Reversible vectorisation of 3D digital planar curves and applications

Isabelle Sivignon [a,*], Florent Dupont [a], Jean-Marc Chassery [b]

[a] *Laboratoire LIRIS - Université Claude Bernard Lyon 1*
UMR 5205 CNRS
Bâtiment Nautibus - 8 boulevard Niels Bohr
69622 Villeurbanne Cedex, France

[b] *Laboratoire LIS - Grenoble*
UMR 5083 CNRS
961, rue de la Houille Blanche
38402 St Martin D'Hères, France

Abstract

This paper tackles the problem of the computation of a planar polygonal curve from a digital planar curve, such that the digital data can be exactly retrieved from the polygonal curve. The proposed transformation also provides an analytical modelling of a digital plane segment as a discrete polygon composed of a face, edges and vertices. A dual space representation of lines and planes is used to ensure that the computed curve remains inside the digital curve, and this tool enables to define a very efficient algorithm. Applied on the digital plane segments resulting from the decomposition of a digital surface, this algorithm provides a set of polygons modelling exactly the digital surface.

Key words: discrete polygon, 3D, vectorisation, modelling.

1 Introduction

Digital objects are defined as sets of grid points in $\mathbb{Z}^n$. Those objects carry redundant geometrical information due to their discrete structure: an object in represented as a set of elementary cells (called pixels in 2D, voxels in 3D).

* Corresponding Author. E-mail: isabelle.sivignon@liris.cnrs.fr - Tel: +33 4724432662 - Fax: +33 472431536

Indeed, consider for instance the representation of a cube of size n. On one hand, in the continuous space, this object is totally defined by six square faces, whatever the size of the cube. On the other hand, about $6n^2$ voxels are needed to represent the surface of this object in the discrete space: each point has to be stored. We see that exploiting geometrical properties of digital surfaces would lead to important advances in digital objects modelling, representation, compression for instance.

The first natural geometrical properties to study are linearity and coplanarity. Indeed, if we draw a parallel with the classical continuous space, one of the most widely used model for 3D objects representation is the polyhedral model, where an object's surface is defined as a set of polygons, edges and vertices.

The definition of digital linear structures like digital lines [1] and digital planes [2,3] originated a lot of works dealing with the decomposition of the contour of a digital object into digital linear primitives [4–6] or according to other geometrical properties [7]. Such a decomposition actually apprehends global geometrical properties of those objects. The result of a decomposition algorithm is usually a labelling of the surface voxels according to the digital plane segment (DPS for short) they belong to. These DPS define digital faces as in the continuous case. Nevertheless, a complete analytical modelling of a surface requires an analytical definition of the boundary of those DPS as a set of digital edges and vertices.

In the 2D case, this problem is similar to the classical vectorisation issue with an additional constraint: the digital curve must be exactly retrieved from the computed polygonal one. This constraint is actually necessary to ensure that the algorithm computes an analytical model of the digital curve. An algorithm using digital geometry has been proposed to solve this problem in [8], and similar tools are used in the algorithm we propose for the 3D case.

In this paper, we deal with the 3D case and the problematics is stated as follows: given a digital planar curve C, compute a polygonal curve which provides an analytical representation of C. If the curve C is the boundary of a digital plane segment and since the voxels of C are coplanar, the DPS is analytically defined by the digital plane containing C and the boundary by the computed polygonal curve.

Parts of this work have been presented in [9]. Main differences include some definitions, proofs, technical details, complexity issues, and application examples. It is composed of six sections. In Section 2, we present the general framework of our algorithm, defining the notions of digital lines and planes and their dual representation. The third section deals with the precise description of our algorithm. Complexity issues, both theoretical and experimental, are presented in Section 4. Modelling results provided by our algorithm are presented in Section 5, which also presents image results about the application of this algorithm on 3D objects digital surfaces, and first thoughts on using this approach for lossless compression.

2 Digital lines, planes and their dual representation

2.1 Digital curve, line and plane definitions

First of all, we define the framework used in this paper. The objects we are dealing with are sets of cells of the three-dimensional regular grid $\mathbb{Z}^3$. Those grid cells are called voxels, by analogy with the 2D term pixel. These objects will be called digital or discrete objects in the rest of the paper. We first recall classical definitions.

Definition 1 (Connectivity) *Two discrete cells (pixels in 2D, voxels in 3D for instance) of $\mathbb{Z}^n$ are k-connected if they share a cell of dimension k ($k < n$). For instance, in 3D, two voxels are* 0*-connected if they share a vertex,* 1*-connected if they share an edge, and* 2*-connected if they share a face.*

Definition 2 (k-curve) *Given a k-connectivity relationship, a set of discrete cells $\{p_i\}_{i=0\ldots n}$ is a k-curve if and only if for all i, p_i has exactly two k-neighbours.*

In the following, we deal with 2-curves in 3D, that we will call digital curves. Instead of considering the discrete cells we will sometimes consider the discrete points (grid points), centre of the cells.

As in the continuous space, straight lines and planes are well defined in the digital space. We provide here the general definition of a discrete hyperplane which covers the 2D (discrete lines) and 3D (discrete planes) cases.

Definition 3 (Discrete hyperplane [10]) *A discrete hyperplane of dimension n and normal vector $N \in \mathbb{R}^n$ is defined as the set of lattice points $X \in \mathbb{Z}^n$ such that:*

$$A \leq \sum_{i=1}^{n} N_i X_i < B$$

with A, $B \in \mathbb{R}$.

The value $B - A$ is called thickness of the hyperplane and usually depends on the normal vector N. The discrete hyperplane connectivity depends on the definition of B and A and for our application, we set $B = \frac{\sum_{i=1}^{n} |N_i|}{2}$ and $A =$

$-B$. Moreover, we assume that there exists j such that $C_j > 0$ and for all $i < j$, $C_i = 0$. Those settings define the so called standard hyperplane. Topologically, standard hyperplanes are the thinnest $(n-1)$-connected hyperplanes without hole: any connected path joining the two background sides of a hyperplane contains at least one point of the hyperplane. An illustration of a standard plane in dimension 3 is given in Figure 1(a).

Geometrically, the standard digitization of a hyperplane is very close to the supercover digitization [11,12] which states that any grid cell crossed by the object belongs to the digitized object. But supercovers may contain "bubbles", *i.e.* many digitization grid cells for one point (points with half-integer coordinates for instance). To cope with that problem, an orientation convention is defined and leads to the standard digitization scheme we use. Our algorithm uses the following well-known property of digital standard hyperplanes, that we set out in the 3D case:

Proposition 4 *Let V be a set of voxels in a digital standard plane. Then there exists a plane p crossing all the voxels of V. p is called a carrier plane of V.*

To end with standard model properties, it has nice geometrical consistency properties which enable to define discrete polygons. This definition was proposed by Andres in [10] and gives an analytical definition of a discrete polygon (a set of linear constraints defining the discrete face, edges and vertices). Our algorithm uses this definition in order to compute a polygonal curve for a given digital curve.

We can now formally define the digital planar curves that are of interest in this work:

Definition 5 (DPC) *A set of voxels $V = \{v_i\}_{i=0\dots n}$ is a digital planar curve if and only if V is a 2-curve and all the v_i belong to the same standard digital plane.*

From Proposition 4, there exists a carrier plane for the voxels of a DPC. Similarly to the continuous case, modelling a DPC is done through the decomposition of the DPC into 3D digital line segments. 3D digital lines are also well-defined in the digital space according to the standard model:

Definition 6 *Consider a 3D straight line of directional vector (a, b, c), and going through the point (x_0, y_0, z_0). Then the standard digitization of this line is the set of integer points fulfilling the conditions given by the following double*

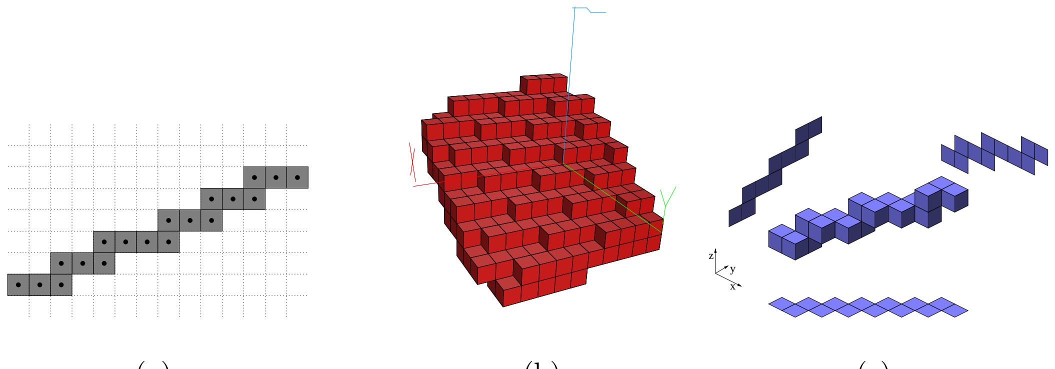


Fig. 1. Example of (a) a 2D discrete standard straight line and (b) a discrete standard plane and (c) a 3D standard digital line with its three projections.

inequalities:

$$\begin{aligned} -\tfrac{|a|+|b|}{2} &\leq bx - ay + ay_0 - bx_0 < \tfrac{|a|+|b|}{2} \\ -\tfrac{|a|+|c|}{2} &\leq cx - az + az_0 - cx_0 < \tfrac{|a|+|c|}{2} \\ -\tfrac{|b|+|c|}{2} &\leq cy - bz + bz_0 - cy_0 < \tfrac{|b|+|c|}{2} \end{aligned}$$

where $(b > 0$ *or* $(b = 0$ *and* $a > 0))$ *and* $(c > 0$ *or* $(c = 0$ *and* $a > 0))$ *and* $(c > 0$ *or* $(c = 0$ *and* $b > 0))$ *(otherwise, the strict and large inequalities of those equations are switched, see [10]).*

From this definition, we derive that a necessary condition for a set of voxels V to be a 3D standard segment is that the three projections of V are 2D standard segments (see Definition 3).

2.2 Parameter spaces and preimages

2.2.1 Parameter spaces

Detection of collinearity is of great importance in this algorithm. Since 3D digital lines definition is based on 2D digital lines definition, our main concerns is about 2D collinearity, and the tools we use are presented in this context.

Many algorithms exist to decide if a set of pixels is a digital segment or not, and three classes of so called recognition algorithms may be defined: chain codes based algorithms (for instance [13,14]), arithmetical algorithms [15] and preimage algorithms [16–18]. Our application needs an online algorithm (pixels may be added one by one), together with the computation of the whole set of lines containing the pixels in their digitization. This last requirement is derived from the fact that the recognition process is constrained in our algorithm (see

Section 3.1). Preimage based algorithms are the only ones fulfilling those two conditions. They use a transformation in a parameter space that we present in the following, and the preimage itself is presented in the next paragraph.

The main idea is that a line in the Cartesian space is represented by a point in the parameter space, and conversely, a point in the Cartesian space corresponds to a line in the parameter space. We say that the point is the dual of the line.

This parameter space is similar to the one defined by the Hough Transform [19] which is classically used in image analysis for shape detection problems. The main difference between the Hough Transform and the transform we use is that the uncertainty induced by the discrete nature of the data is not handled during a quantification step but during the transform itself (see next section on preimages).

An illustration of this mapping is given in Figure 2. Note that in this figure, the dual space is defined according to a normalization along the y direction (lines $x = a$ cannot be represented in this parameter space). Consequently, two parameter spaces can be defined in the 2D space, one for each direction.

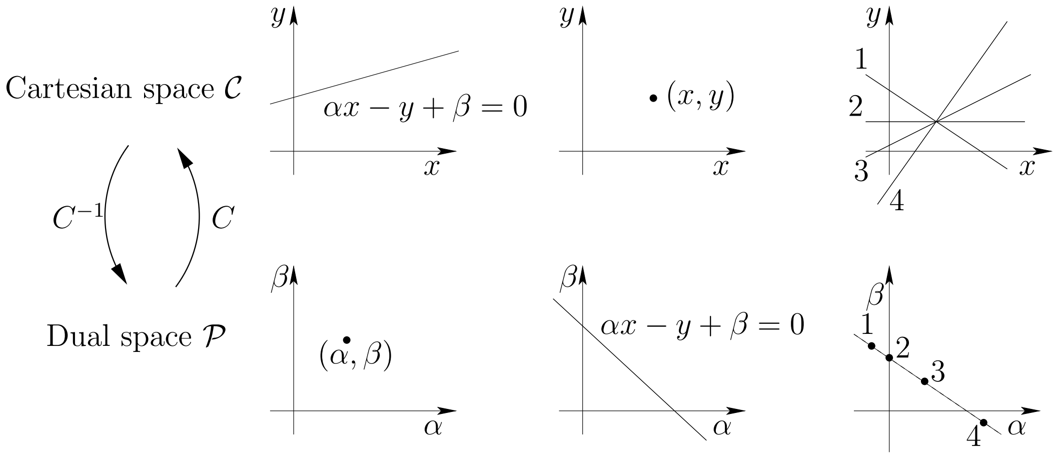


Fig. 2. Representation of the links between the Cartesian (top) and the parameter spaces (bottom) for elementary geometric objects.

Similarly, three parameter spaces $(0\alpha\beta\gamma)$ can be defined in 3D. Using the notations of [8], we denote those parameter spaces by $\mathcal{P}_x$, $\mathcal{P}_y$ and $\mathcal{P}_z$ according to the normalization variable. When no particular parameter space is meant, we use $\mathcal{P}$. One plane in the Cartesian space is represented by one point in each parameter space and conversely. We also consider the intersection between those 3D spaces and the planes $\alpha = 0$ and $\beta = 0$. For instance, $\mathcal{P}_{xz}$ is equal to $\mathcal{P}_x \cap (\alpha = 0)$. Those spaces can be considered either as restrictions of 3D spaces or as two-dimensional parameter spaces. Thus, one point in $\mathcal{P}_{xz}$ can be considered either as a plane perpendicular to $y = 0$ or as a 2D line in $(0xz)$ (see [8] for more details and illustrations). Moreover, a 3D line in the

Cartesian space $\mathcal{C}$ is represented by another 3D line in the parameter space $\mathcal{P}$.

In the following, we denote by C (or C_x, C_y, C_z when a particular parameter space is meant) the operator which transforms one element of the parameter space into its corresponding element in the Cartesian space.

The representation in the parameter space of a 3D line embedded in a plane is a key point of our algorithm. It is actually easy to see that since a 3D line l maps to another 3D line $C^{-1}(l)$ (each point of $C^{-1}(l)$ corresponds to a plane containing l), and since a plane P maps to the point $C^{-1}(P)$, then l is embedded in P if and only if $C^{-1}(l)$ goes through $C^{-1}(P)$ (see Figure 3).

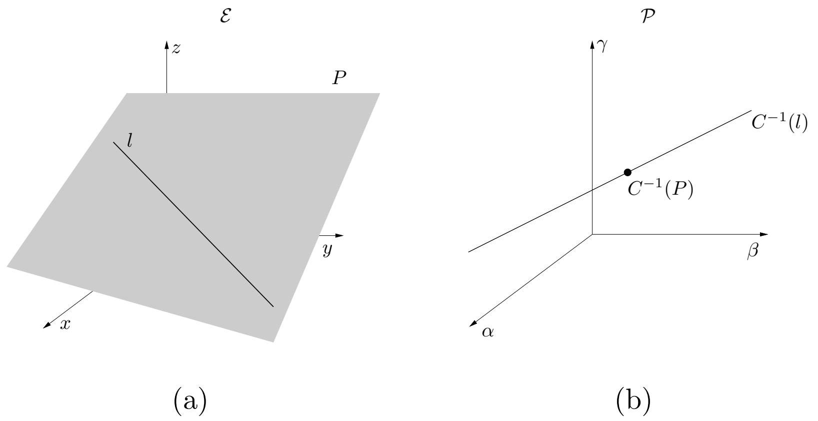


Fig. 3. Representation of a 3D line embedded in a plane in the Cartesian (a) and the parameter (b) spaces.

2.2.2 Preimages

The definition we present for the 2D case of pixel sets can be directly extended in 3D for voxel sets. Consider a set of pixels ϵ and a digitization scheme. We call preimage (or domain) of ϵ the set of Cartesian lines containing ϵ in there digitization. This set is represented as a set of points in the parameter space defined previously.

Let us consider for instance the line l defined by $ax - by + r = 0$ where $a > 0$ and $b > 0$. Then, the standard digitization of l is the set of discrete points (x, y) fulfilling the inequalities $-\frac{a+b}{2} \leq ax - by + r < \frac{a+b}{2}$ (see Definition 3). Therefore, the lines $\alpha x - y + \beta = 0$ containing the point (x_0, y_0) in their digitization fulfill the inequalities $-\frac{\alpha+1}{2} \leq \alpha x_0 - y_0 + \beta < \frac{\alpha+1}{2}$. Thus, a discrete point defines two half-spaces in the parameter space, and the intersection of those half-spaces represents the set of lines containing this discrete point in their digitization. Given a set S of discrete points, the preimage of S is the convex polygon in $\mathcal{P}$ defined by the intersection of the half-spaces related to the discrete points of S (see Figure 4).

3 Reversible vectorisation of a planar 3D discrete curve

In this section, we present an algorithm to compute a polygonal planar curve from a 3D discrete planar curve (DPC) in a reversible way. The outline of this algorithm is briefly presented in Section 3.1 in order to introduce the three key points detailed in the following sections. The overall algorithm is then summarized in Section 3.5.

3.1 Outline of the algorithm

From Proposition 4, there exists a carrier plane p crossing all the voxels of a given DPC. The overall algorithm consists in computing a polygonal line included in p and crossing all the voxels of the DPC.

To compute a polygonal curve from a DPC, we propose the following outline, that we presented in [8] in the case of 2D discrete curves and non coplanar 3D discrete curves. For the sake of clarity and continuity, the notations used in this paper are similar to the ones used in [8].

Consider a DPC described as an ordered set of voxels $\{V_1, V_2, \dots V_n\}$, and a carrier plane p crossing each V_i. A Cartesian point r_1 is chosen inside the first voxel V_1 and the plane p, and the voxels are added one by one (they define a discrete segment s_1) while there exists a Cartesian line going through r_1, through all the voxels of s_1 and embedded in the carrier plane p. In other words, s_1 is incrementally extended while:

(1) s_1 is a 3D discrete segment;
(2) among the lines containing s_1 in their digitization, there exists at least one line that is embedded in p and that goes through the fixed point r_1.

When one of those two conditions does not hold anymore, the first Cartesian segment endpoint r_2 is computed as an intersection point between the computed line and s_1's last voxel. The fixed extremity of the next Cartesian segment is set to r_2 and this process starts over.

This synopsis points out three key points: the recognition of a 3D discrete segment, the control of the planarity of the polygonal curve computed, and the choice of a vertex as starting point of the polygonal curve.

3.2 Recognition of a 3D standard segment

The first step is to define an algorithm to recognize a 3D standard segment, *i.e.* the standard digitization of a 3D line segment. To do so, we use Definition 6 which provides an analytical model for a 3D digital line. We saw that a necessary condition for a set of voxels to be a 3D standard digital segment is that the three projections are 2D standard digital segments.

To ensure this property, and following Section 2.2.2, we compute the preimages of the sets of pixels of the three projections. If the three preimages are not empty, then this condition is fulfilled, otherwise, the set of voxels is not a 3D standard segment. Moreover, we said in Section 3.1 that a Cartesian point is fixed inside one voxel before the recognition process. Thus, we do not consider the whole set of solution lines, but only the lines going through this fixed point. As illustrated in Figure 4(a), the projections of this fixed point p onto the three coordinate planes define three points p_x, p_y and p_z that are represented by three lines in the parameter spaces. Thus, the preimages we work on are no more polygons but simply segments denoted by I_x, I_y and I_z (see Figure 4(b)).

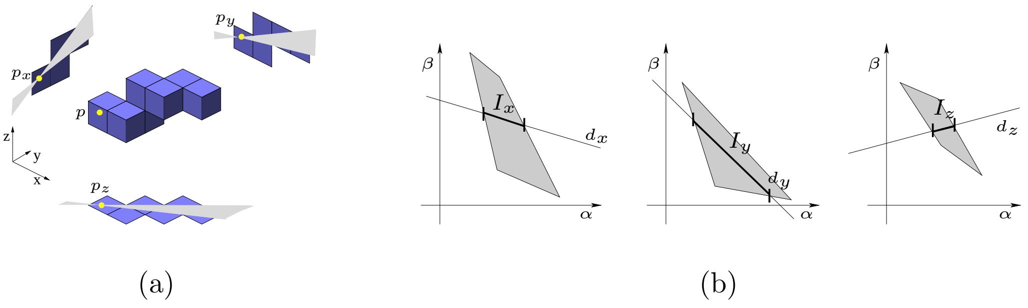


Fig. 4. Recognition of the three projections of a given set of voxels with a fixed point p: when a point p is fixed in the first voxel, the sets of solution lines for the projections are reduced to segments denoted by I_x, I_y and I_z.

Nevertheless, this condition over the three projections is not sufficient to define a 3D standard segment since the projection parameters are not independent: indeed, choosing any three 2D digital lines as projections do not define a 3D digital line. A compatibility condition between the parameters of the three projections needs to be added (see [8]), but we do not go on further details about this particular point since this condition is ensured while considering the embedding of the curve into a plane.

3.3 Ensuring coplanarity

3.3.1 General case

In the following, we consider the general case where the carrier plane p is defined by the equation $ax + by + cz + \mu = 0$, with a, b and c not equal to zero.

It is easy to see that any of the three preimage segments I_x, I_y and I_z can be represented in two out of the three parameter spaces $\mathcal{P}_x$, $\mathcal{P}_y$ and $\mathcal{P}_z$. Indeed, those preimage segments are embedded in the planes $\alpha = 0$ or $\beta = 0$ in those parameter spaces. For instance, consider the parameter space $\mathcal{P}_x$ where the two segments I_z and I_y can be represented. In this space, the carrier plane P is represented by a point $C^{-1}(p)$. Thus, the 3D lines l embedded in p and containing the set of voxels considered in their digitization are those such that $C^{-1}(l)$ crosses $C^{-1}(p)$, I_y and I_z.

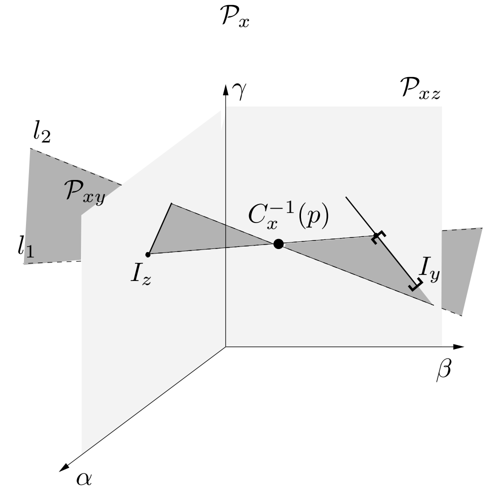


Fig. 5. Reduction of the dual segment I_y according to the dual of the carrier plane p and to I_z. The remaining part of I_y after the reduction is depicted between brackets.

A reduction process of the segments I_z and I_y according to $C^{-1}(p)$ is done, as illustrated in Figure 5. The grey cone drawn on this figure represents all the lines going through one point of I_z and the point $C^{-1}(p)$. Thus, the points of I_y which do not belong to this cone (out of brackets on Figure 5) must be deleted since there does not exist a line going through I_z, $C^{-1}(p)$ and those points. After the reduction of I_y, the reduction of I_z is computed.

Let us detail this reduction process. In the following, we use the generic indices i, j and k to denote any of the space coordinates x, y and z. The basic operation of this reduction process is the computation of the image of an interval point through the $C^{-1}(p)$. Using the notations depicted in Figure 6(a), this function is simply defined as follows:

$$Red((\alpha_j, 0, \gamma_j), (p_\alpha, p_\beta, p_\gamma)) = (0, \beta_k, \gamma_k)$$

where $\beta_k = \frac{\alpha_j p_\beta}{\alpha_j - p_\alpha}$ and $\gamma_k = \frac{\alpha_j}{\alpha_j - p_\alpha}(p_\gamma - \gamma_k) + \gamma_j$. From a geometrical point of view, and in the primal Cartesian space, this equation corresponds to the computation of one the parameters of one projection (the point $(0, \beta_k, \gamma_k)$ in the parameter space) of a 3D line defined by another projection (the point $(\alpha_j, 0, \gamma_j)$ in the parameter space) and a plane (given by $(p_\alpha, p_\beta, p_\gamma)$ in the parameter space).

In order to simplify the computation of these reductions, it is actually possible to do all these operations in the 2D space $(0\alpha\beta)$. This projection is one-to-one and onto since the γ coordinate is defined by the plane $C^{-1}(r)$ (r is the point fixed before the reconstruction), which is by definition not orthogonal to the plane $(0\alpha\beta)$ (see Figure6(b)). Thus, each preimage I is uniquely defined by a one-dimensional interval.

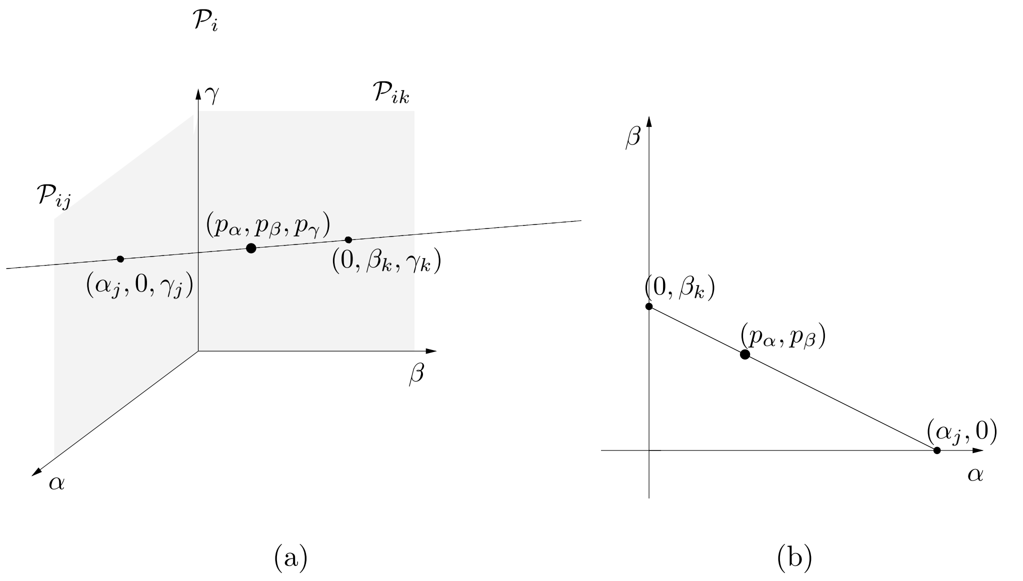


Fig. 6. (a) Dual representation of a 3D line, a plane containing this 3D line (the point $(p_\alpha, p_\beta, p_\gamma)$) and two of its projections (the points $(\alpha_j, 0, \gamma_j)$ and $(0, \beta_k, \gamma_k)$); (b) Same representation in a 2D space.

Using the *Red* function previously defined, the three cases depicted in Figure 7 may occur during the computation of the image of an interval $I = [AB]$ through a point:

- case (a): if p_α is strictly lower than A or strictly greater than B the image of I is a closed interval;
- case (b): if p_α is equal to A or B then the image of I is a half-open interval;

- case (c): if p_α is strictly between A and B, then the image of I is made of two half-open intervals.

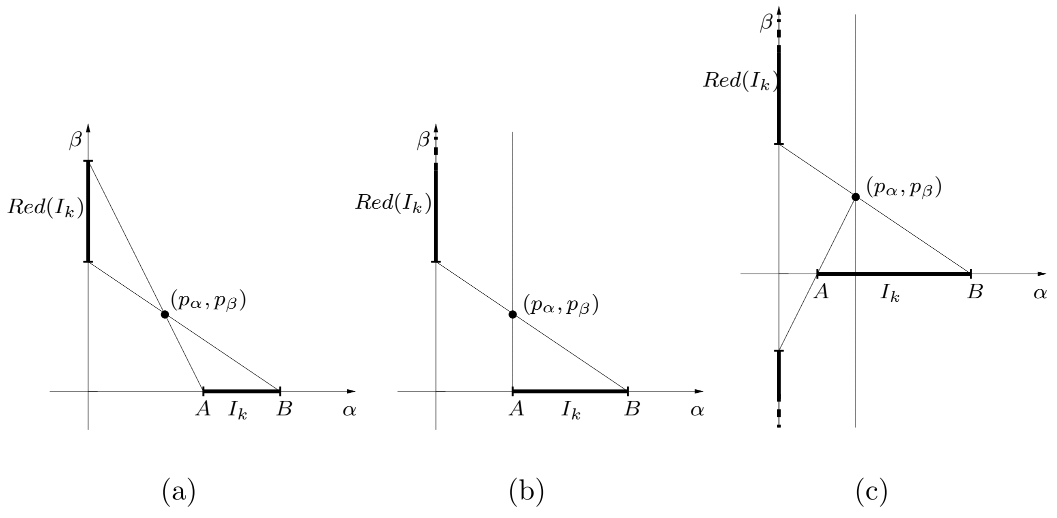


Fig. 7. Three possible cases for the computation of the image $Red(I_k)$ of a dual segment I_k according to a carrier plane which parameters are (p_α, p_β).

Thus, the reduction of a preimage I_k according to another preimage I_j is done in two steps in the parameter space $\mathcal{P}_i$:

- compute the image of I_j using the reduction function defined previously;
- compute the intersection between this image and I_k: the resulting interval is the new preimage I_k.

To complete the reductions in parameter space $\mathcal{P}_i$, the reverse operation (reduce I_j according to I_k) is performed using a reduction function similar to the one defined previously. This pair of reductions is computed in each parameter space $\mathcal{P}_x$, $\mathcal{P}_y$ and $\mathcal{P}_z$, such that each preimage I_x, I_y and I_z is reduced twice. Finally, we have the following result:

Proposition 7 *After the six reductions presented above, the preimages of the three projections of the set of voxels S represent exactly the set of 3D lines solution for S and embedded in the carrier plane.*

PROOF. The key point of the proof is that any point of the preimages I uniquely defines a 3D line in the Cartesian space. Indeed, since we assumed that the normal vector of the carrier plane has no zero coordinates, the couple made of a projection and the carrier plane defines a unique 3D line. Consider the preimage I_x of the projection of the set of voxels S onto the plane $(0yz)$, and the following reductions:

(1) I_x is reduced according to I_y: the points in I_x (and those in I_y) define

the 3D lines embedded in the carrier plane, and which projections are solutions for the projections of S onto $(0yz)$ and $(0xz)$;

(2) I_x is reduced according to I_z: the points in I_x (and those in I_z) define the 3D lines embedded in the carrier plane, and which projections are solutions for the three projections of S onto $(0yz)$, $(0xz)$ and $(0xy)$.

Eventually, the last pair of reductions (I_y and I_z) will only result in a reduction of I_y (I_z will remain unchanged) which defines at this point the 3D lines solutions for only two out of the three projections (unchanged after step (1)). After this cycle of six reductions, each preimage I is finally the set of 3D lines embedded in the carrier plane and whose projections are solutions for the three projections of S. □

This proposition moreover proves that these reductions ensure the projection preimages compatibility, *i.e.* that there exists a 3D line which projections contain the projected sets of pixels in their standard digitization (see Section 3.2).

3.3.2 Particular case when some components of the plane normal vector are zero

For any finite set of voxels, thus for any digital planar curve, a carrier plane with non-zero normal vector components can always be found. Therefore, the general case always holds. Nevertheless, using planes with zero normal vector components, when possible, can lead to nicer solutions (for a cube for instance).

Let us consider a plane p defined by $ax + by + cz + \mu = 0$. If two out of the three parameters a, b, c are zero, two out of the three projections of the lines embedded in p are fixed. Suppose for instance that a and b are equal to zero. Then, the two dual segments I_x and I_y are reduced to single points. On the other hand, the third dual segment I_z is defined as previously, and no reduction between the three projection preimages is required. This case can actually be handled as a 2D case.

Now consider that only one parameter is zero, say $a = 0$. In this case, one projection is fixed and the dual segment I_x is then again reduced to a single point. The other two dual segments shall be reduced accordingly. As in the general case, these reductions can be performed in a 2D space, and each projection is uniquely defined by its slope. The only parameter space in which both I_y and I_z can be represented is $\mathcal{P}_x$. However, the plane p cannot be represented in this parameter space and the general case reduction process cannot be applied. In this context, the idea is to perform the reduction process according to the fixed projection induced by p. Given the slopes α_z and α_y of two projections respectively in $\mathcal{P}_{xy}$ and $\mathcal{P}_{xz}$, the slope of the third projection is equal to $-\frac{\alpha_y}{\alpha_z}$ in $\mathcal{P}_{yz}$. Now if I_x is reduced to a fixed single point A_x, I_y and I_z must be reduced in order to fulfil the following condition: for all $\alpha_y \in I_y$ and all $\alpha_z \in I_z$, $-\frac{\alpha_y}{\alpha_z} = A_x$. This reduction process is illustrated in Figure 8.

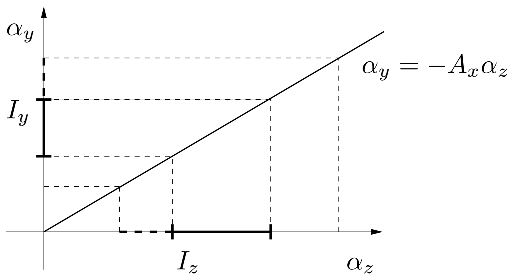


Fig. 8. Reduction of the projection dual segments I_y and I_z in the case of a carrier plane with one zero component (one projection is fixed): I_y and I_z are represented as thick segments, the part removed after reduction is dashed.

3.4 *Choice of fixed extremities*

Our algorithm is initialized with a point belonging both to the first voxel of the curve and to the carrier plane p (see Section 3.1). This point belongs to the intersection between a voxel and a Cartesian plane. From the definition of standard plane, we know that the voxels cut by a given plane are exactly the voxels of the standard digitization of this plane. The geometry of this intersection has been studied by Reveillès [20] and Andres et al. [21] who show that the only five intersection polygons possible are a triangle, a trapezoid, a pentagon, a parallelogram or an hexagon (Figure 9). They moreover characterize the shape of the intersection between a plane and a voxel according to the position of the voxel in the corresponding standard plane. In [20], Reveillès gives the arithmetical expression of intersection vertices coordinates. Thus, the initial point chosen in our algorithm is simply the barycentre of the vertices computed thanks to Reveillès [20] and Andres et al. [21] results.

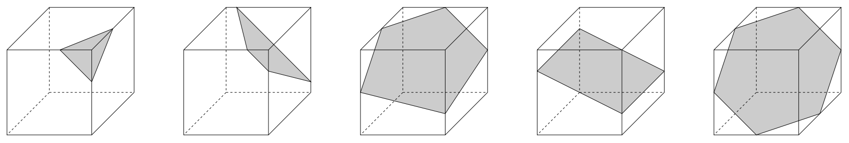


Fig. 9. The five possible intersections between a voxel and a plane.

3.5 *Overall algorithm*

Algorithm 1 finally presents the overall vectorisation algorithm for a 3D digital planar curve. In step 2, a first fixed point v_1 is chosen (see Section 3.4).

Variables i and k respectively count the number of voxels processed and the number of segments computed during the vectorisation algorithm. Steps 6 to 8 initialize the 2D preimages for the three projections of the new discrete segment s_k. The while loop between steps 9 and 15 processes the addition of a new voxel to s_k: on step 12, the three preimages are updated according to the new point (ensures that the projections are 2D discrete segments, see Section 3.2) and step 14 ensures the coplanarity thanks to the reduction process described in Section 3.3. Finally, when the voxel V_i cannot be added to the current segment s_k (one out of the three preimages is empty), we choose a solution line, compute a new fixed point and the process starts over.

Algorithm 1 Vectorisation of a 3D digital planar curve

```
VECTORISATION_PLANAR_3DCURVE(ordered set of voxels $V$, carrier plane $p$)
 1: $i \leftarrow 1$, $k \leftarrow 1$
 2: Let $v_1$ be a point in the first voxel $V_1$ and lying into the plane $P$. {Section
    3.4}
 3: while ($i \leq n$) do
 4:    $s_k \leftarrow \{V_i\}$ {$s_k$ is the current discrete segment}
 5:    $r_k^1 \leftarrow v_k$
 6:    $I_x = \mathcal{P}_{yz} \cap C_y^{-1}(v_k)$
 7:    $I_y = \mathcal{P}_{xz} \cap C_x^{-1}(v_k)$
 8:    $I_z = \mathcal{P}_{xy} \cap C_x^{-1}(v_k)$
 9:    while ($I_x \neq \emptyset$ and $I_y \neq \emptyset$ and $I_z \neq \emptyset$ and $i \leq n$) do
10:       i $\leftarrow$ i+1
11:       $s_k \leftarrow s_k \cup \{V_i\}$
12:       Compute the reductions of the three intervals according to the con-
          straints induced by the three projections of $V_i$. {Section 3.2}
13:       if ($I_x \neq \emptyset$ and $I_y \neq \emptyset$ and $I_z \neq \emptyset$) then
14:          Compute the six reductions of the intervals according to the plane
             $p$. {Section 3.3}
15:       end if
16:    end while
17:    if ($I_x = \emptyset$ or $I_y = \emptyset$ or $I_z = \emptyset$) then
18:       $s_k \leftarrow s_k - \{V_i\}$ and reset $I_x$, $I_y$ and $I_z$ as before adding $V_i$.
19:       $i \leftarrow i - 1$
20:    end if
21:    Choose a 3D solution line $l_k$ in the preimages $I$.
22:    Choose a point $v_{k+1}$ in $V_i$ belonging to the line $l_k$.
23:    $r_k^2 = v_{k+1}$
24:    $k \leftarrow k + 1$
25: end while
```

4 Complexity Issues

4.1 Theoretical study

Let us study the theoretical complexity of this algorithm. The algorithm is greedy and each voxel of the curve is processed only once. For each voxel, the reductions of the preimage intervals on line 12 can be done in constant time (intersection between two lines), but the simplification of the rational coordinates of the preimage extremities requires a linear time in the size of the rational numerators and denominators. The complexity of reduction steps according to the carrier plane (step 14) depends on the number of connected components of the preimages I. Indeed, we saw in Section 3.3 Figure 7(c) that the image of an interval through a point may be composed of two infinite parts, and consequently, the intersection between this image and a preimage could be made of more than one connected part. Nevertheless, we have the following proposition which ensures that the preimages I are made of one single connected part.

Proposition 8 *Given a 3D discrete segment S and a carrier plane p which cuts all the voxels of S, the set of 3D lines lying in p and crossing all the voxels of S is a connected set.*

PROOF. Consider the tiling defined by the intersection of the plane p and the voxels of S. The tiles are the intersections between a plane and a voxel, thus this tiling is made of convex polygons (see Section 3.4). Now let l and l' be two lines lying in p and crossing all the voxels of S. Then l and l' cross all the convex polygons of the tiling. Without loss of generality, suppose that l and l' are not parallel, they intersect in a point P inside $S \cap p$. Using this point as a pivot, we can transform continuously l into l'. Let l'' be a line between l and l' according to this transformation. Consider a voxel V_i in S and choose two points P_i and P'_i in $V_i \cap p$. The segment $[P_iP'_i]$ is included in $V_i \cap p$ since the tiles are convex polygons. By construction l'' crosses $[P_iP'_i]$, and thus crosses the tile $V_i \cap p$. $\square$

This proves that the preimages I are composed of one single interval, and thus that the reductions are done in constant time. Nevertheless, the same remark concerning the preimage rational coordinates simplification holds in this case, and the overall complexity for this phase of reductions is also linear in the size of the numerator and denominator integers.

Consequently, the theoretical complexity of Algorithm 1 is $\mathcal{O}(n \times s)$ where s is the size of the biggest integer value in the preimages coordinates. The value of s is difficult to bound since it depends on the strategy used to choose the solution line (step 21) and the new fixed extremity (step 22). Thus we do not give a theoretical bound on s for a particular strategy but we present some practical hints on how a good strategy can be fixed.

4.2 Hints on practical behaviour

In practice, we use the multiple precision library GMP [22] to handle the very large integers that may appear during the reconstruction process. With this library, we can do exact integer and rational computations along the algorithm, which prevents from numerical errors that could compromise the reversibility of the reconstruction. It is nevertheless important to use integer numbers as small as possible during the reconstruction since all the basic operations like multiplication, division and simplification (see previous section) get costy when the size of the numbers increases.

In Algorithm 1, choices of a solution line and a new fixed point (steps 21 and 22) may induce a huge increasing of the numbers' size. For instance, the basic idea of choosing the middle point is not satisfactory from this point of view: indeed, the middle fraction of two fractions $\frac{a}{b}$ and $\frac{c}{d}$ is $\frac{ad+bc}{2bd}$, which means that the result's numerators and denominators are twice longer than those of the original fractions. In the following, we use the notations of Algorithm 1.

To choose a solution line l_k, we have to pick a point in a preimage I, i.e. we have to choose a fraction between two other fractions. We propose two strategies:

(1) find the best solution: given two fractions, we look for the fraction in between with the smallest denominator. This can be done using a binary tree ordered representation of all rational fractions (called Stern-Brocot tree [23–25]). In this tree, each node is an irreducible rational fraction, and the smallest fraction between two fractions is the nearest common ancestor of the two corresponding nodes. This solution gives good results, but is very costy if the numbers get big;
(2) find an intermediate solution: in an interval of size $\frac{1}{d}$, $d > 1$, we can find a fraction with denominator d. Computing this fraction is really fast and is on average much smaller than the middle fraction.

Moreover, the new fixed point v_{k+1} lies in the intersection between line l_k and voxel V_i. The two extremities of this intersection segments lie on V_i' faces and then have at least one half-integer coordinate (Figure 10). Consequently, choosing one of these two points as v_{k+1} is a reasonable solution in order to slow integer size increase. Since l_k is a solution line for the current 3D discrete segment s_k, and since s_k is at least 2 voxels long, then the first extremity of the intersection segment lies on the common face of V_{i-1} and V_i. Thus, it is strictly included in the discrete curve and can be chosen as v_{k+1} (see Figure 10). This property cannot be ensured for the other extremity.

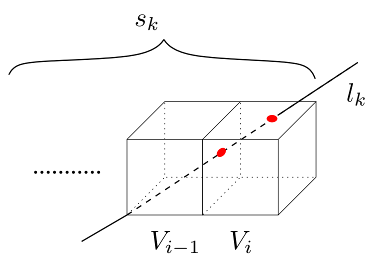


Fig. 10. Choice of a new fixed point v_{k+1} on the line l_k and inside the voxel V_i: choosing the point on the interface bewtween V_{i-1} and V_i enable a slower increase of integer sizes.

In the following, we give some results about execution times and size of the biggest integer. To evaluate the size of the maximum integer used in the reconstruction process, since checking every integer value used would be too costy, we only point out the size of the fixed point coordinates denominators : indeed, the fixed point coordinates determines the preimage extremities initialization, and has therefore an influence on the solution lines, and the coordinates numerators depend on the image size and the denominators.

Using those strategies, or at least one of the two, improves greatly the execution time of the algorithm together with the memory needed. Indeed, when the middle points are chosen both for l_k and v_{k+1}, the algorithm is very slow : for instance, for a sphere of radius 10, it takes only 3.7 seconds and the biggest integer is 84 bits long to compute polygons for the 73 DPS of the surface with our strategies, whereas it takes 60.7 seconds and the biggest integer is 126890 bits long if the middle points are chosen. Thus we see that even for very small volumes, integer size is a very important issue. For bigger volumes (more than 10000 surface surfels for instance) computation times are usually greater than 20 minutes, which is not reasonable.

Table 1 gives some execution time results for a collection of objects together with the size (in bits) of maximum integers used during the reconstruction process. Time results reported do not take into account the segmentation process, but only the reconstruction of all the discrete faces. Since choosing middle points for l_k and v_{k+1} gives very bad results, we only report results assuming than one good solution is used: whether l_k is a function of the preimage interval length ($l_k \leftarrow f(length)$) or v_{k+1} belongs to a voxel's face ($v_{k+1} \leftarrow f(\frac{1}{2})$). This table shows that using those strategies enables to work with quite small integers, and thus to achieve computation times around or under one minute for volumes $100 \times 100 \times 100$, especially if both improvements are used (last column).

object	# surfels	# DPS	$v_{k+1} \leftarrow$ middle, $l_k \leftarrow f(length)$		$v_{k+1} \leftarrow f(\frac{1}{2})$, $l_k \leftarrow$middle		$v_{k+1} \leftarrow f(\frac{1}{2})$, $l_k \leftarrow f(length)$	
			t(s)	size	t(s)	size	t(s)	size
sphere25	12318	363	25.2	567	22.0	717	21.6	319
torus	26416	723	84.0	705	83.0	5968	75.3	176
hand	10240	1042	34.5	727	31.4	348	29.8	126
fandisk	20058	1113	71.9	1605	63.2	935	62.6	186
Al	21556	1703	67.1	801	67.7	397	66.4	143

Table 1
Comparison of execution times according to different choices for l_k and v_{k+1}

5 Results: images, modelling and compression

5.1 Modelling results

The result of this algorithm over a DPC is represented in Figure 11. In (a) and (c), the voxels are represented by their center, so that the connectivity is easily visible. Moreover, the discrete points belonging to the same digital plane as the DPC (for a given carrier plane) are depicted. In this case, the DPC is the boundary of a given digital plane segment (DPS for short). In (b), a polygonal curve embedded in a carrier plane of the DPC is computed: the voxels are represented on this figure to illustrate the inclusion of the polygonal curve into the discrete curve. Finally, the polygon computed is represented in (c), together with the set of discrete points of the carrier plane: the standard digitization of the polygon is exactly the set of discrete points.

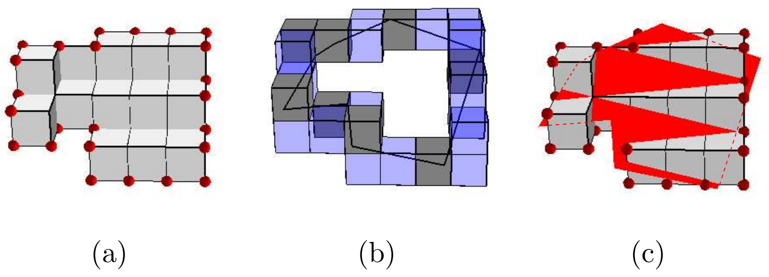
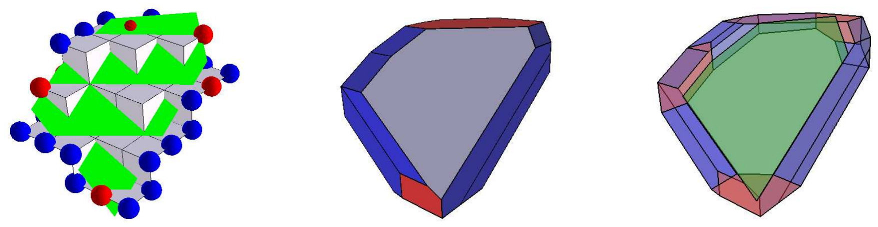


Fig. 11. Different steps of the polygonalization of a discrete face: (a) the DPC discrete points, (b) computation of the polygonal line, (c) DPC and polygon superimposed.

The reconstruction process we present in this paper is based on the standard digitization scheme as said in Section 2.1. This model, analytically defined by Andres in [10], is geometrically consistent and well adapted to modelling applications. Indeed, standard simplexes are well defined in this framework, as a set of linear discrete inequalities for faces, edges and vertices. Then our reconstruction algorithm for DPC completes the following inverse problem: given a digital plane segment which boundary is a DPC, find an analytical description of this set as a discrete polygon. The constraints defining the DPS are computed directly from the carrier plane, and those defining its boundary are computed from the polygonal curve edges parameters and vertices coordinates. The construction follows the definition of a standard simplex, and consists in finding the constraints for each basic element for the three projections.

An example is presented in Figure 12: on the left (a), a digital plane segment, its boundary, and the computed polygon are depicted: vertices discrete points are represented in red, edges discrete points in blue. The plane defines two inequalities, the six edges define 18 inequalities (one inequality for each edge and each projection), and the vertices define six inequalities as a bounding box for the polygon. The convex polyhedron resulting from those constraints is depicted in (b) and (c), where the constraints related to vertices are in red, those related to edges in blue, and the two inequalities related to the plane in light blue. The DPS discrete points are exactly the set of grid points included in this polyhedron.



(a) (b) (c)

Fig. 12. Analytical description of a DPS: (a) DPS with vertices (in red) and edges (in blue) discrete points, and reconstructed polygon; (b) analytical view; (c) reconstructed polygon and analytical view.

5.2 Application on digital surfaces

This algorithm can be used to analytically represent the surface of a digital object by a set of polygons. The prerequisites to this application are, first, to set the definition of surface used (surface elements and connectivity), and next to design an algorithm for the decomposition of the surface into DPS, such that the boundary of those DPS are digital planar curves. Our algorithm is then applied on each DPS of the decomposition.

Concerning the definition of discrete surfaces, two main approaches exist: the surface elements are either object voxels or object voxels' faces. In this work, we define the object surface as the set of voxels' faces (called *surfels*, see Figure 1(b)) belonging to an object voxel and a background voxel. In other words, the surface is composed of the faces visible when the object is displayed. This definition of surface is well-adapted to our framework that is based on standard discrete planes: discrete (grid) points are not the object voxels but the vertices of those voxels (called *pointels*, see Figure 1(b)). Pointels of a surface form a 2-connected set, which is consistent with the use of standard planes.

Using standard planes also induces the connectivity we consider for the object. Standard planes have a combinatorial structure of 2-dimensional manifolds [26,27]. Thus, the discrete surface we work on should have the properties of a $2D$ combinatorial manifold as well, which implies that 2-connectivity has to be considered for the discrete object. An example of surfels adjacency using 2-connectivity is depicted in Figure 13 (a)-(b).

A decomposition algorithm consists in labelling each pointel of the surface. Let P be a set of discrete points of the same DPS (same label), and let p be a carrier plane of P. Then P must fulfil the following conditions:

- the four pointels adjacent to the same surfel belong to a common DPS;
- the projection of P along its principal direction (direction given by the maximum parameter of p's normal vector) is a set of 1-connected discrete points;
- P is homeomorphic to a topological disk.

These conditions ensure that each DPS is a combinatorial 2-manifold with boundary, and that this boundary is a DPC. Notice that the first condition implies that some pointels may belong to several DPS (on the DPS boundaries).

These conditions are nevertheless not sufficient to ensure that one pointel is visited only once during a DPS boundary tracking. Such a pointel may lead to self-crossing polygonal faces, thus a fourth condition may be added: DPS should not contain surfels connected only by a pointel or not neighbours according to the 2-connectivity (see Figure 1(c)-(d)).

Figure 13(a) gives an example of a result we get with a decomposition algorithm fulfilling those four conditions (see [28]). Note that the top of the torus is decomposed into two DPS instead of one, so that each DPS is homeomorphic to a disk.

Algorithm 1 is applied on each DPS boundary of the surface decomposition. This results in a set of polygons the standard digitization of which is exactly the surface pointels of the initial discrete object. Figure 14 presents two results

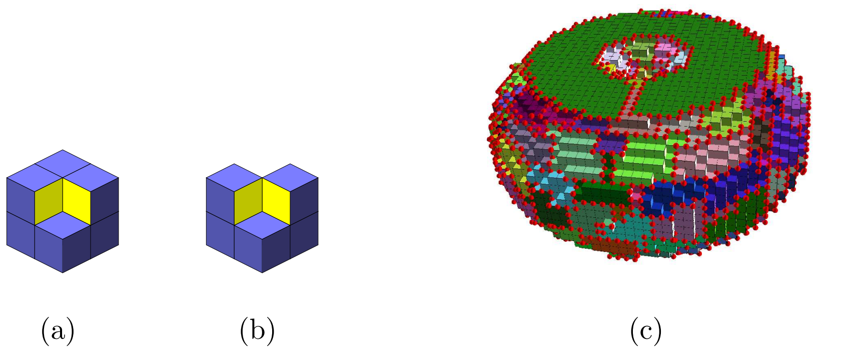


Fig. 13. (a)-(b) Surfels adjacency with 2-connectivity : in (c) the two yellow surfels are neighbours, whereas they are not in (b); (c) Example of a decomposition result on a torus (DPS boundaries are depicted).

over a torus and the image named "Al" (used for time comparison results in Section 4.2): the initial discrete object surface decomposed into DPS is depicted on the left, and the set of polygons computed is represented on the right. These pictures show that even if the reconstruction process gives an exact representation of the object's surface, the visualization is not satisfactory since the polygons are not linked together. This is nevertheless a compact representation of an object that preserves its geometry, and we can even go further on the compression issue as we see in Section5.3.

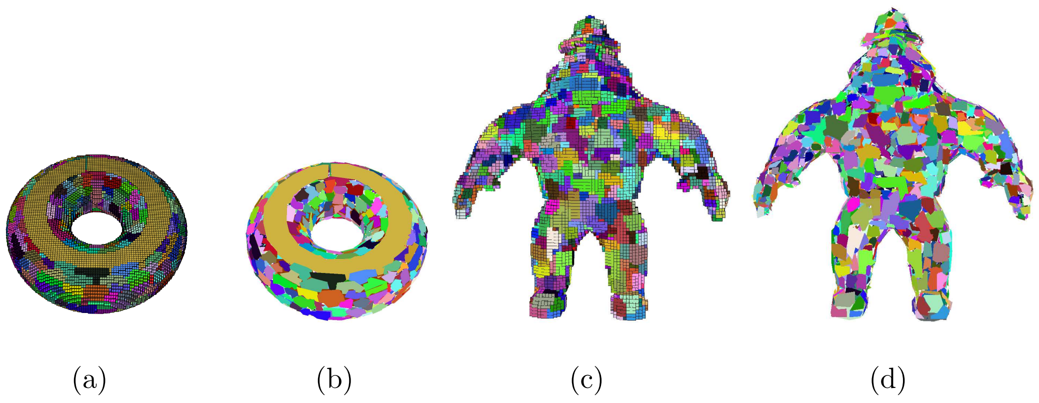


Fig. 14. Results of the vectorisation algorithm over complex objects surfaces.

5.3 Prospective work: lossless compression

Decomposing a discrete surface into DPS and then computing a polygonal face as the exact representation of the DPS enables to reduce the redundancy induced by the discrete structure of the data. Indeed, a set of coplanar voxels is now represented by a single polygon. Thus, a natural application of this work is 3D discrete objects lossless compression. In this section, we propose first thoughts about this problem, but this is still a prospective research.

The problem is to find a compact and exact representation of the polygons computed by our algorithm. The storage of one polygon only requires the vertices coordinates. Nevertheless, as we said in Section 4.1, the bound on the size of the rational fractions defining the vertices coordinates is huge, and this does not lead to an efficient encoding. To overcome this problem, one could think of rounding the vertices coordinates in order to store floating point numbers, but in this case, ensuring the reversibility property, and thus lossless compression is not possible anymore.

Other encoding should then be proposed and a promising idea is to take advantage of the following properties of digital planes and digital lines:

- DPS normal vector coordinates are bounded by the size of the DPS;
- digital segment normal vector coordinates are bounded by the digital segment length;
- standard planes are "nearly" functional: at most two voxels have the same projection pixel along the main direction of the plane (given by the maximum of normal vector coordinates).

Thus, a DPS may be encoded in the following way:

(1) store the normal vector coordinates of the DPS;
(2) store an encoding of the DPS boundary projection along the main direction;
(3) for the first extremity of each edge, store a flag pointing out if the 3D edge extremity is the upper or the lower voxel (in the case of non bijective projection).

The second step is the encoding of a 2D digital curve decomposed into digital straight segments, and some encoding schemes have been proposed to solve this problem [29].

Roughly speaking, if we denote by n_p the number of DPS, s the maximum number of points in a DPS, n_e the total number of edges, and l the maximum length (in voxels) of the edges, the number of bits required to store the surface of an object in a 2^n bounding box would be (without any entropy coding):

$$N = \overbrace{(n_p \times 3 \times \log_2 s)}^{\text{DPS normal vectors}} + \overbrace{(n_p \times (\log_2 s + \log_2 n))}^{\text{DPS positions}} + \overbrace{(n_e \times 2 \times \log_2 l)}^{\text{edge directions}} + \overbrace{(n_e \times (\log_2 l + \log_2 n))}^{\text{edge positions}} + \overbrace{n_e}^{\text{correction}}$$

For example, the data collected for a sphere of radius 25 are the following: $n = 52$, $n_p = 363$, $n_e = 2907$, $s = 119$ and $l = 12$. With these values, $N = 61095$ bits, i.e. 7.6 kbytes are required to store the surface of the object using this method. In comparison, the size of the uncompressed raw file is 144 kbytes, and the file compressed with gzip is about 3 kbytes.

These preliminary results are promising especially because many improvements may be proposed, either to encode the polygons as computed by the algorithm presented in this paper, or modifying the algorithm to obtain a better encoding (taking advantage of the common boundaries of adjacent polygons for instance). But this is out of the scope of this paper.

6 Conclusion

In this paper we proposed the first algorithm to compute a planar polygonal curve from a digital planar curve. The computed polygonal curve is an exact representation of the digital one and also provides an analytical representation of a digital planar curve. This algorithm also solves the inverse problem of the description of a digital plane segment as a discrete polygon made of a face, edges and vertices.

A study of the theoretical complexity of this algorithm is provided, and a discussion on practical behaviour concerns is proposed.

Setting an adapted but simple framework for the decomposition of a digital surface into digital plane segments, we also gave some results on the application of this algorithm on the boundary of each DPS of a surface. We get a set of polygons modelling the discrete surface in a reversible way: the standard digitization of each polygon is exactly a discrete face of the segmentation.

Future works are related both to theoretical improvements and more practical applications. As we saw in the last part, a first interesting application and prospective work concerns 3D discrete objects compression. Few methods dedicated to this kind of objects exist, and consequently a lot of work remains to be done. Another interesting application, that would combine this algorithm and blurred digital planes, concerns digital surface denoising.

Finally, we saw that our algorithm gives an exact analytical modelling of one digital plane segment. For visualization and modelling purposes of discrete surfaces, an important future work is to extend this algorithm in order to get a consistent geometrical description of a discrete surface as a set of faces, edges and vertices. This should moreover lead to the construction of a hole and intersection free polygonal surface while preserving the reversibility property. This problem may be related to the polygonal reconstruction of several adjacent discrete regions in 2D.

Acknowledgments

The authors thank the anonymous reviewers for their comments which contributed to the improvement of this paper. Many thanks also to D. Coeurjolly and P.-F. Dutot for fruitful discussions. Some of the test data come from the IAPR Technical Committee 18 webpage (`http://www.cb.uu.se/~tc18/`). Many thanks to the contributors.

References

[1] Rosenfeld, A., Klette, R.: Digital straightness. In: Int. Workshop on Combinatorial Image Analysis. Volume 46 of Electronic Notes in Theoretical Computer Science., Elsevier Science Publishers (2001)

[2] Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Series in Computer Graphics and Geometric Modelin. Morgan Kaufmann (2004)

[3] Brimkov, V., Coeurjolly, D., Klette, R.: Digital planarity - a review. Technical report, Laboratoire LIRIS, Université Claude Bernard Lyon 1 (2004) `http://liris.cnrs.fr/publis/?id=1933`.

[4] Borianne, P., Françon, J.: Reversible polyhedrization of discrete volumes. In: DGCI, Grenoble, France (1994) 157–167

[5] Françon, J., Papier, L.: Polyhedrization of the boundary of a voxel object. In Bertrand, G., Couprie, M., Perroton, L., eds.: DGCI. Volume 1568 of LNCS., Springer-Verlag (1999) 425–434

[6] Coeurjolly, D., Guillaume, A., Sivignon, I.: Reversible discrete volume polyhedrization using Marching-Cubes simplification. In Latecki, L.J., Mount, D.M., Wu, A.Y., eds.: SPIE Vision Geometry XII. Volume 5300 of Proceedings of SPIE. (2004) 1–11

[7] Burguet, J., Malgouyres, R.: Strong thinning and polyhedrization of the surface of a voxel object. In Borgefors, G., Nyström, I., Sanniti di Baja, G., eds.: DGCI. Volume 1953 of LNCS., Uppsala, Sude, Springer-Verlag (2000) 222–234

[8] Sivignon, I., Breton, R., Dupont, F., Andres, E.: Discrete analytical curve reconstruction without patches. Image and Vision Computing **23** (2005) 191–202

[9] Sivignon, I., Dupont, F., Chassery, J.M.: Reversible polygonalization of a 3D planar discrete curve: Application on discrete surfaces. In Andres, E., Damiand, G., Lienhardt, P., eds.: DGCI. Volume 3429 of LNCS., Poitiers, France, Springer-Verlag (2005) 347–358

[10] Andres, E.: Discrete linear objects in dimension n : The standard model. Graphical Models and Image Processing **65** (2003) 92–111

[11] Andres, E., Nehlig, P., Françon, J.: Tunnel-free supercover 3d polygons and polyhedra. In: Computer Graphics Forum. Volume 16., Eurographics'97 (Budapest), Blackwell Publishers (1997) C3–C13

[12] Lincke, C., Wüthrich, C.A.: Surface digitizations by dilations which are tunnel-free. Discrete Applied Math. **125** (2003) 81–91

[13] Wu, L.D.: On the chain code of a line. IEEE Trans. on Pattern Anal. and Mach. Intell. **4** (1982) 347–353

[14] Kim, C.E.: On cellular staight line segments. Computer Graphics and Image Processing **18** (1982) 369–381

[15] Debled-Rennesson, I., Reveillès, J.P.: A linear algorithm for segmentation of digital curves. International Journal of Pattern Recognition and Artificial Intelligence **9** (1995) 635–662

[16] Dorst, L., Smeulders, A.N.M.: Discrete representation of straight lines. IEEE Trans. on Pattern Anal. and Mach. Intell. **6** (1984) 450–463

[17] McIlroy, M.D.: A note on discrete representation of lines. AT&T Technical Journal **64** (1985) 481–490

[18] Lindenbaum, M., Bruckstein, A.: On recursive, $\mathcal{O}(n)$ partitioning of a digitized curve into digital straight segments. IEEE Trans. on Pattern Anal. and Mach. Intell. **15** (1993) 949–953

[19] Hough, P.: Method and means for recognizing complex patterns. United States Patent, n3, 069, 654 (1962)

[20] Reveillès, J.P.: The geometry of the intersection of voxel spaces. In Fourey, S., Herman, G.T., Kong, T.Y., eds.: IWCIA. Volume 46 of Electronic Notes in Theoretical Computer Science., Philadeplhie, Elsevier (2001)

[21] Andres, E., Sibata, C., Acharya, R., Shin, K.: New methods in oblique slice generation. In: SPIE Medical Imaging. Volume 2707 of Proceedings of SPIE. (1996) 580–589

[22] (GNU Multiple Precision Arithmetic Library) http://www.swox.com/gmp/.

[23] Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers. Oxford Society (1989)

[24] Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics. Addisson-Wesley (1994)

[25] Hayes, B.: On the teeth of wheels. Computing Science - American Scientist **88** (2000) 296–300

[26] Françon, J.: Discrete combinatorial surfaces. Graphical Models and Image Processing **51** (1995) 20–26

[27] Françon, J.: Sur la topologie d'un plan arithmétique. Theoretical Computer Science **156** (1996) 159–176

[28] Sivignon, I., Dupont, F., Chassery, J.M.: Discrete surface segmentation into discrete planes. In Klette, R., Zunic, J., eds.: IWCIA. Volume 3322 of LNCS., Springer-Verlag (2004) 458–473

[29] Kovalevsky, V.: Applications of digital straight segments to economical image encoding. In Borgefors, G., Nyström, I., Sanniti di Baja, G., eds.: Discrete Geometry for Computer Imagery. Volume 1953 of LNCS., Uppsala, Sude, Springer-Verlag (1997) 222–234