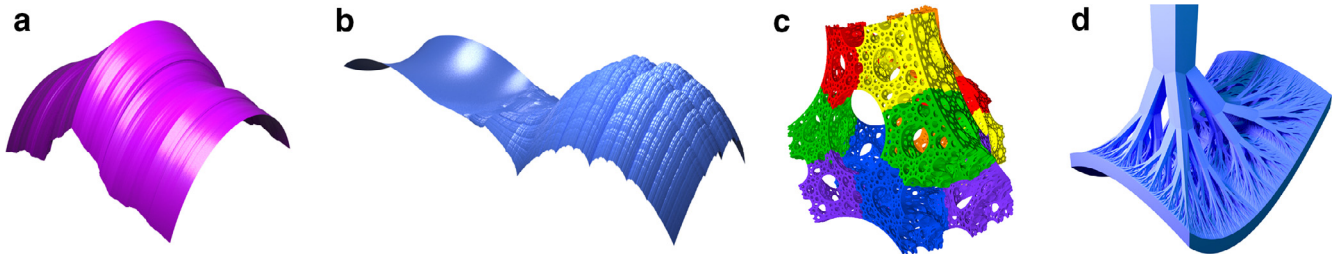

Fig. 11. From left to right: a tensor-surface generated from a NURBS and a Takagi curve, a junction between a NURBS bi-cubic tensor-surfaces and a Takagi tensor-surfaces, a lacunar volume with highlight subdivision process, and a tree iteratively defined such that its leaves correspond to the NURBS surface of Fig. 10 (left). All were designed using the iterative modeler MODITERE – LIB.

applications, currently limited to uniform B-Splines, are henceforth extended to non-uniform case.

6.1. Tensor surfaces with blending natures

Once CIFS-automaton tensor-product is defined, it can be used to generate tensor-surfaces from two generative curves which are not of the same nature, with the only condition that both are defined as CIFS-automata. For instance, in Fig. 11(a) is presented a tensor-surface generated from a cubic NURBS and a Takagi curve. Thanks to the overlaps of Takagi curve, the surface is mechanically more resistant than an usual NURBS tensor-surface while keeping its smoothness in one direction to evacuate fluids.

6.2. Junctions between surfaces of different natures

Podkorytov et al. [11] proves that a junction J between surfaces of different natures (A and B) is always possible since they are both represented as CIFS automata. These two surfaces are both bounded by a curve which is defined by a subset of the control mesh. For instance bi-quadratic NURBS tensor-surfaces, which are defined at least by a (3×3)-grid, are bounded by a quadratic NURBS which is defined by a (3×2)-grid. Both bounding-curves meshes are merged in a new CIFS of six transformations: two of them map the new mesh to a mesh of type A, two others map the mesh to a mesh of type B, and the two last ones map the mesh to a mesh of the new type J (see Fig. 12).

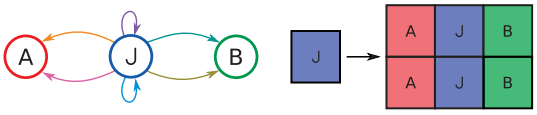

Fig. 12. Generic junction J between two surfaces of different types A and B. On left side, the CIFS automaton where A and B states are the usual CIFS automata of corresponding surfaces. On right side, the topological subdivision of J into six subsurfaces of the three types.

The transformations that create surface of type A or B are constrained to conserve the continuity at the junction with bounding surfaces. Central transformations are created depending on the four others in order to ensure a maximum continuity around the junction. If the central transformations are over-constrained, some phantom points can be added in the central mesh in order to release these constraints. An example can be found in Fig. 11(b) where a NURBS tensor-surface is joined to a Takagi tensor-surfaces.

6.3. Topological optimization

Some CAGD-pieces are subject to optimization while respecting constraints. For instance, they should be as light as possible and must efficiently evacuate heat without affecting their strength. A solution that directly proceeds on surfaces is the creation of lacunar surfaces (an example of lacunar NURBS-tensor surfaces is given in Fig. 11(c)). Another challenge that needs topological optimization is the creation of a support subject to the same constraints as described before. Arborescent structure that join a CIFS automaton have been proposed by Sokolov et al. [18] as a solution for uniform B-Splines tensor-surfaces. For instance, the tree in Fig. 11(d) is defined as a volumic subdivision bordered by a bi-quadratic NURBS tensor-surface. Constrains are defined to ensure that the subdivision of the volume is compatible with the subdivision of the surface.

7. Conclusion

In the present article, we show that NURBS curves and tensor-surfaces can be represented with a Controlled Iterated Function System (CIFS). As most of CAGD-systems are NURBS-based, adding these new surfaces to the CIFS-model is an important progress to a transition from usual CAGD-engineering to a CIFS-modeling.

We show the associated automaton is finite, and composed of limited number of stationary states. Furthermore, we provide two methods: the first one generates CIFS-automata that correspond to NURBS of any degree and the second one explains how to transpose tensor-product from curves to automata.

Once NURBS tensor-surfaces defined as CIFS automata, many interactions between them and object that are already defined in the formalism (fractals, subdivision surfaces...) become possible or easier.

The main issue NURBS have to face is the management of extraordinary vertices that are usually unavoidable in CAGD-engineering. Sederberg et al. [19] and Müller et al. [20] proposed another solution which is Non-Uniform Rational Subdivision Surfaces but have been restricted to low-degree (quadratic or cubic) surfaces. Thereafter, Cashman et al. [21] found a way to generalize [19] to high-degree surfaces but only for odd degree.

For this purpose, we need to manage non-tensor NURBS surfaces. Corresponding CIFS-automata should handle the evolution of both knot-vectors (which are no longer limited to two “orthogonal” ones) and the topology of the mesh. For quadratic and cubic cases, the shape (states and transitions) of automata seems to be at the cross of this paper for the non-uniformity and [10] for the management of extraordinary vertices. Coefficients of matrices associated with transitions should be deduced from [19].

Even if low-degree cases may be quite direct to integrate, a real issue will take place with higher degree. This comes from the fact that the neighborhood of vertices needed to compute a piece of surface is composed of a central vertex (for quadratic case) or a central face (for cubic case) and a unique ring of vertices that surrounds it. This ring is composed of vertices that share a face with the central vertex or a vertex of the central face. In these cases, the possibilities of irregularities are very limited and can be all listed. For higher degree, the patch owns several nested rings; thus the possibilities of irregularities are subject to combinatorial explosion. The solution proposed by Cashman et al. [21] for any odd-degree should be closely studied to find a solution to fill this gap.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Aknowledgments

We would like thank Gilles Gouaty for the implementation of automata, and Dominique Michelucci for his helpful comments and suggestions for scientific coherence and English correctness.

Appendix A. Mathematical proof

We fix $n = 2d - 1$ for $d \geq 1$. Let $\mathcal{M}_n$ be the set of words of length $n \geq 1$ over the alphabet $\{1, 2, \ldots, n\}$, *i.e.*, all words $w = w_1 w_2 \ldots w_n$ with $w_i \in \{1, 2, \ldots, n\}$ for $1 \leq i \leq n$. w_i corresponds to the ith letter of the word w and $\forall i, w_i \leq w_{i+1}$ The conjugate of a word $w = w_1 \ldots w_n$ is the word $\overline{w} = n + 1 - w_n, n + 1 - w_{n-1}, \ldots, n + 1 - w_1$.

From a word $w \in \mathcal{M}_n$, we construct the word $W \in \mathcal{M}_{2n}$ defined by $W = w_1 w_1 w_2 w_2 \ldots w_n w_n$. Note that $W_n = W_{n+1} = w_d$. Also, we define the maps $\mathcal{L}$ and $\mathcal{R}$ on $\mathcal{M}_n$ as follows:

$$\begin{cases} \mathcal{L}(w_1 \ldots w_n) = W_{n-d+1} \ldots W_{n-2} W_{n-1} W_n W_{n+1} \ldots W_{n+d-1}, \\ \mathcal{R}(w_1 \ldots w_n) = W_{n-d+2} \ldots W_{n-1} W_n W_{n+1} W_{n+2} \ldots W_{n+d}. \end{cases}$$

For instance, whenever $n = 7$ (or equivalently $d = 4$) and $w = 1234567$, we have $W = 11223344556677$, $\mathcal{L}(w) = 2334455$, $\mathcal{R}(w) = 3344556$, $\mathcal{L}(\mathcal{L}(w)) = 3334444$ and $\mathcal{R}(\mathcal{L}(w)) = 3344445$.

For $n \geq 1$, let us consider the set $\mathcal{S}_n$ of words obtained from $12 \ldots n$ after all possible combinations of $\mathcal{L}$ and $\mathcal{R}$. For instance, $\mathcal{S}_7 = \{1234567, 2334455, 3344556, 3334444, 3344445, 3444455, 4444555, 3444444, 3344444, 4444445, 4444455, 4444444\}$.

It is straightforward to check the following properties:

Fact 1. Any word w in $\mathcal{S}_n$ satisfies $w_d = d$.

Fact 2. For $1 \leq i \leq n - 1$, two consecutive letters w_i and w_{i+1} in a word $w \in \mathcal{S}_n$ satisfy $0 \leq w_{i+1} - w_i \leq 1$.

Fact 3. For any $w \in \mathcal{S}_n$, its conjugate $\overline{w}$ also lies in $\mathcal{S}_n$.

Fact 4. (i) d^n is the unique word w where $\mathcal{L}(w) = \mathcal{R}(w) = w$.

(ii) $(d-1)^{d-1} d^d$ is the unique word w where $\mathcal{L}(w) = w$ and $\mathcal{R}(w) \neq w$.

(iii) $d^d (d+1)^{d-1}$ is the unique word w where $\mathcal{R}(w) = w$ and $\mathcal{L}(w) \neq w$.

Fact 5. For any $w \in \mathcal{S}_n$, the number of letters d in $\mathcal{L}(w)$ (resp. $\mathcal{R}(w)$) is at least that of w.

Lemma 1. *Any word $w \in \mathcal{S}_n$ is a subword of $m_k = 1^{2^k} 2^{2^k} \ldots (n-1)^{2^k} n^{2^k}$ for some $k \geq 0$.*

Proof. Let w be a word obtained after applying $r \geq 0$ consecutive transformations $\mathcal{L}$ and/or $\mathcal{R}$, and let us prove by induction on r that w is a subword of $m_k = 1^{2^k}2^{2^k}\dots(n-1)^{2^k}n^{2^k}$ for some k. Obviously, if $r = 1$ then w is one of the two words $\mathcal{L}(12\dots n)$ and $\mathcal{R}(12\dots n)$, which are subwords of $m_1 = 1122\dots nn$ (by definition of $\mathcal{L}$ and $\mathcal{R}$). Let us assume that w is subword of m_k with $w_d = d$, and let us prove that $\mathcal{L}(w)$ and $\mathcal{R}(w)$ are some subwords of m_{k+1}.

By definition, $w' = \mathcal{L}(w)$ (resp. $w' = \mathcal{R}(w)$) is obtained as a subword of $W = w_1w_1w_2w_2\dots w_nw_n$ after matching w'_d with $W_n = w_d$ (resp. $W_{n+1} = w_d$). Since w is a subword of m_k such that $w_d = d$, this implies that $W = w_1w_1\dots w_nw_n$ is a subword of m_{k+1}, and thus $w' = \mathcal{L}(w)$ (resp. $w' = \mathcal{R}(w)$) is a subword of m_{k+1}. □

Lemma 2. *Let w be a subword of length n of m_k, $k \geq 0$, such that $w_d = d$. Then, w belongs to $\mathcal{S}_n$.*

Proof. The proof is obtained by induction on k. Obviously, when $k = 0$ the unique length n subword of $12\dots(n-1)n$ is $12\dots n$, and it belongs to $\mathcal{S}_n$. For $k = 1$, w is a subword of $W = 1122\dots dd\dots nn$ with $w_d = W_n = W_{n+1} = d$. In the case where w_d matches W_n, $w = \mathcal{L}(12\dots n)$; otherwise w_d matches W_{n+1} and we have $w = \mathcal{R}(12\dots n)$. In the two cases, w lies into $\mathcal{S}_n$.

Now, let us assume the statement holds for $i \leq k$, and let us prove it holds for $k+1$. Let w be a length n subword of m_{k+1} such that $w_d = d$. Let $r \geq 1$ so that w_d matches the rth value of m_{k+1}. Since $w_d = d$ we necessarily have $r \geq (d-1)2^{k+1} \geq n$, and $r \leq n2^{k+1} - (d-1)2^{k+1} < n2^{k+1} - n$. These two last inequalities ensure that there is a length $2n$ subword W of m_{k+1} such that W_n or W_{n+1} matches w_d. In the case where w_d matches W_n, then we define the subword v of length n by deleting all values of odd ranks in W, and v satisfies $\mathcal{L}(v) = w$. In the case where w_d matches W_{n+1}, then we define the subword v of length n by deleting all values of even ranks in W, and v satisfies $\mathcal{R}(v) = w$. By construction, v is a subword of m_k with $v_d = d$ and after applying the recurrence hypothesis, v belongs to $\mathcal{S}_n$. As w is either $\mathcal{R}(v)$ or $\mathcal{L}(v)$, we deduce that w also is in $\mathcal{S}_n$. We conclude by induction. □

Theorem 1. *For $d \geq 1$, the cardinality of the set $\mathcal{S}_{2d-1}$ is $4(d-1)$.*

Proof. Let $r \geq 1$ be the integer such that $2^{r-1} < n = 2d-1 < 2^r$, *i.e.* $r = \lceil \log_2(n) \rceil$. Using the previous two lemmas, it suffices to count the subwords w of length n in m_k, $k \geq 0$, satisfying $w_d = d$. From $k = 0$ to $r-1$, the number of such subwords in m_k is 2^k since such a word contains exactly 2^k letters d, and we have 2^k ways to fix $w_d = d$.

Now, let us count the subwords having exactly α letters d for $\alpha > 2^{r-1}$. It is straigthforward to see that such a word is a subword of m_r and for $\alpha < n$, there are two ways to choose such a word (either $w_d = d$ is the leftmost d in m_r or $w_d = d$ is the rightmost d in m_r). Then there are $2(n-1-2^{r-1})$ possible words whenever $2^{r-1} < \alpha < n$. Finally, for $\alpha = n$, there is only one subword, that is d^n.

Combining all these subcases, the number of words in $\mathcal{S}_n$ is:

$$\sum_{k=0}^{r-1} 2^k + 2(n-1-2^{r-1}) + 1 = 2^r - 1 + 2(n-1) - 2^r + 1 \qquad \text{(A.1)}$$

$$= 2(2d-2) = 4(d-1) \qquad \text{(A.2)}$$

□

Corollary 1. *From $12\dots n$, we can reach any word in $\mathcal{S}_n$ in at most $\lceil \log_2(n) \rceil$ transformations $\mathcal{L}$ and/or $\mathcal{R}$. We need $\lceil \log_2(n) \rceil$ transformations to reach the word d^n.*

Proof. Let $r \geq 0$ be the integer such that $2^{r-1} < n < 2^r$. We have $r = \lceil \log_2(n) \rceil$. Using the fact that d^n is obtained as a subword of 2^r, we need r transformations to reach it. From the proof of Theorem 1, all others words can be reached with at most r transformations, which gives the results. □

Corollary 2. *Let w be a word in $\mathcal{S}_n$ such that $w \notin \{d^n, (d-1)^{d-1}d^d, d^d(d+1)^{d-1}\}$. It is impossible to find a sequence ξ of $\mathcal{L}$ and $\mathcal{R}$ such as $w = \xi(w)$.*

Proof. Since w does not belong to $\{d^n, (d-1)^{d-1}d^d, d^d(d+1)^{d-1}\}$, Fact 4 proves that $\mathcal{R}(w)$ and $\mathcal{L}(w)$ differ from w. Let r be the number of letters d in w. Then, $\mathcal{L}(w)$ (resp. $\mathcal{R}(w)$) has at least one letter d more than w, and since $\mathcal{L}$ and $\mathcal{R}$ cannot decrease the number of d (see Fact 5), the proof is completed. □

Appendix B. Cubic matrices

$$M_{\mathcal{L}}(u_0\ \dots\ u_4) = \begin{pmatrix} \frac{u_1+2u_2}{2(u_0+u_1+u_2)} & \frac{u_2}{2(u_1+u_2)}\frac{u_1+2u_2}{2(u_0+u_1+u_2)} & 0 & 0 \\ \frac{2u_0+u_1}{2(u_0+u_1+u_2)} & \frac{u_2}{2(u_1+u_2)}\frac{2u_0+u_1}{2(u_0+u_1+u_2)} + \frac{2u_1+u_2}{2(u_1+u_2)}\frac{u_1+2u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{u_3}{2(u_2+u_3)}\frac{u_2+2u_3}{2(u_1+u_2+u_3)} \\ 0 & \frac{2u_1+u_2}{2(u_1+u_2)}\frac{u_1}{2(u_1+u_2+u_3)} & \frac{2u_1+u_2}{2(u_1+u_2+u_3)} & \frac{u_3}{2(u_2+u_3)}\frac{2u_1+u_2}{2(u_1+u_2+u_3)} + \frac{2u_2+u_3}{2(u_2+u_3)}\frac{u_2+2u_3+2u_4}{2(u_2+u_3+u_4)} \\ 0 & 0 & 0 & \frac{2u_2+u_3}{2(u_2+u_3)}\frac{u_2}{2(u_2+u_3+u_4)} \end{pmatrix}$$

$$M_{\mathcal{R}}(u_0\ \dots\ u_4) = \begin{pmatrix} \frac{u_2}{2(u_1+u_2)}\frac{u_1+2u_2}{2(u_0+u_1+u_2)} & 0 & 0 & 0 \\ \frac{u_2}{2(u_1+u_2)}\frac{2u_0+u_1}{2(u_0+u_1+u_2)} + \frac{2u_1+u_2}{2(u_1+u_2)}\frac{u_1+2u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{u_3}{2(u_2+u_3)}\frac{u_2+2u_3}{2(u_1+u_2+u_3)} & 0 \\ \frac{2u_1+u_2}{2(u_1+u_2)}\frac{u_1}{2(u_1+u_2+u_3)} & \frac{2u_1+u_2}{2(u_1+u_2+u_3)} & \frac{u_3}{2(u_2+u_3)}\frac{2u_1+u_2}{2(u_1+u_2+u_3)} + \frac{2u_2+u_3}{2(u_2+u_3)}\frac{u_3+2u_4}{2(u_2+u_3+u_4)} & \frac{u_3+2u_4}{2(u_2+u_3+u_4)} \\ 0 & 0 & \frac{2u_2+u_3}{2(u_2+u_3)}\frac{u_2}{2(u_2+u_3+u_4)} & \frac{2u_2+u_3}{2(u_2+u_3+u_4)} \end{pmatrix}$$

References

[1] Peitgen H-O, Richter P. The beauty of fractals: Images of complex dynamical systems. Springer-Verlag, Heidelberg; 1986.
[2] Zhou H, Sun J, Turk G, Rehg JM. Terrain synthesis from digital elevation models. IEEE Trans Vis Comput Graph 2007;13(4):834–48.
[3] Prusinkiewicz P, Lindenmayer A. The algorithmic beauty of plants. New York, NY, USA: Springer-Verlag New York, Inc.; 1990. ISBN 0-387-97297-8.
[4] Hutchinson JE. Fractals and self similarity. Indiana Univ Math J 1981;30(5):713–47.
[5] Prusinkiewicz P, Hammel M. Language-restricted iterated function systems, koch constructions, and l-systems. In: Proceedings of the SIGGRAPH'94 course notes; 1994.
[6] Barnsley M. Fractals everywhere. Academic Press; 2014.
[7] Goldman R. The fractal nature of Bézier curves. In: Proceedings of the geometric modeling and processing, 2004. IEEE; 2004. p. 3–11.
[8] Schaefer S, Levin D, Goldman R. Subdivision schemes and attractors. In: Proceedings of the symposium on geometry processing. Citeseer; 2005. p. 171–80.
[9] Zair CE, Tosan E. Fractal modeling using free form techniques. Comput Graph Forum 1996;15:269–78.
[10] Morlet L, Neveu M, Lanquetin S, Gentil C. Barycentric combinations based subdivision shaders. Plzen, Czech Republic. J WSCG 2018;25.
[11] Podkorytov S, Gentil C, Sokolov D, Lanquetin S. Joining primal/dual subdivision surfaces. In: Proceedings of the eighth international conference mathematical methods for curves and surfaces, MMCS 2012, Oslo, Norway, June 28–July 3, 2012, Revised Selected Papers; 2012. p. 403–24. doi:10.1007/978-3-642-54382-1_23.
[12] Halstead M, Kass M, DeRose T. Efficient, fair interpolation using Catmull–Clark surfaces. In: Proceedings of the twentieth annual conference on computer graphics and interactive techniques. ACM; 1993. p. 35–44.
[13] Stam J. Exact evaluation of Catmull–Clark subdivision surfaces at arbitrary parameter values. In: Proceedings of the twenty-fifth annual conference on computer graphics and interactive techniques SIGGRAPH '98. New York, NY, USA: ACM; 1998. p. 395–404. ISBN 0-89791-999-8. doi:10.1145/280814.280945.
[14] Cox MG. The numerical evaluation of B-splines. IMA J Appl Math 1972;10(2):134–49.
[15] Ramshaw L. Blossoming: A connect-the-dots approach to splines. Digital Equipment Corporation Palo Alto; 1987.
[16] Boehm W. Inserting new knots into b-spline curves. Comput Aided Des 1980;12(4):199–201.
[17] Cohen E, Lyche T, Riesenfeld R. Discrete b-splines and subdivision techniques in computer-aided geometric design and computer graphics. Comput Graph Image Process 1980;14(2):87–111.
[18] Sokolov D, Gouaty G, Gentil C, Mishkinis A. Boundary controlled iterated function systems. In: Boissonnat J-D, Cohen A, Gibaru O, Gout C, Lyche T, Mazure M-L, et al., editors. Curves and surfaces. Cham: Springer International Publishing; 2015. p. 414–32.
[19] Sederberg TW, Zheng J, Sewell D, Sabin M. Non-uniform recursive subdivision surfaces. In: Proceedings of the twenty-fifth annual conference on Computer graphics and interactive techniques. ACM; 1998. p. 387–94.
[20] Müller K, Reusche L, Fellner D. Extended subdivision surfaces: building a bridge between NURBS and Catmull–Clark surfaces. ACM Trans Graph (TOG) 2006;25(2):268–92.
[21] Cashman TJ, Augsdörfer UH, Dodgson NA, Sabin MA. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. ACM Trans Graph (TOG) 2009;28(3):46.