Digital Surface of Revolution with Hand-Drawn Generatrix

Éric Andrès, Lydie Richaume, Gaëlle Largeteau-Skapin

Accepted in JMIV manuscript No.
(Accepted by JMIV - This is a self-edited version by the authors)

Digital Surface of Revolution with Hand-Drawn Generatrix

Digital Surface of Revolution

Eric Andres · Lydie Richaume · Gaelle Largeteau-Skapin

not yet published

Abstract In this paper we present a simple method to create general 3D digital surfaces of revolution based on a 2D implicit curve of revolution (therefore not limited to a circle) and a hand-drawn generatrix. Our method can handle any sequence of Euclidean 2D points, that represents a curve, as generatrix. One can choose the topology of the surface that may have 1-tunnels, 0-tunnels or no tunnels with applications in 3D printing for instance. An online tool that illustrates the method is proposed.

Keywords digital surfaces, implicit functions, surface of revolution.

1 Introduction

Surfaces of revolution (as a particular class of sweep surfaces), are an important class of surfaces in computer modeling [1]. It allowed early on, with limited user interface capabilities, to design common 3D objects based on 2D forms (a 2D circle and 2D generatrix): pen, glass, flower pot, chess piece, ... Moreover, fundamental surfaces such as spheres, cylinders, torii, ... can be defined as surfaces of revolution. A surface of revolution is defined by a 2D *revolution curve* (classically a circle) and a curve as profile, most often an explicit function, called *generatrix*. In this paper we are interested in *digital surfaces of revolution* with a free-shaped generatrix and an implicitly defined curve of revolution with some control on the topological properties of the digital surface. The digital surface of revolution is defined by a flake digitization method which is defined as all the flakes centered on digital points cut by the continuous surface of revolution.

In our DGCI 2016 paper [2], we proposed a flake based simple digital surface of generation tool with a generatrix limited to explicit functions $y = g(z)$ (see section 1.1 and section 2.3 for more details). This was somewhat limiting. One of the motivations for our work on digital surfaces is to propose flexible generation tools for the general public and a local artist, Aurélie Mourier, who works with us. Classically for an explicitly defined generatrix $y = g(z)$, for each z value, there is one and only one y value. In the present paper, the generatrix can be any curve represented by an ordered sequence of 2D continuous points, therefore there can be more than one y value for a given z. This may correspond to a hand-drawn curve which was the main motivation for this work (see section 4.1), an open or closed sampled parametric curve (section 4.2), or a 2D digital curve (section 4.3).

In order to handle such generatrices, we devised a surface of revolution generation algorithm with the following steps (see section 3.4):

- The generatrix can be any finite and open sequence of points. The generatrix is divided into monotonic explicit or horizontal point sequences. Each generatrix section can then be digitized independently. This works because our method is based on morphological type digitizations: we ensure that way that the *union of the digitizations* is the same as the *digitization of the unions*.
- For horizontal generatrix sections (see section 3.3) we simply need the minimum and maximum y value in the sequence.
- For monotonic explicit sequences of points defining a section of the generatrix, the sequence is recomputed in order to have a sequence of y for integer and

Université de Poitiers, Laboratoire XLIM, SIC, UMR CNRS 7252, BP 30179, F-86962 Futuroscope Chasseneuil, France
E-mail: {eric.andres,gaelle.largeteau.skapin}@univ-poitiers.fr, lydie.richaume@etu.univ-poitiers.fr

half-integer z values (see section 3.1.1, section 3.1.2 and section 3.2.1). The flake digitization method we are using only requires to compute y values for integer and half-integer z values of the generatrix. This allows a direct $O(1)$ constant time look-up (see section 3.1.2) for the generatrix curve values.
- For all $(x, y, z) \in \mathbb{Z}^3$ in the generation window, for each generatrix section, we test if the corresponding flake centered in (x, y, z) is cut by the continuous surface of revolution. If it is, the voxel of coordinates (x, y, z) belongs to the digital surface. There are however some difficulties that need to be handled. A monotonic generatrix section may be defined only on a finite z domain and thus only partially cut some flakes. We introduce in section 3.2.2 and section 3.2.3 the notion of cropped flake and cropped adjacency ball that can handle such partial cuts. Without this, there could be holes at the junction of two generatrix sections in the surface since the end of one monotonic sequence will typically be the beginning of the next monotonic or horizontal generatrix section. Note that the generatrix curve does not have to be a closed curve.

To illustrate our method, we propose an online tool (that can be found at: `http://xlim-sic.labo.univ-poitiers.fr/demonstrateurs/DSoR_Generator/?lang=en`), that was used for most of the examples in this paper, in particular for a 3D printed chess queen (see Figure 9). After some basic notations and the analytical implicit surface digitization method in section two, we present, in section three, how we handle an explicit finite or an horizontal generatrix. In section four we propose some results, limitations, an algorithm and we conclude and provide perspectives in section five.

1.1 Previous work

There are relatively few papers that have dealt specifically with the problem of defining or generating digital surfaces of revolution. N. Stolte et al. have defined a generation method for digital implicit surfaces with cylindrical and spherical coordinates [3,4]. Their goal was not so much to define a generation method for digital surfaces as to propose a visualization method for implicit surfaces. Their method is limited to supercover type surfaces (tunnel-free surfaces) that are defined with cylindrical and spherical coordinates. More recently, G.Kumar and P.Bhowmick [5,6] proposed a generation algorithm for digital surfaces of revolution for the design of virtual pottery. Their idea was to superpose 2D digital annuli since a rasterized horizontal slice of height one of a surface of revolution corresponds to a 2D digital annuli. The authors worked with a classical notion akin to the Bresenham circle [7], defined only for integer radii and center. The main drawback is that concentric Bresenham circles of increasing radii leave points that do not belong to any circle. The authors therefore determine what they call *absentee* voxels, to fill-up the 6-connected holes in the digital surface [8]. The method for filling the holes with absentee voxels is however very complicated and limited to circular curves of revolution. One way around this problem could have been to use Andres circles [9,10,11] that do fill space and are defined for arbitrary center and radii. Their method defines only naive type surfaces (2-tunnel free surfaces). It can be note however, that they allow a general form of generatrix akin to the type of generatrix we are proposing in the present paper.

A surface of revolution can be defined as an implicit surface [1]. This was the basis for our DGCI 2016 paper [2], where we proposed an approach based on a morphology type digitization method for implicit nD surfaces [12,13] (generalizing the approach of Laine [14]). The digital surface of revolution presented in [2] was defined by an explicit function $y = g(z)$ as generatrix and a 2D implicit function $r(x, y) = 0$ as curve of revolution (not limited to 2D circles). The method is very simple and straight-forward to implement. Furthermore, under some regularity constraints [12], the digitization method offers a control over the digital surfaces' topology by defining k-tunnel free digital surfaces. This allows the design of digital surfaces adapted to specific applications: for visualization purposes, the thinnest, so called naive surface, 2-tunnel free surface might be adequate. For 3D printing however, one might want to choose 2-connected, so called supercover, tunnel free surfaces. The present paper is an extension of this last paper.

2 Notations, basics and recalls

Let $\{e_1, \ldots, e_n\}$ denote the canonical basis of the n-dimensional Euclidean vector space. Let $\mathbb{Z}^n$ be the subset of $\mathbb{R}^n$ that consists of all the integer coordinate points. A *digital (resp. Euclidean) point* is an element of $\mathbb{Z}^n$ (resp. $\mathbb{R}^n$). We denote by x_i the i-th coordinate of a point x, that is its coordinate associated to e_i. In 3D, for the sake of simplicity, we will often refer to the classical xyz coordinate system corresponding to $\{e_1, e_2, e_3\}$. A *digital (resp. Euclidean) object* is a set of digital (resp. Euclidean) points. The Chebychev norm of a vector $\mathbf{x}$ is defined by $\|\mathbf{x}\|_\infty = \max(|x_i|)$. The Manhattan norm of a vector $\mathbf{x}$ is defined by $\|\mathbf{x}\|_1 = \sum |x_i|$.

For all $k \in \{0, \ldots, n-1\}$, two integer points v and w are said to be *k-adjacent* or *k-neighbors*, if for all $i \in \{1, \ldots, n\}$, $|v_i - w_i| \leq 1$ and $\sum_{j=1}^{n} |v_j - w_j| \leq n - k$. In the 3-dimensional space, the 0-, 1- and 2-neighborhood notations correspond respectively to the classical 26-, 18- and 6-neighborhood notations.

A *k-path* is a sequence of integer points such that every two consecutive points in the sequence are k-adjacent. A digital object E is k-connected if there exists a k-path in E between any two points of E. A maximum k-connected subset of E is called a *k-connected component*. If the complement of a digital object E, $\mathbb{Z}^n \setminus \mathrm{E}$ admits exactly two k-connected components C_1 and C_2, *i.e.* there exists no k-path joining integer points of C_1 and C_2, then E is said to be *k-separating* in $\mathbb{Z}^n$.

Let $\oplus$ be the Minkowski addition, known as dilation, such that $\mathcal{A} \oplus \mathcal{B} = \cup_{b \in \mathcal{B}} \{a + b : a \in \mathcal{A}\}$.

2.1 Adjacency norms

The adjacency norms have been introduced in [15]. Every digital adjacency relationship can be associated to a norm called an *adjacency norm* defined as follows:

Definition 1 (Adjacency norms [15]) Let n be the dimension of the space. Let k be a positive integer lower than n. The *k-adjacency norm* $[\cdot]_k$ is defined as:

$$\forall x \in \mathbb{R}^n, [x]_k = \max \left\{ \|x\|_\infty, \frac{\|x\|_1}{n-k} \right\}.$$

The name *adjacency norms* stems for the following property:

Lemma 1 (Digital adjacency and adjacency norms [15]) *Let v and $w \in \mathbb{Z}^n$. Then, v and w are k-adjacent* iff $[v - w]_k \leq 1$.

Let $\mathcal{B}_{[\cdot]_k}(\frac{1}{2})$ be the ball of radius $\frac{1}{2}$ under the norm $[\cdot]_k$. The associated distance is denoted by d_k. It is easy to see that the 0-adjacency norm corresponds to the Chebychev norm and the $(n-1)$-adjacency norm to the Manhattan norm.

2.2 Implicit surface digitization

In this paper we are considering morphological digitization schemes [13,15,16,17]. A summary on this type of digitization schemes and how they can be used to define digital objects can be found in [13]. The basic idea is to say that a digital object is defined as the digital points inside a Minkowksy sum between a structuring element E and the, *to be digitized*, continuous object S:

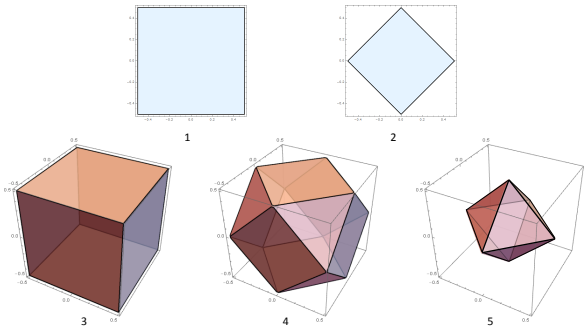

Fig. 1 (1,2) 2D adjacency balls $\mathcal{B}_{[\cdot]_0}(\frac{1}{2})$ and $\mathcal{B}_{[\cdot]_1}(\frac{1}{2})$; (3,4,5) 3D adjacency balls $\mathcal{B}_{[\cdot]_0}(\frac{1}{2})$, $\mathcal{B}_{[\cdot]_1}(\frac{1}{2})$ and $\mathcal{B}_{[\cdot]_2}(\frac{1}{2})$.

Definition 2 (Morphological Digitization) Let us consider an object S and a structuring element E both defined in $\mathbb{R}^n$. The morphological digitization of S by E is defined by :

$$\mathcal{D}_E(S) = (S \oplus E) \cap \mathbb{Z}^n = \{v \in \mathbb{Z}^n : (v \oplus E) \cap S \neq \varnothing\}.$$

Definition 3 (Adjacency Digitization [12,2,15]) The k-adjacency digitization is the $\mathcal{D}_{\mathcal{B}_{[\cdot]_k}(\frac{1}{2})}$ morphological digitization.

There are two fundamental properties that can almost immediately be deduced from this definition:

Proposition 1 (Adjacency Norm Digitization Properties)

- $\mathcal{D}_E(S \cup T) = \mathcal{D}_E(S) \cup \mathcal{D}_E(T)$ *[2,16,17,18];*
- *if S is a (n-1)-dimensional object, then $\mathcal{D}_{\mathcal{B}_{[\cdot]_k}(\frac{1}{2})}(S)$ is locally k-tunnel free [2,12,13,15] (see Figure 3).*

The first of these properties is very interesting when it comes to modeling. The second means that we have some type of control on the topology of a digitized surface (see Figure 3). The problem is that the Minkowsky sum (or the intersection of a surface and a adjacency ball) of a continuous object with an adjacency ball can be difficult to compute (see [15] for the complicated equations describing the adjacency ball digitization of 3D spheres). In order to simplify the analytical characterization, and subsequently the generation algorithm, of a digital 3D surface, we introduced the notion of *adjacency flakes* in [15] and used those to propose a digitization method for digital implicit surfaces [12]. An *adjacency flake* can be described as the union of a finite number of straight segments joining the opposite vertices of a k-adjacency ball (see Figure 2).

Definition 4 (Adjacency Flake [12,15])
Let $0 \leq k < 3$. A 3D k-adjacency flake, F_k is defined by:

$$F_k = \left\{ \lambda u : \lambda \in \left[0, \frac{1}{2}\right], u \in \{-1,0,1\}^3, \sum_{i=1}^{3} |u_i| = 3-k \right\}.$$

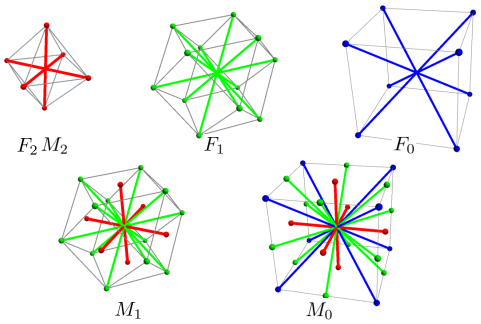

Fig. 2 F_k Adjacency and M_k Matryoshka flakes in 3D.

Instead of an adjacency digitization, with an adjacency ball as structuring element, we are considering a digitization with the corresponding adjacency flake for implicitly defined surfaces. Let $\mathcal{S}$ be an implicit surface $\mathcal{S} = \left\{x \in \mathbb{R}^3 : f(x) = 0\right\}$ which separates space into one (or several) region(s) where $f(x) < 0$ and one (or several) region(s) where $f(x) > 0$:

Definition 5 (Flake Digitization [12,13,15]) The k-Flake digitization of a 3D surface S is defined by $\mathcal{D}_{F_k}(S) = \left\{v \in \mathbb{Z}^3 : (v \oplus F_k) \cap S \neq \varnothing\right\}$.

When the surface S verifies a regularity condition, meaning S is r-regular [19] with $r > \left(\sqrt{3-k} + \sqrt{3}\right)/2$ [12], then the k-flake digitization can be analytically characterized by considering only the vertices of the k-flake. Let us introduce such a digitization scheme:

Definition 6 (Simple Analytical Flake Digitization [12])

$$\mathcal{A}_k(S) = \left\{ v \in \mathbb{Z}^3 : \begin{array}{c} min\{f(x) : x \in (v \oplus F_k)\} \leq 0 \\ and\ max\{f(x) : x \in (v \oplus F_k)\} \geq 0 \end{array} \right\}.$$

When the surface verifies the regularity conditions previously mentioned then $\mathcal{A}_k(S) = \mathcal{D}_{F_k}(S)$ and thus verifies all the properties of morphological digitizations and adjacency flake digitizations. Otherwise there are some differences that may in some cases create topological problems [12] (see section 4.5 and Figure 11). This is largely compensated by the fact that $\mathcal{A}_k(S)$ is easy to construct while $\mathcal{D}_{F_k}(S)$ may not. Figure 3 illustrates the differences between the three digitizations $\mathcal{A}_2, \mathcal{A}_1$ and $\mathcal{A}_0$.

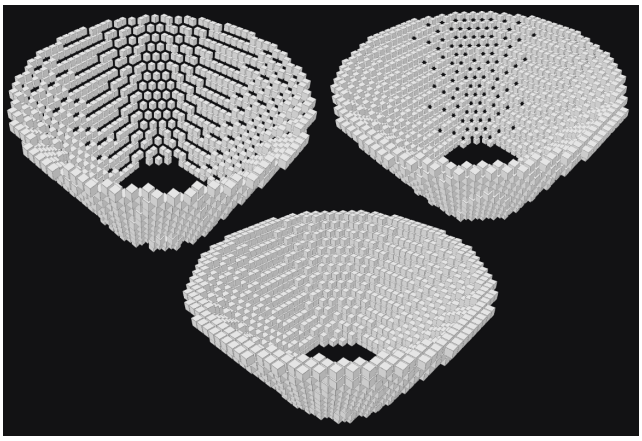

Fig. 3 Illustration of the topological properties with $\mathcal{A}_2, \mathcal{A}_1, \mathcal{A}_0$ digitizations of a simple surface of revolution.

Let us mention one natural-seeming property that is missing with the way we have defined the adjacency flakes: a digitization inclusion property. For the adjacency balls, we have $\mathcal{B}_{[\cdot]_2}(\frac{1}{2}) \subset \mathcal{B}_{[\cdot]_1}(\frac{1}{2}) \subset \mathcal{B}_{[\cdot]_0}(\frac{1}{2})$ which means that for a given Euclidean object S, $D_{\mathcal{B}_{[\cdot]_2}(\frac{1}{2})}(S) \subseteq D_{\mathcal{B}_{[\cdot]_1}(\frac{1}{2})}(S) \subseteq D_{\mathcal{B}_{[\cdot]_0}(\frac{1}{2})}(S)$. This is not the case for adjacency balls as we have defined them since $F_2 \not\subseteq F_1 \not\subseteq F_0$. One way around this is simply to define a sequence of flakes that have the inclusion property:

Definition 7 (3D Matryoshka Flakes) Three dimensional Matryoshka flakes M_k are defined by: $M_2 = F_2$, $M_1 = M_2 \cup F_1$ and $M_0 = M_1 \cup F_0$ (Figure 2).

The implicit surfaces digitizated with the M_1 and M_0 Matryoshka flakes still have the same topological properties than the one digitized with the adjacency flakes but now we have $D_{M_2}(S) \subseteq D_{M_1}(S) \subseteq D_{M_0}(S)$. This can be used to solve some problems with non regular surfaces (see Figure 11).

2.3 Digital surface of revolution with explicit generatrix [2]

Classically, a surface of revolution is defined by the rotation of a curve (the generatrix) around a straight line (the axis). The axis is most often simply the z-axis of a classical xyz-axis system. In our previous work [2], we considered the generatrix as an explicit function $y = g(z)$, defined for all $z \in \mathbb{R}$. The horizontal slice at height z of the surface of revolution is classically a circle (the curve of revolution) of radius $g(z)$. The generatrix acts as an homothetic function. It is easy to see that this definition corresponds to an implicit 3D surface where the curve of revolution is not limited to a circle:

Definition 8 (Implicitely defined Euclidean Surface of Revolution [1,2])

$$S(g,r) = \left\{ (x,y,z) \in \mathbb{R}^3,\ r\left(\frac{x}{g(z)}, \frac{y}{g(z)}\right) = 0 \right\},$$

where $r(x,y) = 0, (x,y) \in [-1,1]^2 \subset \mathbb{R}^2$, is an implicitly defined curve of revolution that separates the window $[-1,1]^2$ into regions of opposite signs, and $\forall z \in \mathbb{R},\ g(z) > 0$.

The implicit surface of revolution can be digitized into a *Digital Surface of Revolution with an infinite explicit generatrix* using the $\mathcal{A}_k$-digitization. This corresponds to the DGCI 2016 paper [2].

3 Digital surface of revolution generation algorithm

In this section we are going to explore our proposed method of digital surface of revolution generation with a hand-drawn generatrix. The particularity of a hand-drawn generatrix is that we cannot assume anything particular about the generatrix: it may correspond to an infinite explicit function (defined on the whole height interval of definition of the surface. Section 2.3), to a finite explicit curve (section 3.2.1), to an horizontal curve (section 3.3), or to the union of some or all of those cases (section 3.4).

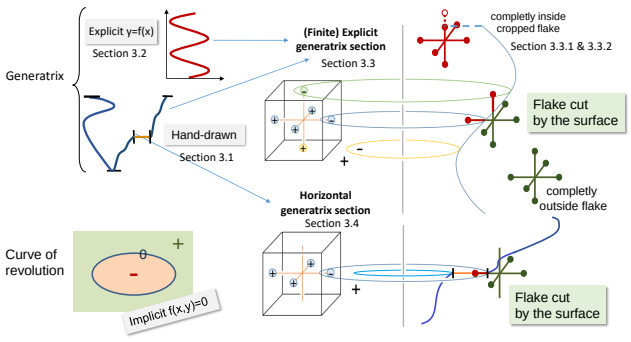

Fig. 4 Illustration of the flake digitization for different type of generatrices.

The input for the algorithm is a curve of revolution based on an implicit 2D function defined in a window $[-1,1]^2$ and an ordered sequence of Euclidean points (z_i, y_i), $1 \leq i \leq n$, typically following the order in which the generatrix has been drawn and representing a sampled curve $g(z)$. In this case there can be several values $g(z)$ for a given z. The function is not necessarily explicit anymore. In order to generate the digital surface of revolution, we will define a finite window with, in particular, a generation domain $[z_{min}, z_{max}], (z_{min}, z_{max}) \in \mathbb{Z}^2$, along the z-axis. For what follows we will consider only one (connected) hand-drawn curve but it is easy to expand the method to work with several different generatrix curves as, by essence of morphological digitizations each surface of revolution corresponding to each generatrix curve can be digitized independently. The different steps of the digital surface of revolution generation algorithm are presented in the algorithm 1. Let us detail the different aspects of this algorithm:

Algorithm 1: Surface of revolution digitization

input : A List **Gen** of 2D Euclidean points as generatrix, an implicit 2D function *Rev* as curve of revolution, a connexity *conex*, and a digitization window W $(x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$.
output: A list $\mathbf{V}$ of voxel belonging to the digitization of the surface of revolution.

```
begin
    Initialize the list V as empty;
    GenParts=DivideAndRecompute(Gen) (* see section 3.1*);
    foreach digital point P(x,y, z)∈ W do
        foreach Part in GenParts do
            if IsHorizontal(Part) then
                if inHorizontal(Rev,conex,x,y,z,Part) then
                    Add the voxel to V;
            else
                if inMonotonic(Rev,conex,x,y,z,Part) then
                    Add the voxel to V;
    return V;
```

3.1 First step of the Algorithm: divide and recompute each generatrix section

The generatrix acts as an homothetic factor to the curve of revolution. For a generatrix defined as a sequence of 2D points representing a sampled curve $g(z)$, there can be more than one such value for a given z. In order to handle this, the generatrix is divided into a set of *strictly monotonic* (in z, increasing or decreasing) or *horizontal* sequences. The end point of one sequence is duplicated as the starting point of the next sequence. The considered morphological digitization methods mean we can simply digitize each sequence separately.

Now, let us remark that we need to compute a value $g(z)$ at the vertices of flakes that have only integer and half-integer coordinates (with the exception of cropped flakes that will be introduced in the following section 3.2.2). Each strictly monotonic sequence (not the horizontal sequences) is replaced by a sequence with interpolated y values for integer and half-integer z values. One can use any kind of interpolation function to compute a y value for each integer and half-integer z value. For all of the following figures and for the online program, we simply used a linear interpolation scheme between appropriate points of the initial sequence. For each strictly monotonic sequence, we need to keep the starting and ending points that define the domain of definition of the finite explicit generatrix (see subsection 3.2.1). For the horizontal sequences, the points with minimum and maximum y are the only two points we need (see subsection 3.3).

3.1.1 Example

Let us give a simple example of this recomputation process:

- Let us start with the generatrix defined by the following sequence of (z_i, y_i) points: $\{(16.9, 12.15), (17.8, 13), (18.1, 14.05), (18.1, 16.95), (18.1, 18.95), (17.2, 20.85), (16.4, 22)\}$;
- This is decomposed into three sequences $\{(16.9, 12.15), (17.8, 13), (18.1, 14.05)\}$, $\{(18.1, 14.05), (18.1, 16.95), (18.1, 18.95)\}$, and $\{(16.4, 22), (17.2, 20.85), (18.1, 18.95)\}$. The first and the last of those sequences are strictly monotonic: one with increasing z values and one with decreasing z values which is reversed to allow a uniform treatment for all monotonic sequences. The second sequence corresponds to an horizontal sequence (in z).
- The first strictly monotonic sequence is interpolated and replaced by: $\{(16.9, 12.15), (17., 12.24), (17.5, 12.72), (18., 13.7), (18.1, 14.05)\}$;
- The horizontal sequence is replaced by: $\{(18.1, 14.05), (18.1, 18.95)\}$. We do not need the other values;
- The last strictly monotonic sequence is interpolated and replaced by: $\{(16.4, 22), (16.5, 21.86), (17., 21.14), (17.5, 20.22), (18., 19.16), (18.1, 18.95)\}$.

3.1.2 Direct access to a monotonic sequence value

The interesting thing about recomputing a strictly monotonic sequence in such a way is that given the z value, one can immediately determine the place in the sequence in a $O(1)$ constant time look up: let us say that we have a monotonic sequence $\{(z_1, y_1), (z_2, y_2), \ldots, (z_n, y_n)\}$: the index in the list to return is:

- If $z < z_2$ then the index to return is 1 ;
- if $z > z_{n-1}$ then the index to return is n ;
- otherwise the index to return is $2z - \lceil 2z_1 \rceil + 2$.

Where $\lceil x \rceil$ is the ceiling of x (i.e. the smallest integer greater than x). For example, in the last monotonic sequence in the previous example, for $z = 17.5$, we have $2 * 17.5 - \lceil 2 * 16.4 \rceil + 2 = 4$. This allows a direct access to the value we are looking for. The formula should be modified according to the starting index.

3.2 Handling strictly monotonic generatrix sequences

A first simple case comes when the strictly monotonic sequence is defined on the whole z generation domain: Let us assume now that we have a strictly monotonic sequence of Euclidean points $\{(z_1, y_1), \ldots, (z_n, y_n)\}$ with $z_1 < \ldots < z_n$ and a generation domain interval $[z_{min}, z_{max}]$ along the z-axis. If $z_1 < z_{min}$ and $z_n > z_{max}$, we are in the case where the generatrix defined by the sequence of points can be handled as an infinite explicit generatrix just as we did in [2] with the appropriate interpolation and resampling as presented in subsection 3.1. It gets more complicated when this is not the case.

Algorithm 2: inMonotonic(see section 3.2.1)

input : An implicit function *Rev*, a connexity *conex*, a voxel center $\mathbf{V_x}$,$\mathbf{V_y}$, $\mathbf{V_z}$ and a Euclidean point list *Part* which is a strictly monotonic increasing part of the generatrix .
output: A boolean that is True if the voxel is in the digitization of this part of the surface ($Rev \times Part$) and false otherwise.

```
begin
    Compute the flake vertices of the voxel for the chosen
    connexity (* see section 3.2.3 and 3.2.2 *);
    Compute the values Lvalues by applying the
    revolution function on these flake vertices using the
    homothetic factor found in Part;
    if min(Lvalues)<0 and max (Lvalues>0) then
        return(TRUE);
    else
        return(FALSE);
```

3.2.1 Explicit finite generatrix

Let us now consider a strictly monotonic sequence of Euclidean points $\{(z_1, y_1), \ldots, (z_n, y_n)\}$ with $z_1 < \ldots <$

z_n, and a generation domain interval $[z_{min}, z_{max}]$ along the z-axis such that $z_1 > z_{min}$ or $z_n < z_{max}$. This case needs to be handled specifically because a digitization of the surface of revolution by a digitization scheme $\mathcal{A}_k$ supposes that we are able, for all points in $\frac{1}{2}\mathbb{Z}^3$, to compute a value for $r\left(\frac{x}{g(z)}, \frac{y}{g(z)}\right)$ where r is the implicitly defined curve of revolution and g is the interpolated generatrix function. In the finite case, when testing a voxel, it is possible that one or several vertices of the k-flake fall outside the definition domain of the generatrix: $g(z)$ is not defined for such a vertex. While it seems logical that the voxel is not part of the digital surface if g is not defined for any of the vertices of a flake, when some vertices of the flake are inside the definition domain of the generatrix and some not then the voxel may be simply partially cut (see 2D illustration in Figure 5.2-8) and not simply crossed by the continuous surface of revolution (see 2D illustration in Figure 5.1). Simply ignoring the vertices for which we have no value can lead to many wrongly discarded voxels and lead to holes at the junction of consecutive generatrix sequences.

To get these values, we can choose substitute points, to better adjust to the domain of the function. For this we introduce *cropped flakes* and *cropped adjacency balls*. These points are actually easy to choose since the domain is restricted along only one coordinate: the z-axis.

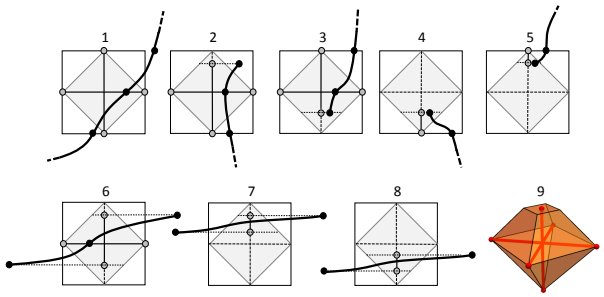

Fig. 5 The different configurations that can occur in the digitization process when cropping flakes.

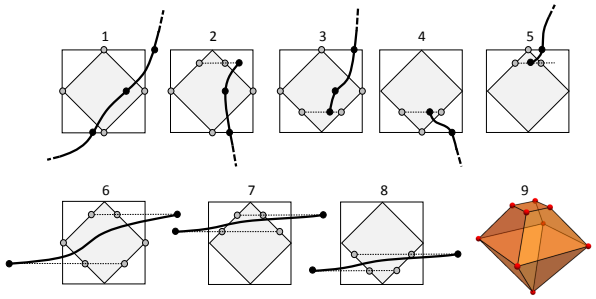

Fig. 6 The different configurations that can occur in the digitization process when cropping balls.

3.2.2 Cropped flakes

There are two ways of considering alternate flake vertices. The first method consists simply in considering only the parts of the flake that are inside the domain. The considered vertices for the computations are the endpoints of the *cropped flake* line segments. For the sake of clarity, Figure 5 shows all the different configurations in 2D for an $\mathcal{A}_2$ digitization (it is easy to extend to $\mathcal{A}_1$ and $\mathcal{A}_0$). An example, Figure 5.9 shows the configuration of Figure 5.2 in 3D.

3.2.3 Cropped adjacency ball

The second method consists in considering the vertices of the *cropped adjacency ball* which the k-flake approximates. There are a few more points for the $\mathcal{A}_1$ and $\mathcal{A}_2$ digitizations, but the results are much closer to what we would get by computing the intersection between the surface and the adjacency balls.

Again, for the sake of clarity, Figure 6 shows, in 2D, which points should be computed in every possible configuration for the $\mathcal{A}_2$ digitization. Figure 6.9 shows the corresponding 3D configuration to Figure 6.2.

3.3 Handling a horizontal generatrix

Let us now consider an horizontal sequence $\{(z_0, y_1), \ldots, (z_0, y_n)\}$. As shown in the example of section 3.1.1, we only keep the minimum and maximum of the y values: $y_{min} = \min(y_1, \ldots, y_n)$, $y_{max} = \max(y_1, \ldots, y_n)$. An horizontal generatrix would be defined by $z = z_0$, $z_0 \in \mathbb{R}$ on an interval $y \in [y_{min}, y_{max}] \subset \mathbb{R}$. Such a generatrix, associated to a circle as the curve of revolution, would result in an annulus as the generated surface. To determine if a voxel is part of this annulus, we simply need to know if it is inside the large circle ($radius = y_{max}$) and outside the small one ($radius = y_{min}$). This works of course exactly in the same way for any implicitly defined curve of revolution. We can test this simply using the $\mathcal{A}_k$ digitization. The main difficulty is the same as in 3.2.1: it will most likely be impossible to compute the value corresponding to any vertices, since only one value is possible for z. Using the same solution as before we will get a slice of adjacency ball (or a slice of flake) defining the substitute points we should use.

Algorithm 3: inHorizontal (see section 3.3)

input : An implicit function Rev, a connexity $conex$, a voxel center $\mathbf{V_x}$,$\mathbf{V_y}$, $\mathbf{V_z}$ and a Euclidean point list $Part$ which is an horizontal part of the generatrix .
output: A boolean that is True if the voxel is in the digitization of this part of the surface ($Rev \times Part$) and false otherwise.

```
begin
    if The horizontal Part crosses the voxel
    (|Part_z − V_z| ≤ 0.5) then
        Compute the flake vertices of the voxel for the
        chosen connexity (* see section 3.2.3 and 3.2.2 *);
        Compute the values Lvalues by applying the
        revolution function on these flake vertices using
        the homothetic factor found in Part;
        if min(Lvalues)<0 and max (Lvalues>0) then
            return(TRUE);
        else
            return(FALSE);
    else
        return (FALSE);
```

3.4 Additional considerations

We have now all we need to generate the surface (see algorithm 1). Since the basis of the generation is a morphological type digitization scheme, Proposition 1 allows us to generate each single sequence independently as described in the previous sections. The most immediate approach is simply to test if each voxel, in a 3D finite generation window, belongs to the surface of revolution of a generatrix point sequence. One can also determine a starting point as seed and generate the surface by neighborhood propagation. This is the main generation method proposed in our online tool. This is usually faster but there is no guarantee that the surface of revolution is formed by only one connected component or that it is not composed of all the voxels of the generation window. In terms of complexity, we test each voxel of the 3D window where we generate the digital surface of revolution, and this for each of the sequences in the set of generatrix point sequences. For a generation window of size n^3 with a set of m generatrix point sequences, the worst case complexity would be $O(m.n^3)$ with no absolute limit on m (one can think of a generatrix based on a function such as $z = sin(1/y)$). There could be some improvement implemented here in future works. There are some obvious optimizations that help during the generation: if a point (x, y, z) is part of the digital revolution surface for one generatrix point sequence, there is no need to test further if it belongs also to the digital surface of another generatrix point sequence. Moreover, each generatrix point sequence is defined on some interval $z_{min} \leq z \leq z_{max}$. We only need to test the voxels (x, y, z) verifying $z_{min} - 0.5 \leq z \leq z_{max} + 0.5$.

4 Results

In this section we present results of our digital generation algorithm with various ways to obtain the point list for the generatrix.

4.1 Hand-drawn generatrix and online creation tool

The first way to create a generatrix as a list of points is to consider a hand-drawn curve defined by a sequence of 2D Euclidean points. This was the main incentive for this work: proposing a natural method to create a digital surface of revolution. We developed an online digital surface creation tool that can be found at the following address: `http://xlim-sic.labo.univ-poitiers.fr/demonstrateurs/DSoR_Generator/?lang=en`. A screenshot of the interface with an example of hand-drawn generatrix can be seen in Figure 10. Some other examples of surfaces generated by the online tool are presented in Figure 14. In Figure 9 we show the work of a local artist, Aurélie Mourier, who used our online tool to create a set of chess pieces. We printed out the *White Queen* piece in two halfs so as to show that it is an actual surface we are creating (see Figure 9). In this case, the surface is based on the $M0$ Matrioshka flake in order to have a *robust* 2-connected tunnel free surface.

4.2 Parametric generatrix

An alternative way of generating a list of points as generatrix is to sample a parametric function. You will find two examples in Figure 7. An adaptative sampling might be useful depending on the curve [20].

4.3 Digital curve as generatrix

A generatrix (or a set of generatrices) can also be extracted from a digital image as a (set of) digital curve(s). This can be done in many different ways. The main problem here is extracting an ordered list of digital points which depends on the type of digital curves present in the image. The main idea for handling such digital curves is to decompose them into a set of digital line segments and then taking the continuous analog of those digital line and regenerate the y values for integer and half-integer z values. Other types of interpolation

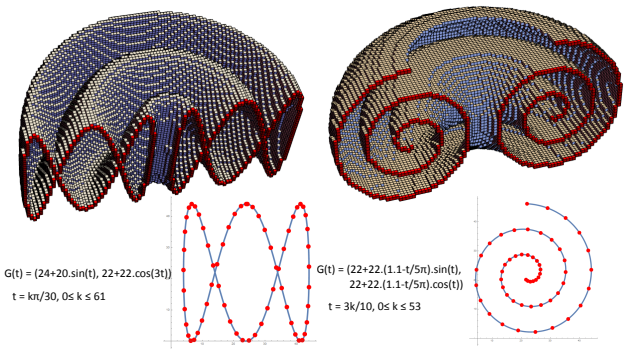

Fig. 7 Two examples of surfaces of Revolution with a parametric curve as generatrix and a circular curve of revolution.

schemes can of course be used as well. The important thing here is not simply to consider the set of digital points as input to the algorithm otherwise the surface may not look smooth. In Figure 8, we decomposed a digital curve into line segments with a regular line segment recognition method [21]. For each digital line of analytical equation $0 \leq ax - by + c < \max(|a|, |b|)$, we considered the continuous line $ax - by - c - |b|/2 = 0$ or $ax - by - c - |a|/2 = 0$ depending on the orientation. A little care has to be taken for the end points of the different digital line segments. If the intersection of two consecutive line segments lies outside of the digital starting/end point of the respective line segment then a little patch function has to be added (see [22]) or one has to use an adapted line recognition where this problem does not occur [23]. This creates a piecewise defined generatrix. Figure 8 shows a digitized sinusoid that we have decomposed into digital line segments. In this example, the curve of revolution is a circle.

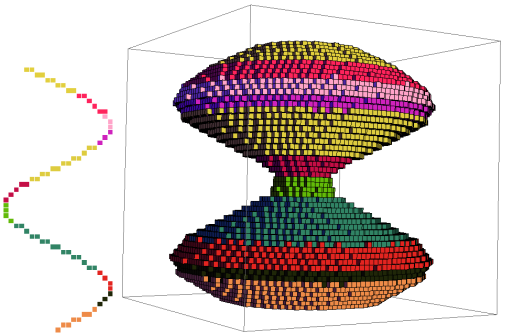

Fig. 8 A digital generatrix decomposed into digital straight segments (left) and the resulting revolution surface (right) using the unit implicit circle as revolution curve.

4.4 Crenelated digital surfaces

As shown in Figure 12.2, the digitization of an implicit surface can lead to a crenelated surface edge. Figure 12.2 has been obtained with the $\mathcal{A}_2$ digitization (with the direct mathematical approach presented in [2]) of the implicitly defined surface shown in Figure 12.1. This proves that this is not a side effect of our present approach, which, of course, may also create crenelated surfaces. The explanation for this crenelation can be seen in Figure 12.3: when, locally, the generatrix is defined on a finite domain, for instance for $z \leq z_0$, for some $z_0 \in \mathbb{R}$, then the corresponding surface of revolution might only cut a disconnected set of adjacency balls on layer $z = \lfloor z_0 + \frac{1}{2} \rfloor$. This leads to a crenelated result.

4.5 Limitations

The main limitation of this digital revolution surface generation method is that the analytical digitization may miss some points. This may happen if the surface is not r-regular [19] with $r > \left(\sqrt{3-k} + \sqrt{3}\right)/2$ [12]. As mentioned in section 2.2, if the continuous surface of revolution is not r-regular, the digitization where we consider only the vertices of the flakes is not equivalent to the one with the whole flakes and the topological properties may be lost. Figure 11.1 illustrates this with a curve that crosses a voxel but the vertices of the flake are all on the same side of the implicit curve of revolution. The voxel is therefore wrongly discarded from the digitization result and disconnections or holes appear. This is classically dealt with interval arithmetics [24] but interval arithmetics works only for digitizations corresponding to F_0-adjacency flakes. One way to reduce the occurrence of such problems is to use Matryoshka flakes as illustrated in Figure 11. In figure 11.3 we have a 0-adjacency flake digitization with a lemniscate as curve of revolution (Figure 11.2). The surface is disconnected. The use of the Matryoshka M_0-flake (figure 11.4) solves the problem. Of course, there will always be cases where the topological properties can not be guaranteed. As a concluding remark, let us note that a flake digitization does not guarantee and optimal k-tunnel free surface even though the regularity conditions are met. There can be simple points with regard to the k-tunnel freeness property. Ensuring topological optimality is still an open question.

5 Conclusion and perspectives

In this paper we proposed a simple algorithm to generate a large class of surfaces of revolution based on an im-

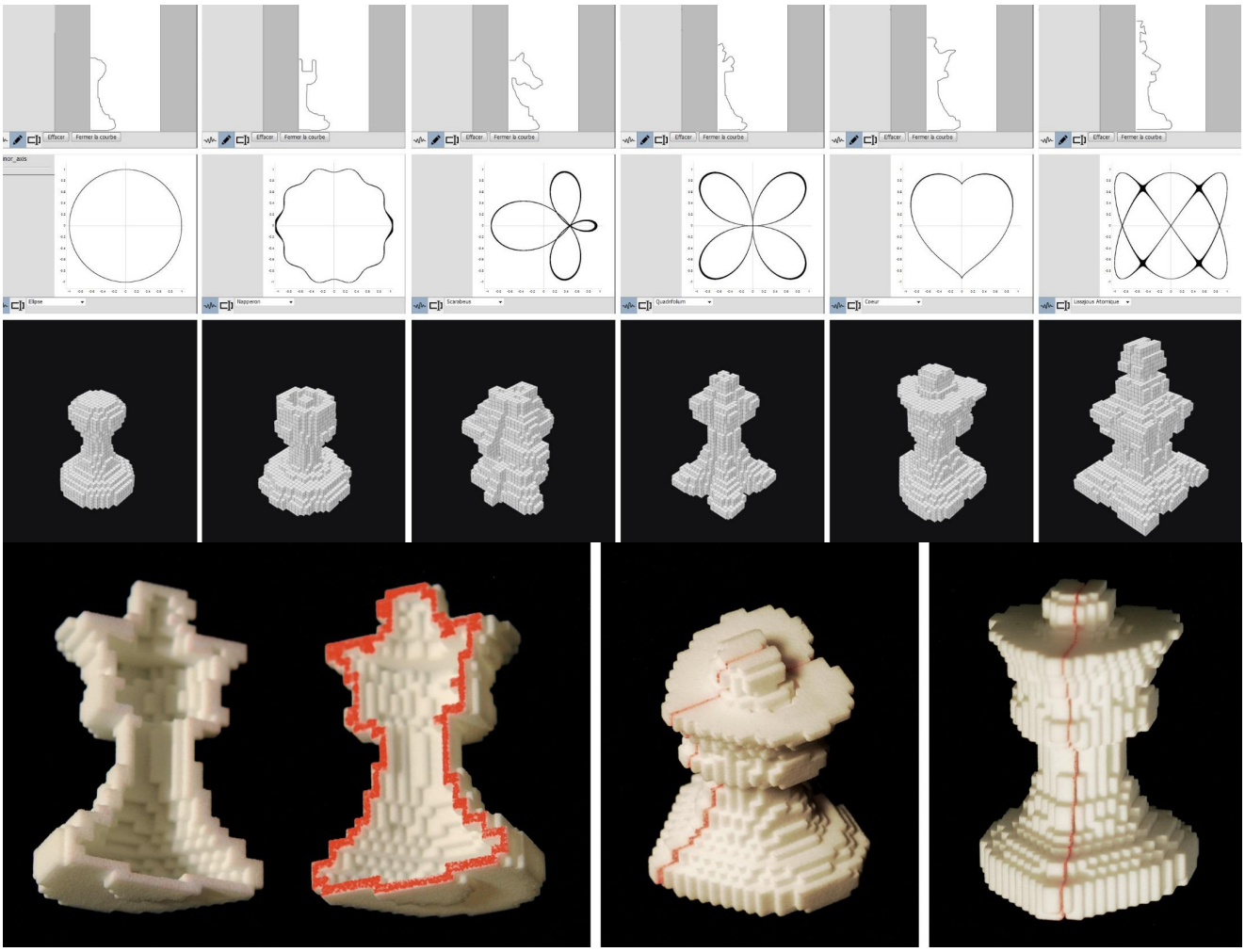

Fig. 9 Set of chess pieces and example of 3D printed *White Queen*. First row: the generatrices; second row: the revolution curves; third row: the resulting surfaces; last row: several views of the 3D printed *White Queen*.

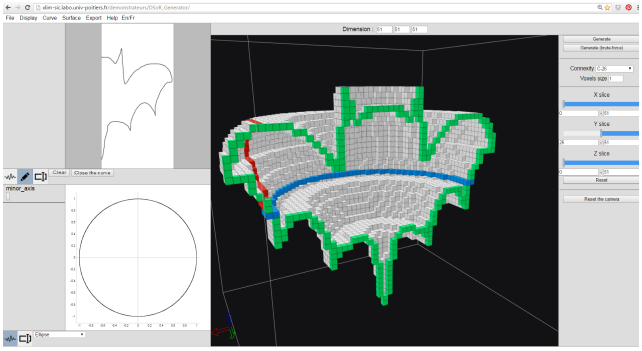

Fig. 10 A hand-drawn generatrix and the obtained surface of revolution using the unit implicit circle as revolution curve. Online tool can be found at : http://xlim-sic.labo.univ-poitiers.fr/demonstrateurs/DSoR_Generator/?lang=en .

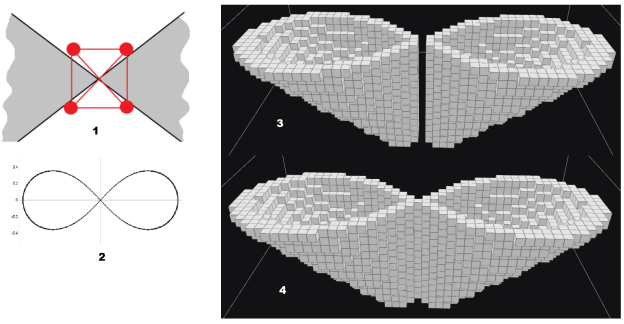

Fig. 11 1. Case where flake misses a point. 2. Lemniscate curve of revolution 3. Disconnected 2-connected surface. 4. 2-connected surface with Matrioschka M_0-flake.

plicit 2D curve as curve of revolution and a hand-drawn generatrix. Any ordered sequence of points representing a 2D curve can be used to define a generatrix. Given some regularity conditions for the surface, we control the topology of the resulting digital surface: the type of tunnels that appear in the surface can be defined, which allows to generate surfaces that are adapted for

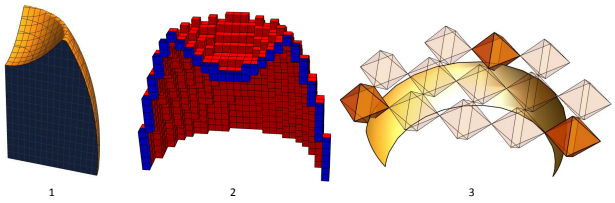

Fig. 12 1. An implicit 3D surface. 2. Its $\mathcal{A}_2$ digitization. 3. Only some adjacency balls are cut by the surface.

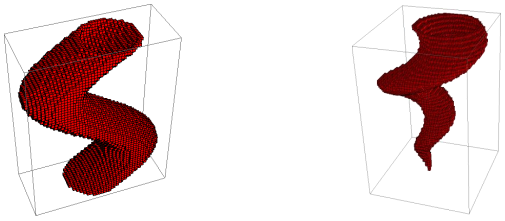

Fig. 13 Two examples of some other surfaces we can build with our algorithm.

various applications (rendering, 3D-printing, etc.). We propose an online tool that illustrates the proposed method and that can be used to create surfaces and export them into various formats including 3D printing formats. The online tool can be found at the following address: `http://xlim-sic.labo.univ-poitiers.fr/demonstrateurs/DSoR_Generator/?lang=en`. We set up an *imgur animated gif repository* for some of our digital surface creations : `http://imgur.com/a/eDFbY`. Let us be noted that the repository holds not only digital surfaces of revolutions but also swept tubes and may contain examples of future extensions. The methods we propose are not limited to digital surfaces of revolution and can be adapted to more general types of surfaces. For example the generatrix can be used as central axis for the revolution curve (see the left of Figure 13):

Definition 9

$$S(g,r) = \left\{x,y,z \in \mathbb{R}^3,\ r\left(x, y - g(z)\right) = 0\right\}.$$

We can also combine an homothetic function $h(z)$ and two translation functions $t(z), u(z)$ for the center of the revolution curve (see the right of figure 13):

Definition 10

$$S(h,t,r) = \left\{x,y,z \in \mathbb{R}^3,\ r\left(\frac{x}{h(z)} - u(z), \frac{y}{h(z)} - t(z)\right) = 0\right\}.$$

This will be handled in a forthcoming work. One of the next steps will be to consider hand-drawn curves of revolution which will allow further control of the types of surfaces that can be generated.

Acknowledgement: This work has been supported by the CPER 2015-2020, NUMERIC Program and FEDER-FSE MODEGA Project of the Nouvelle-Aquitaine Region, France. we would like to thank *Aurélie Mourier*, `http://www.aureliemourier.net/`, a local artist, that worked with us on the online tool and on creating the chess pieces.

References

1. Salomon, D.: Curves and surfaces for computer graphics. Springer (2006)
2. Andres, E., Largeteau-Skapin, G.: Digital Surfaces of Revolution Made Simple. In: Discrete Geometry for Computer Imagery: 19th IAPR International Conference, DGCI 2016, Nantes, France, April 18-20, 2016.Volume 9647 of the series Lecture Notes in Computer Science. Springer International Publishing (2016) 244–255
3. Stolte, N., Kaufman, A.E.: Novel techniques for robust voxelization and visualization of implicit surfaces. Graphical Models **63**(6) (2001) 387–412
4. Yongsheng, L., Stolte, N.: Robust voxelization based ray tracing of implicit surfaces. In: Proc. 6th Sixth IASTED Honolulu, Hawaii (USA). (2003) 177–180
5. Bhowmick, P., Bera, S., Bhattacharya, B.B.: Digital circularity and its applications. In Wiederhold, P., Barneva, R.P., eds.: Combinatorial Image Analysis, 13th International Workshop, IWCIA Playa del Carmen, Mexico, November 24-27, 2009. Proceedings. Volume 5852 of Lecture Notes in Computer Science., Springer (2009) 1–15
6. Kumar, G., Sharma, N.K., Bhowmick, P.: Wheel-throwing in digital space using number-theoretic approach. IJART **4**(2) (2011) 196–215
7. Bresenham, J.: A linear algorithm for incremental digital display of circular arcs. Commun. ACM **20**(2) (1977) 100–106
8. Bera, S., Bhowmick, P., Bhattacharya, B.B.: On the characterization of absentee-voxels in a spherical surface and volume of revolution in $\mathbb{Z}^3$. Journal of Mathematical Imaging and Vision (2016) 1–19
9. Andres, E.: Discrete circles, rings and spheres. Computer and Graphics **18**(5) (1994) 695–706
10. Andres, E., Jacob, M.A.: The discrete analytical hyperspheres. IEEE Trans. on Vis. and Comp. Graphics **3**(1) (1997) 75–86
11. Andres, E., Roussillon, T.: Analytical description of digital circles. In: 16th DGCI, Nancy (France). Volume 6607 of LNCS., Springer (2011) 235–246
12. Toutant, J., Andres, E., Largeteau-Skapin, G., Zrour, R.: Implicit digital surfaces in arbitrary dimensions. In: 18th DGCI, Siena (Italy). Volume 8668 of LNCS., Springer (2014) 332–343
13. Andres, E.: Digital Analytical Geometry: How Do I Define a Digital Analytical Object? In: Combinatorial Image Analysis: 17th International Workshop, IWCIA 2015, Kolkata, India, November 24-27, 2015. Volume 9448 of the series Lecture Notes in Computer Science. Springer International Publishing (2015) 3–17
14. Laine, S.: A topological approach to voxelization. Computer Graphics Forum **32**(4) (2013) 77–86
15. Toutant, J., Andres, E., Roussillon, T.: Digital circles, spheres and hyperspheres: From morphological models to analytical characterizations and topological properties. Discrete Applied Mathematics **161**(16-17) (2013) 2662–2677

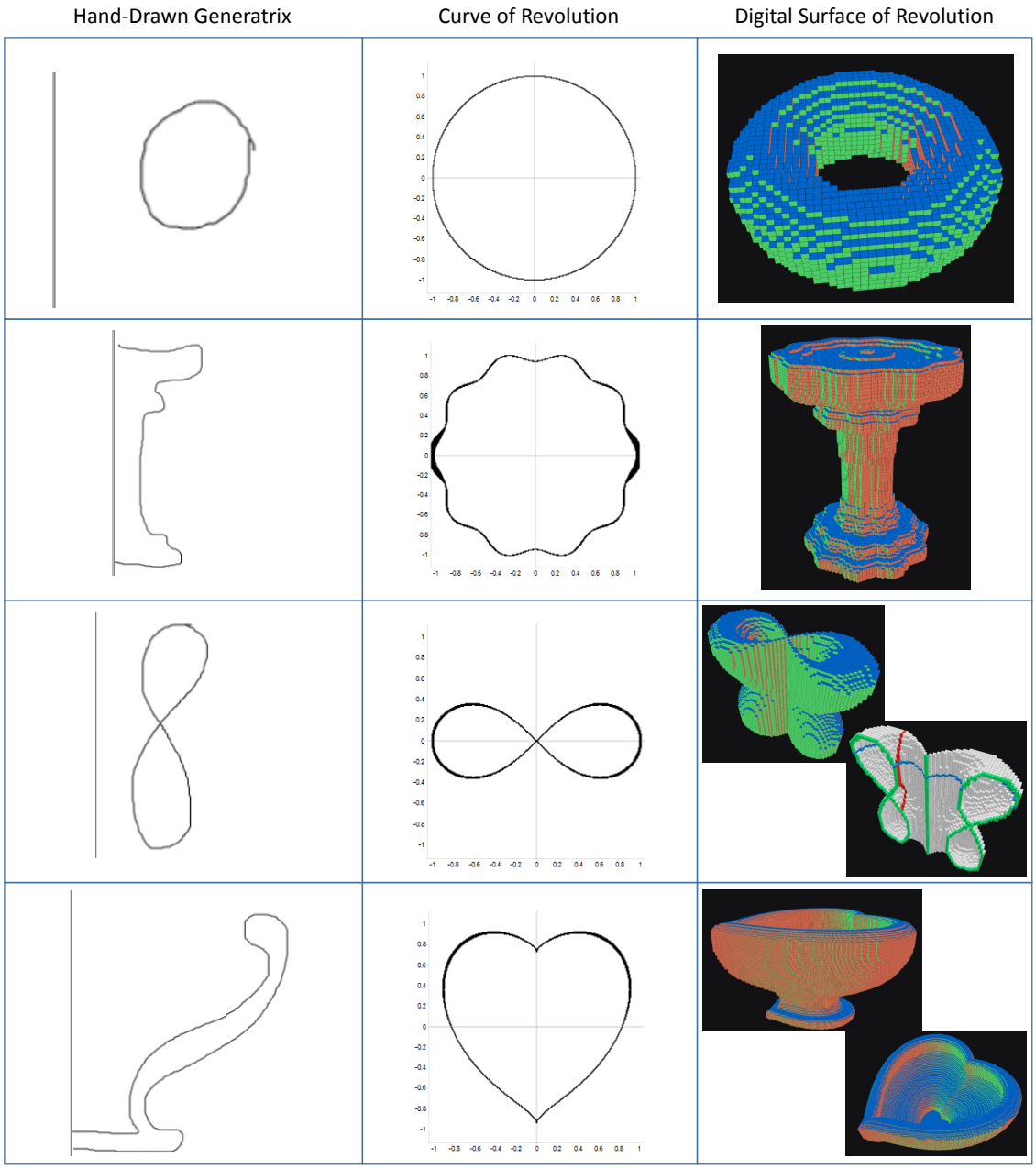

Fig. 14 Examples of hand-drawn generatrix, implicit curve of revolution and the corresponding digital surface of revolution.

16. Heijmans, H.: Morphological discretization. In: Geometrical Problem in Image Processing, U. Eckhardt and al. Eds, Akademie Verlag, Berlin. (1991) 99–106
17. Heijmans, H., Toet, A.: Morphological sampling. Graphics and Image Processing: Image Understanding **54**(3) (1991) 384–400
18. Andres, E.: The supercover of an m-flat is a discrete analytical object. Theor. Comput. Sci. **406**(1-2) (2008) 8–14
19. Stelldinger, P., Köthe, U.: Towards a general sampling theory for shape preservation. Image and Vision Computing **23**(2) (2005) 237–248
20. de Figueiredo, L.H.: Adaptive sampling of parametric curves. Graphics Gems V **5** (1995) 173–178
21. Debled-Rennesson, I., Reveillès, J.: A linear algorithm for segmentation of digital curves. IJPRAI **9**(4) (1995) 635–662
22. Breton, R., Sivignon, I., Dupont, F., Andres, E.: Towards an invertible euclidean reconstruction of a discrete object. In: 11th DGCI Naples (Italy). Volume 2886 of LNCS., Springer (2003) 246–256
23. Sivignon, I., Breton, R., Dupont, F., Andres, E.: Discrete analytical curve reconstruction without patches. Image and Vision Computing **23**(2) (2005) 191–202
24. Duff, T.: Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry. In: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992. (1992) 131–138