Discrete analytical curve reconstruction without patches

Isabelle Sivignon, Rodolphe Breton, Florent Dupont, Eric Andres

Discrete Analytical Curve Reconstruction without patches

Isabelle Sivignon [a,*],
Rodolphe Breton [b],Florent Dupont [c],Eric Andrès [b],

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

[b] *Laboratoire SIC, Université de Poitiers*
FRE 2731 CNRS
BP 30179
86962 Futuroscope Chasseneuil Cedex, France

[c] *Laboratoire LIRIS - Université Claude Bernard Lyon 1*
FRE 2672 CNRS
Bâtiment Nautibus - 8 boulevard Niels Bohr
69622 Villeurbanne Cedex, France

Abstract

Invertible Euclidean reconstruction methods without patches for 2D and 3D discrete curves are proposed. From a discrete 4-connected curve in 2D, or 6-connected curve in 3D, the proposed algorithms compute a polygonal line which digitization with the standard model is equal to all the pixels or voxels of the curve. The framework of this method is the discrete analytical geometry and parameter spaces are used in order to simplify the algorithms. Moreover, the reconstructed polyline is more compact than classical methods such as the Marching Cubes.

Key words: discrete object, invertible Euclidean reconstruction.

* Corresponding Author. E-mail: sivignon@lis.inpg.fr - Tel: +33 476826467 - Fax: +33 476826384

1 Introduction

The reconstruction of discrete objects is mainly performed in practice with the "Marching Cubes" method [1] (and all its follow ups). For a couple of years another approach, based on discrete analytical geometry, is investigated in the discrete geometry community. The aim is to decompose the boundary of a discrete object into discrete analytical polygons and then these polygons into Euclidean polygons. The method has to be *invertible*, i.e. the discretization of the reconstructed boundary has to be equal to the original discrete object. We do not want any information to be added nor lost. The aim of this new approach is to provide a more compact reconstruction. Several other attempts have already been made in this direction that are not satisfying and usually not invertible (see [2] for details). Indeed, most the algorithms consist in decomposing a discrete curve (surface) into discrete segments (planes) [3] [4] without looking for an exact Euclidean equivalent of the discrete object.

Our method is based on Vittone's recognition algorithm [5] for the decomposition of the discrete boundary into discrete line pieces in 2D. The analytical framework is provided by the standard discrete analytical model that defines 2D and 3D discrete polygons [6].

In [7], we proposed a working algorithm for the 2D case and some hints on how to tackle the 3D surface reconstruction problem. The method we proposed for 2D curves worked basically as follows: a discrete 4-connected boundary is decomposed with Vittone's algorithm [5] into discrete line segments in 2D. For each discrete segment a corresponding Euclidean line segment was chosen in the set of all possible solutions (set provided by Vittone's algorithm). The problem we faced then was that these different Euclidean lines could intersect outside the discrete line segments losing the reversibility property. To avoid that, patches (small line segments) where added that force the different Euclidean lines to intersect inside the vertex pixels of the discrete line segments preserving the reversibility property. Nevertheless, extension to dimension 3 is not straightforward. Indeed, the equivalent of patches in 3D is small polygons which are a lot more difficult to define than segments. Although some 3D solutions where proposed in [7], they are not very easy to set up.

For that reason we propose in this paper another way of reconstructing a discrete 4-connected curve without patches. The method works basically as follows: we still use Vittone's algorithm. Instead, this time, of performing a recognition of discrete line segments and then replacing them with Euclidean line segments, both are performed at the same time. Indeed, the main idea is to force the first extremity of the segments of the reconstructed polyline to lie inside the discrete curve. Hence, the discrete segment recognition algorithm is constrained by this fixed point. Finally, post-treatments are no more needed

to ensure the reversibility of the solution.

A second part of this work shows how to extend this algorithm to the polygonalization of 6-connected 3D discrete lines. We first propose an algorithm to recognize 3D standard line segments and show how the polygonalization process induces many simplifications of this algorithm. Basically, the recognition algorithm relies on three simultaneous 2D recognitions on the projections of the 3D curve. We show that an extra condition has to be settled in order to ensure the correctness of the algorithm. As to the polygonalization process for 3D curves, it works mainly as for the 2D curves.

This paper is organized as follows: Section 2 presents some recalls on the tools of discrete geometry we need. In Section 3, the new polygonalization algorithm for 2D standard curves is presented and results are given. Section 4 deals with the 3D curve polygonalization problem: we present a new algorithm to polygonalize in an invertible way any 3D 6-connected curve. Some results are proposed.

2 Recalls on discrete geometry

In this section, we present some basic notions of discrete geometry that are used in this work. We first introduce the standard digitization model and then present the notion of duality we use to recognize discrete segments.

2.1 The standard model

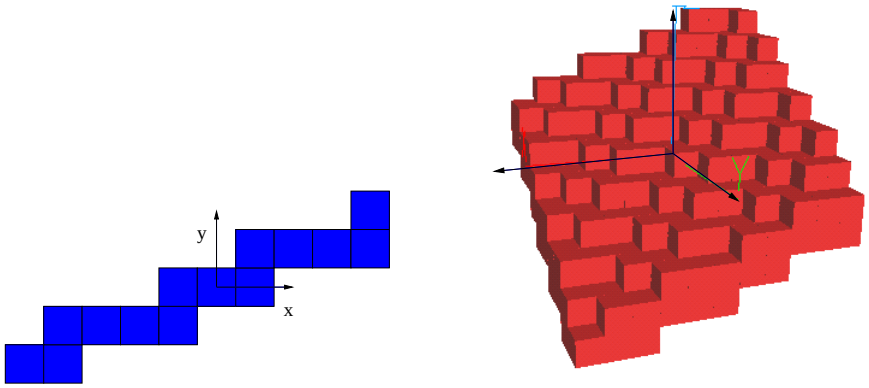
The standard digitization of a Euclidean object consists in all the pixels (resp. voxels) that are cut by the object. The standard lines (resp. planes) can be defined arithmetically [8].

Definition 1 *A discrete standard line (resp. plane) of parameters* (a, b, μ) *(resp.* (a, b, c, μ)*) is the set of integer points* (x, y) *(resp.* (x, y, z)*) verifying:*

$$-\omega \leq ax + by(resp. + cy) + \mu < \omega$$

where $\omega = \frac{|a|+|b|\,(resp.+|c|\,)}{2}$*. A standard line (resp. plane) is a 4-connected line (resp. 6-connected plane).*

Remark that the standard digitization requires a so-called orientation convention since the inequalities are not strictly symmetric. This is done by ensuring that $a > 0$ or if $a = 0$ that $b > 0$ or, in the case of planes, if $a = b = 0$ that $c > 0$. An illustration of this definition is given in Figure 1.



(a) (b)

Fig. 1. (a) A standard line of parameters $(2, -5, 0)$; (b) A standard plane of parameters $(2, 3, 5, 0)$.

A 3D standard line is a 6-connected discrete line which projections are 2D standard lines. The exact analytical definition together with an example are given in Section 4.

If we denote $St(O)$ the standard digitization of the object O, the following useful properties can be derived from the geometrical definition of this model: $St(O_1 \cap O_2) \subseteq St(O_1) \cap St(O_2)$ and $St(O_1 \cup O_2) = St(O_1) \cup St(O_2)$.

2.2 Parameter space

In image processing and especially pattern recognition, the dual transformation called Hough Transform [9] is classically used (see [10] for an overview on this method). This transformation is indeed very efficient to recognize parametric shapes in a given image. The general principle of this transformation is to work in a parameter space where each point represents a shape of given parameters. Thus, each selected point in an image is represented by all the shapes that go through it in the parameter space.

In this work, we use a parameter space $(0\alpha\beta)$ where a point (α_0, β_0) stands for the Euclidean line of equation $\alpha_0 x - y + \beta_0 = 0$. Then, each point (x_0, y_0) in the Cartesian space maps to the line $\alpha x_0 - y_0 + \beta = 0$ in the parameter space. Figure 2 illustrates the properties [11] which link the Cartesian space with this parameter space. More recently, many geometrical properties on Hough transforms have been studied (see for instance [12] and [13]).

This parameter space $\mathcal{P}$ is also very useful in the framework of discrete geometry. Indeed, consider a straight line $y = \alpha_0 x + \beta_0$ $(0 \leq \alpha_0 \leq 1)$, the digitization

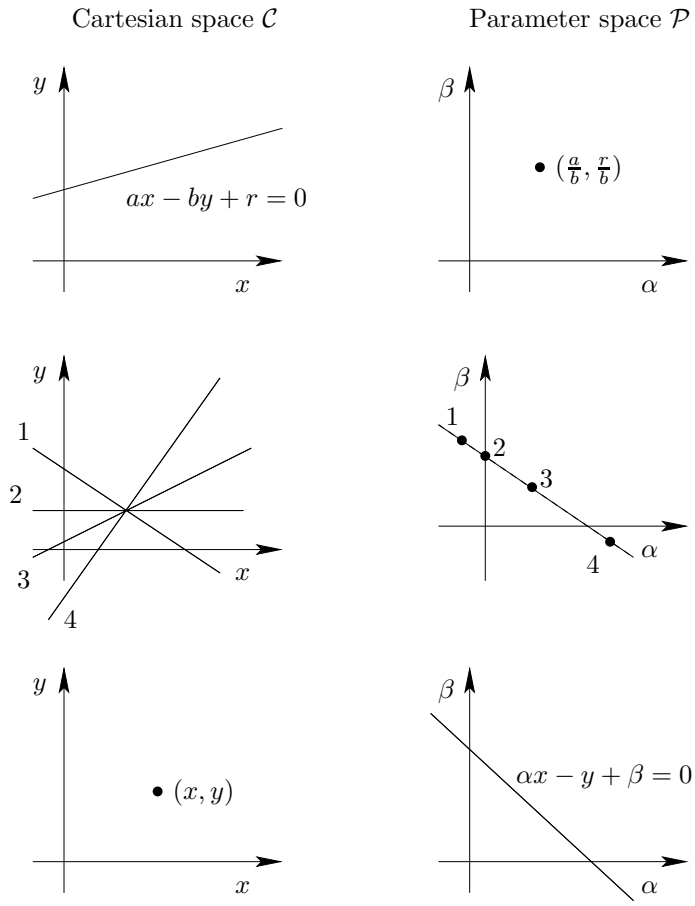


Fig. 2. Illustration of the links between the Cartesian space and the parameter space.

of this line along the Object Boundary Quantization (see [14]) on an $n \times n$ grid is the set of grid point $L = \{(x, y) \in n \times n \mid \lfloor \alpha_0 x + \beta_0 - y \rfloor = 0\}$. Then we can define the domain of a set of discrete points.

Definition 2 *Let S be a set of discrete points. The domain of S denoted $Dom(S)$ is the set of parameters (α, β) verifying:*

$$Dom(S) = \{(\alpha, \beta) \mid \forall (x, y) \in S,\ 0 \leq \alpha x - y + \beta < 1\}$$

Then, the domain of a set of pixels is either empty (if the pixels are not collinear) or a convex set since it is defined as the intersection of linear inequalities in the parameter space. Many works have been achieved in order to characterize this domain [15–17], and an important result is that if S is a connected discrete segment, then the domain is a (3 or 4)-vertex convex polygon that can only have one of the five shapes illustrated in Figure 3.

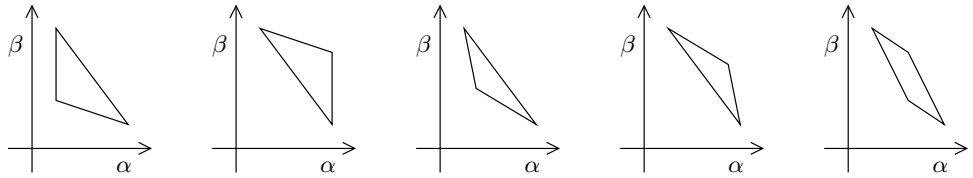


Fig. 3. The 5 possible shapes of the solution set.

3 Reconstruction of a 2D discrete simple curve

3.1 Principle

The reconstruction algorithm works on 4-connected curves $P = (P_1, \ldots, P_n)$. The algorithm starts with a Euclidean point p inside the starting point pixel P_1 of the discrete curve. A recognition direction is chosen once and for all. There is a need to choose a recognition direction especially in case of a closed curve since you can choose two directions from a given starting point. A recognition process determines if a set of pixels is a discrete line segment and if there exists a Euclidean line d that passes through p and through all the pixels of the discrete segment. As long as that holds, a new point P_k of the curve, in the recognition direction, is added. Once it does not, a new Euclidean point on the Euclidean line d and inside the last but one pixel is chosen, and the recognition process is started over along the curve until we reach P_n.

3.2 Starting point and Recognition direction

The method presented in this paper depends on the starting point and on the recognition direction. For each different choice of starting point and recognition direction we get a different reconstruction. We propose a convention which defines a starting point and a recognition direction for any curve. Thus, applying this convention, the reconstruction process always provides the same solution for a given discrete curve. Open and closed simple curves lead to different conventions:

- an open simple curve: the starting point is the curve end-point with lowest abscissa and lowest ordinate in the case of two end-points having the same abscissa. Moreover, the starting point chosen actually induces the recognition direction.
- a closed simple curve: the starting point is the curve point with lowest abscissa and lowest ordinate in the case of multiple points having the same abscissa. The clockwise direction is chosen as recognition direction.

This strategy can of course be improved. Different alternative solutions are under investigation. One improvement could be provided by choosing discrete cusps as starting points as in [7]. Other cases such as closed non simple curves or open curves with vertices of degree 3 or 4 (a discrete point has at most 4 4-connected neighbours) can be treated pretty much in the same way, as what is presented in section 3.3.2 except that there are more constraints. This is however still under investigation because for multiple neighbouring regions, topological information is helpful.

3.3 Recognition process

The recognition process is based on the recognition algorithm proposed by Vittone [18]. This algorithm tells if a set of 8-connected pixels is a discrete line segment and provides the domain of this set of pixels in the parameter space (see Section 2 and [19] for more details).

The recognition algorithm proposed by Vittone updates the domain in the parameter space as new points of the discrete curve are added. When the parameter space polygon is empty, the given set of pixels in not a discrete straight line segment anymore.

Different modifications to Vittone's algorithm needed however to be done to adapt it to our problem. Firstly, Vittone's algorithm works with 8-connected curves whereas we consider 4-connected curves. A simple shear transform maps the pixels of a 4-connected curve to the pixels of a 8-connected curve: for instance, a pixel (x, y) of a curve lying in the first octant maps to the pixel $(x+y, y)$. Secondly, Vittone's algorithm provides parameter values that do not correspond to what's usual for a discretization. Let's consider the first octant. Vittone's algorithm recognizes a discrete straight line $0 \leq ax - by + \mu < b$ (with $b \geq a \geq 0$) as a line with parameters $(\frac{a}{b}, \frac{\mu}{b})$. If we have a Euclidean line $ax - by + \mu = 0$ (with $b \geq a \geq 0$) the 8-connected discretization of this line provides the following discrete analytical line: $-\frac{b}{2} \leq ax - by + \mu < \frac{b}{2}$. It is a centered solution. Vittone's algorithm would therefore recognize the discretization of the straight line $ax-by+\mu = 0$ as a straight line of parameter $(\frac{a}{b}, \frac{\mu}{b} + \frac{1}{2})$ and not $(\frac{a}{b}, \frac{\mu}{b})$ as expected. A translation of the solutions is needed to correct this. Table 1 presents these corrections. These transformations lead to translations of the domain's vertices, and in the following, the so-called "domain" stands for this translated polygon.

3.3.1 Polygonalization algorithm

The polygonalization algorithm is described in Algorithm 1. We detail each step of this algorithm referring to its line numbers.

	8-connected	shear trans.	4-connected
Vittone's settings	$0 \leq ax - by + \mu < b$	$\overset{b=b-a}{\longrightarrow}$	$0 \leq ax - (b-a)y + \mu < b$
	$\downarrow \mu = \mu - \frac{b}{2}$		$\downarrow \mu = \mu - \frac{b}{2}$
After translation	$0 \leq ax - by + \mu - \frac{b}{2} < b$	$\overset{b=b-a}{\longrightarrow}$	$0 \leq ax - (b-a)y + \mu - \frac{b}{2} < b$

Table 1
Résumé of the different translations from a 8-connected curve to a 4-connected one.

The aim is to reconstruct a Euclidean polyline $(p_1, \ldots, p_j)$ from a 4-connected simple curve given by the ordered set of pixels $P = (P_1, P_2, \ldots, P_n)$. The first point P_1 is the starting point of the discrete curve. A Euclidean point p_1 is chosen inside the pixel P_1 (line 2). Usually the center of the pixel is chosen unless other constraints exist.

Let us suppose that we have already reconstructed the discrete points $(P_1, \ldots, P_i)$ into the Euclidean polyline $(p_1, \ldots, p_k)$. Then the discrete points $(P_i, \ldots, P_n)$ remain (line 4). The Euclidean point p_k is inside the pixel P_i.

A few transformations are needed before the recognition step. Vittone's algorithm works on lines lying in the first octant (for lines $ax - by + \mu = 0$ where $b \geq a \geq 0$). But the serie of 4-connected discrete points can have an arbitrary direction. The general direction (octant) of the discrete point serie is determined using the Chain code of the set of pixels. Indeed, it is well known that the Chain code of a discrete segment is composed of at most two different directions [20]. From those two directions we derive the octant in which the line lies (line 5). A symmetry is then applied in order to transpose the set of pixels into the first octant. Next the shear transform defined in Section 3.3 is applied and transforms the 4-connected set of pixels $(P_i, P_{i+1}, \ldots, P_n)$ into the 8-connected one $\left(P'_i, P'_{i+1}, \ldots, P_n\right)$. The same transformation is also applied to the Euclidean point p_k which results in p'_k (line 6). The Euclidean point $p'_k = (u_k, v_k)$ corresponds to the Euclidean line $d_k : u_k x + y - v_k = 0$ in $\mathcal{P}$ (line 7). Each point of d_k corresponds to a Euclidean line in $\mathcal{C}$ that goes through p'_k.

We can now use Vittone's algorithm. The points P'_{i+j}, $j > 0$ are added one by one in the discrete segment s_k until the domain of s_k does not intersect the Euclidean line d_k anymore (lines 8 to 12). When that happens, if we denote P'_{i+m} the last solution pixel, we know that the discrete points $(P'_i, \ldots, P'_{i+m})$ are recognized as a discrete straight line segment. The intersection between the polygon and the straight line d_k in $\mathcal{P}$ (denoted I in Algorithm 1) corresponds to all the Euclidean lines in $\mathcal{C}$ going through p'_k and containing $(P'_i, \ldots, P'_{i+m})$ in their digitization (see Fig. 4(a) and (b)).

We choose the middle point π'_k of I as the Euclidean reconstruction of the discrete points $(P'_i, \ldots, P'_{i+m})$ (line 17). The point π'_k in $\mathcal{P}$ corresponds to a

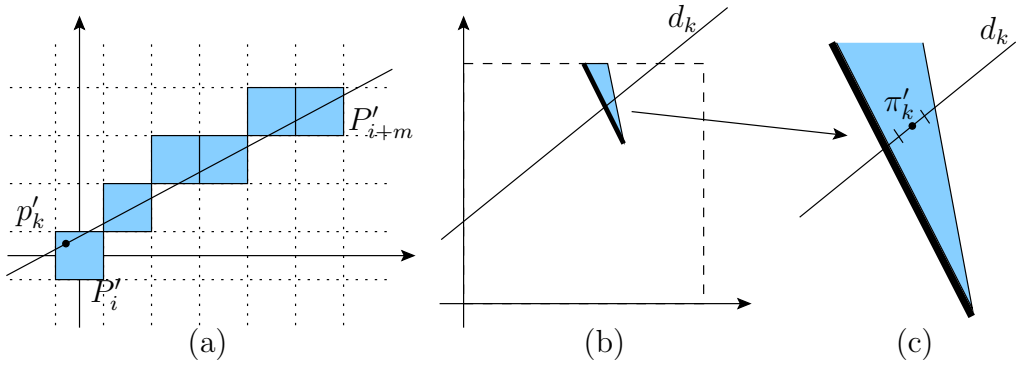


Fig. 4. (a) A set of pixels that has been recognized as a discrete line segment. (b) The domain of the pixels and the Euclidean line corresponding to p'_k in the parameter space. (c) The solution π'_k in the parameter space is a Euclidean line (in (a)) going through p'_k.

Euclidean line that contains p'_k in $\mathcal{C}$ (Fig. 4). The transformations described in Table 1 are applied back so that we get a Euclidean straight line D that passes through p_k and all the pixels $P_i, \ldots, P_{i+m}$ (line 18). The straight line D intersects the pixel P_{i+m} as an interval. The center of this interval is the point p_{k+1} in the reconstruction process (line 19). The recognition steps start then all over with the discrete points $(P_{i+m}, P_{i+m+1}, \ldots, P_n)$ and the Euclidean point p_{k+1} (line 20).

3.3.2 End of the polygonalization process

For a simple curve the recognition process ends simply when we have reached the last discrete point P_n of the curve. A Euclidean vertex is set inside the last pixel as last point of the reconstructed polyline. The matter becomes however a little more difficult when we deal with a closed curve. For a closed curve we have $P_1 = P_n$. It is of course reasonable to expect a close Euclidean polyline as result of a closed discrete curve. This means that the last discrete straight line segment that is reconstructed has to have $p_{k+1} = p_1$. This corresponds to an additional condition that has to be setup is Vittone's algorithm. When we reach the last discrete points of the curve (when $P_n = P_1$ is one of the possible points that will be recognized at this step), we have to ensure that both the Euclidean line d_k (corresponding to p'_k) and the Euclidean line d_1 (corresponding to p'_1) intersect the polygon in the parameter space (see Figure 5). The recognition algorithm progresses as long as the intersection point of both lines belongs to the parameter polygon. If this goes as far as point $P_n = P_1$ then the reconstruction process is finished. If this condition is not verified it means that the Euclidean line $p_k p_1$ does not contain the discrete points $(P_{i+m-1}, \ldots, P_n)$ and we start over with p_{k+1}.

Figure 6 presents a result obtained with this algorithm.

Algorithm 1 Polygonalization of a 2D 4-connected curve

POLYGONALIZATION_2DCURVE(ordered set of pixels P)

$i \leftarrow 1$, $k \leftarrow 1$
Choose a real point p_1 inside P_1.
while $(i \leq n)$ **do**
  $s_k \leftarrow \{P_i\}$
  Find the current octant with Chain codes.
  $\{P_i, \dots P_n\}$ become $\{P'_i, \dots P'_n\}$ and p_k becomes p'_k with the shear transform.
  $I = Dom(s_k) \cap d_k$
  while $(I \neq \emptyset$ and $i \leq n)$ **do**
    i ← i+1
    $s_k \leftarrow s_k \cup \{P'_i\}$
    Compute the reduction of I according to the constraints of P'_i.
  end while
  if $(I = \emptyset)$ **then**
    $s_k \leftarrow s_k - \{P'_i\}$ and retrieve I before the addition of P'_i.
    $i \leftarrow i - 1$
  end if
  Choose a solution π'_k in I.
  Apply the inverse shear transform: π'_k becomes π_k.
  Choose a real point p_{k+1} such that p_{k+1} belongs to the line of parameters π_k and $p_{k+1} \in P_i$.
  $k \leftarrow k + 1$
end while

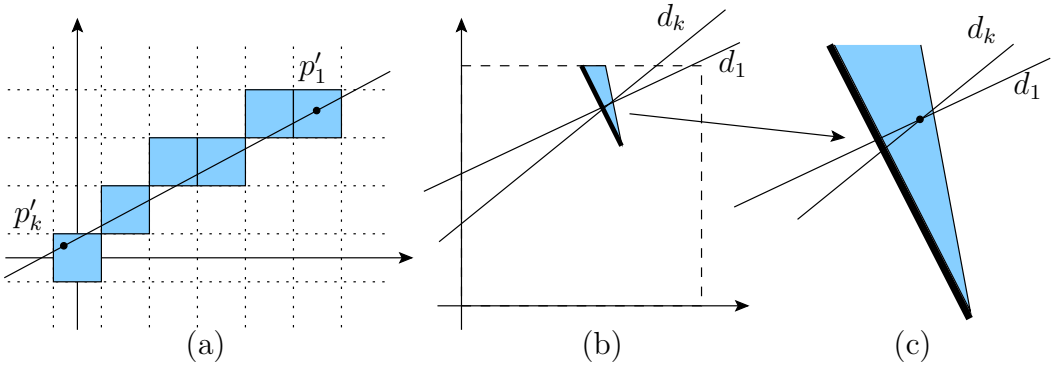


Fig. 5. Same comments as in Figure 4 except that we look for a Euclidean line joining both points p'_k and p'_1. In (b) and (c) we can see how this translates in the parameter space.

4 3D Discrete curve polygonalization

In the previous section, we have presented a new algorithm to polygonalize any 2D discrete 4-connected curve without post-processing treatments in an

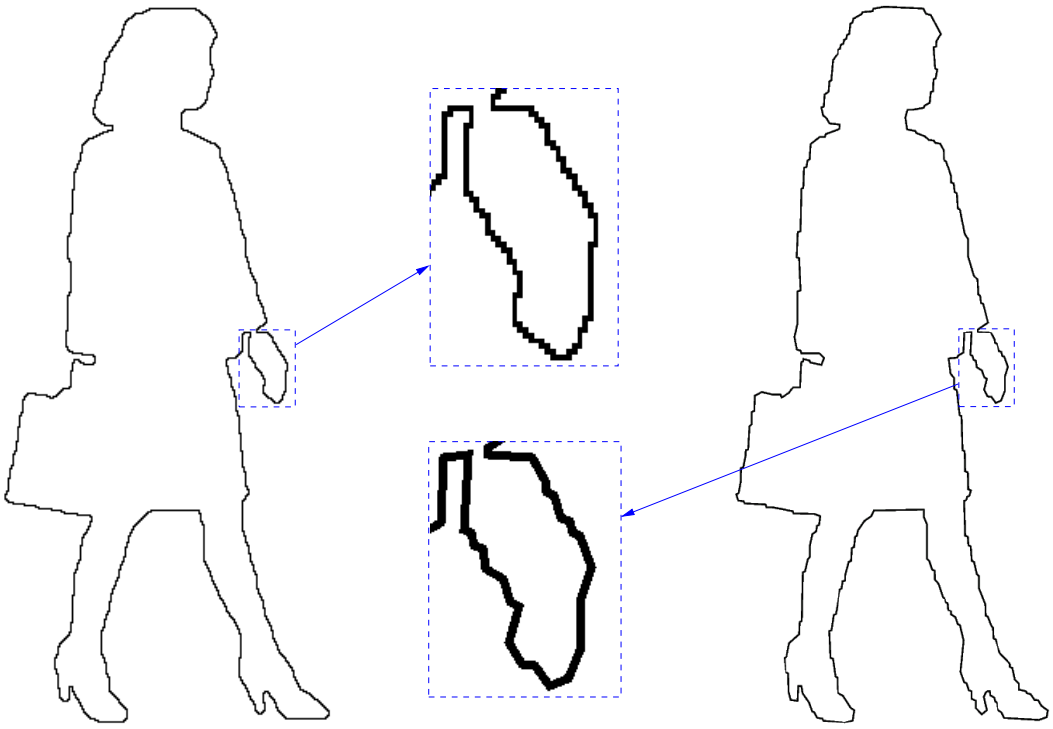


Fig. 6. Result of the polygonalization algorithm: on the left, a discrete curve and a zoom on the woman hand; on the right, the polygonal curve computed by Algorithm 1 and the same hand detail.

invertible way. In this section, we extend this method to 6-connected 3D curves in a three-dimensional cubic grid.

4.1 Principle

The principle of the algorithm is exactly the same as in the 2D case. The reconstruction algorithm works on 3D 6-connected curves $V = (V_1, \ldots, V_n)$. The algorithm starts with a Euclidean point p inside the starting voxel V_1 of the discrete curve. A recognition direction is chosen, and a 3D digital line recognition process is applied: it determines if a set of voxels is a discrete 3D line segment and if there exists a Euclidean line d that passes through p and through all the voxels of this segment. The voxels are added one by one along the recognition direction as long as this holds. Once this condition fails, a new Euclidean point is chosen inside the last but one voxel and the recognition process starts over along the curve until the voxel V_n.

4.2 Parameter Spaces and Domains

The parameter space used in the 2D case has been presented in Section 2, and we present here the parameter spaces (extensions of the 2D parameter space) used in this part.

We denote by $\mathcal{C}$ the usual Cartesian space and by (x, y, z) a point in this space. In 3D, we can define three parameter spaces $\mathcal{P}_x$, $\mathcal{P}_y$ and $\mathcal{P}_z$ and the three polymorphic operators C_x, C_y and C_z which link the parameter spaces with the Cartesian space. Their definitions are given in Table 2. The operators C from $\mathcal{P}$ to $\mathcal{C}$ link one basic geometrical object of $\mathcal{P}$ with its representation in $\mathcal{C}$: a point maps to a plane, a plane maps to a point and finally a line maps to a line.

C_x :	$\mathcal{P}_x$	$\rightarrow$	$\mathcal{C}$
	(α, β, γ)	$\mapsto$	$x + \alpha y + \beta z + \gamma = 0$
	$x + \alpha y + \beta z + \gamma = 0$	$\mapsto$	(x, y, z)

C_y :	$\mathcal{P}_y$	$\rightarrow$	$\mathcal{C}$
	(α, β, γ)	$\mapsto$	$\alpha x + y + \beta z + \gamma = 0$
	$\alpha x + y + \beta z + \gamma = 0$	$\mapsto$	(x, y, z)

C_z :	$\mathcal{P}_z$	$\rightarrow$	$\mathcal{C}$
	(α, β, γ)	$\mapsto$	$\alpha x + \beta y + z + \gamma = 0$
	$\alpha x + \beta y + \gamma + z = 0$	$\mapsto$	(x, y, z)

Table 2
Definitions of the three three-dimensional parameter spaces.

Let us also consider the intersection between those spaces and the planes $\alpha = 0$ and $\beta = 0$. Those intersections define six two-dimensional parameter spaces (Table 3) included in the three three-dimensional parameter spaces. For instance, the two-dimensional parameter space $\mathcal{P}_{xz}$ is equal to $\mathcal{P}_x \cap (\alpha = 0)$. Those spaces can be considered either as restrictions of a 3D parameter space or as two-dimensional parameter spaces. Thus, one point of $\mathcal{P}_{xz}$ can be considered either as a plane perpendicular to the plane $y = 0$ in $\mathcal{C}$ or as a line in the 2D Cartesian space $(0xz)$. An example which illustrates those definitions is depicted in Figure 7.

The preimage or domain of a set of voxels is defined according to those spaces and the standard digitization scheme St presented in Section 2.

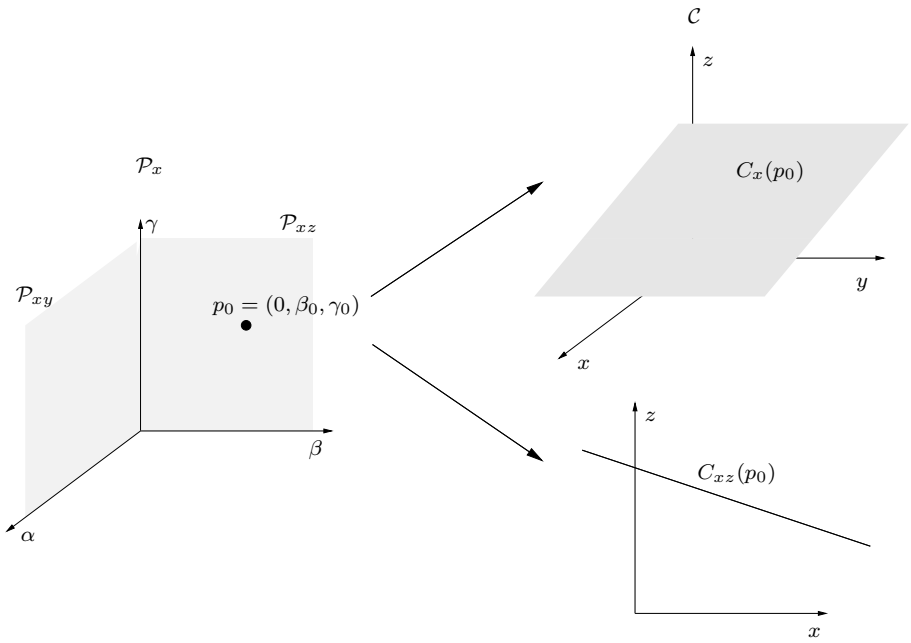


Fig. 7. Illustration of the parameter spaces and operators definitions : the image of the point $p_0 \in \mathcal{P}_x$ (left) by the operator C_x is the plane $x + \beta_0 z + \gamma_0 = 0$. This equation is either the equation of a plane in the 3D space $(0xyz)$ (upper right) or the equation of a line is the 2D space $(0xz)$ (lower right).

$C_{xz} : \mathcal{P}_{xz} \rightarrow \mathcal{C}$ $(\beta, \gamma) \mapsto x + \beta z + \gamma = 0$	$C_{xy} : \mathcal{P}_{xy} \rightarrow \mathcal{C}$ $(\alpha, \gamma) \mapsto x + \alpha y + \gamma = 0$
$C_{yz} : \mathcal{P}_{yz} \rightarrow \mathcal{C}$ $(\beta, \gamma) \mapsto y + \beta z + \gamma = 0$	$C_{yx} : \mathcal{P}_{yx} \rightarrow \mathcal{C}$ $(\alpha, \gamma) \mapsto \alpha x + y + \gamma = 0$
$C_{zy} : \mathcal{P}_{zy} \rightarrow \mathcal{C}$ $(\beta, \gamma) \mapsto \beta y + z + \gamma = 0$	$C_{zx} : \mathcal{P}_{zx} \rightarrow \mathcal{C}$ $(\alpha, \gamma) \mapsto \alpha x + z + \gamma = 0$

Table 3
Definitions of the six two-dimensional parameter spaces in 3D.

Definition 3 (domain) *Consider a set of voxels V. The preimage or domain of V along the x coordinate, denoted $Dom_x(V)$ is:*

$$Dom_x(V) = \{(\alpha, \beta, \gamma) \in \mathcal{P}_x \mid V \subset St(C_x(\alpha, \beta, \gamma))\}$$

where St denotes the standard digitization scheme. $Dom_y(V)$ and $Dom_z(V)$ can be defined in the same way according to $\mathcal{P}_y$ and $\mathcal{P}_z$.

The domain of a given set V is the set of Cartesian planes containing V in their standard digitization. This set of planes can be represented in the three three-dimensional parameter spaces up to a projection operation (see [19]) and

in the following, we use the notation $Dom(V)$ when no particular parameter space needs to be precised or when we deal with the domains of 2D standard lines (see Section 2).

4.3 3D Standard line segment recognition

Let us first recall the definition of a standard 3D line [6,8].

Definition 4 (3D standard line) *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 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 the double inequalities are oriented along the standard convention (see Section 2 and [6,8]).

In order to design a 3D standard line recognition algorithm, we rewrite this definition using the operators of projection in the Cartesian and discrete spaces and the notion of compatible parameters.

Consider a set of voxels V in $\mathbb{Z}^3$ and an Euclidean object $F \subset \mathbb{R}^3$. We define three projection operators denoted by π for the discrete space and three others denoted by p for the Euclidean space as:

$$\begin{aligned} \pi_x(V) &= \{(y,z) \in \mathbb{Z}^2 \mid \exists (x,y,z) \in V\} \quad p_x(V) = \{(y,z) \in \mathbb{R}^2 \mid \exists (x,y,z) \in F\} \\ \pi_y(V) &= \{(x,z) \in \mathbb{Z}^2 \mid \exists (x,y,z) \in V\} \quad p_y(V) = \{(x,z) \in \mathbb{R}^2 \mid \exists (x,y,z) \in F\} \\ \pi_z(V) &= \{(x,y) \in \mathbb{Z}^2 \mid \exists (x,y,z) \in V\} \quad p_z(V) = \{(x,y) \in \mathbb{R}^2 \mid \exists (x,y,z) \in F\} \end{aligned}$$

An illustration of a 3D standard line, a 3D straight line and their projections is given in figure 8.

Definition 5 (compatible) *Consider the three projections $\pi_x(S)$, $\pi_y(S)$ and $\pi_z(S)$ of a 3D discrete segment S. Those projections have compatible parameters if and only if there exists a 3D straight line $\mathcal{L}$ such that for the three coordinates $c \in \{x, y, z\}$, the standard digitization of $p_c(\mathcal{L})$ contains the pixels*

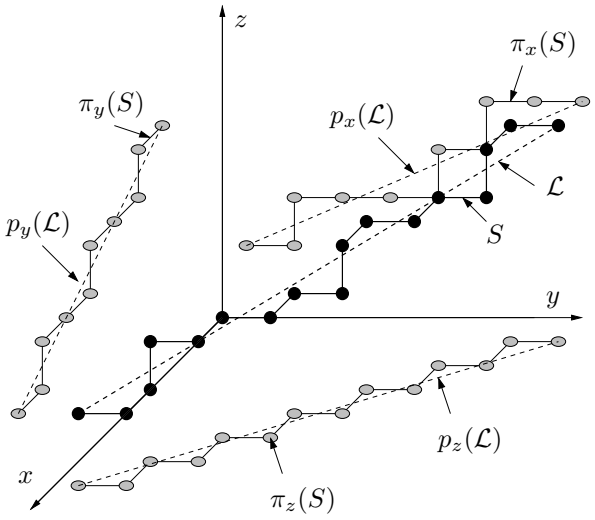


Fig. 8. Illustration of a discrete and a straight line and their projections.

of $\pi_c(S)$. In other words, the parameters of $p_c(\mathcal{L})$ have to belong to the 2D domain (see Section 2) of the set of pixels $\pi_c(S)$.

Hence, a 6-connected 3D discrete curve S is a standard 3D line segment if and only if:

(1) the three projections $\pi_x(S)$, $\pi_y(S)$ and $\pi_z(S)$ are 2D standard line segments
(2) the parameters of those 2D standard line segments are compatible.

As a consequence, the recognition of a 3D standard line is done in two steps: first check that the projections of this line are 2D standard lines, and next, ensure that there exists a solution 3D straight line.

We use the 2D standard line recognition algorithm presented in the previous section to compute the domains related to the three projections of S. Each point of the 3D curve may induce a modification of two out of the three projections.

Each domain of the projections is a convex polygon (see Section 2) that can be represented in two out of the six two-dimensional parameter spaces presented in part 4.2. For instance, the domain of the projection $\pi_z(S)$ can be represented in $\mathcal{P}_{xy}$ and in $\mathcal{P}_{yx}$. Hence, two out of the three projections' domains can be represented in each three-dimensional parameter space. Figure 9 illustrates the embedding of the projections' domains in the parameter spaces for a given set of voxels S.

In order to ensure the compatibility property in the parameter spaces (i.e. during the recognition process), a characterization of the preimage of a 3D Euclidean line is required.

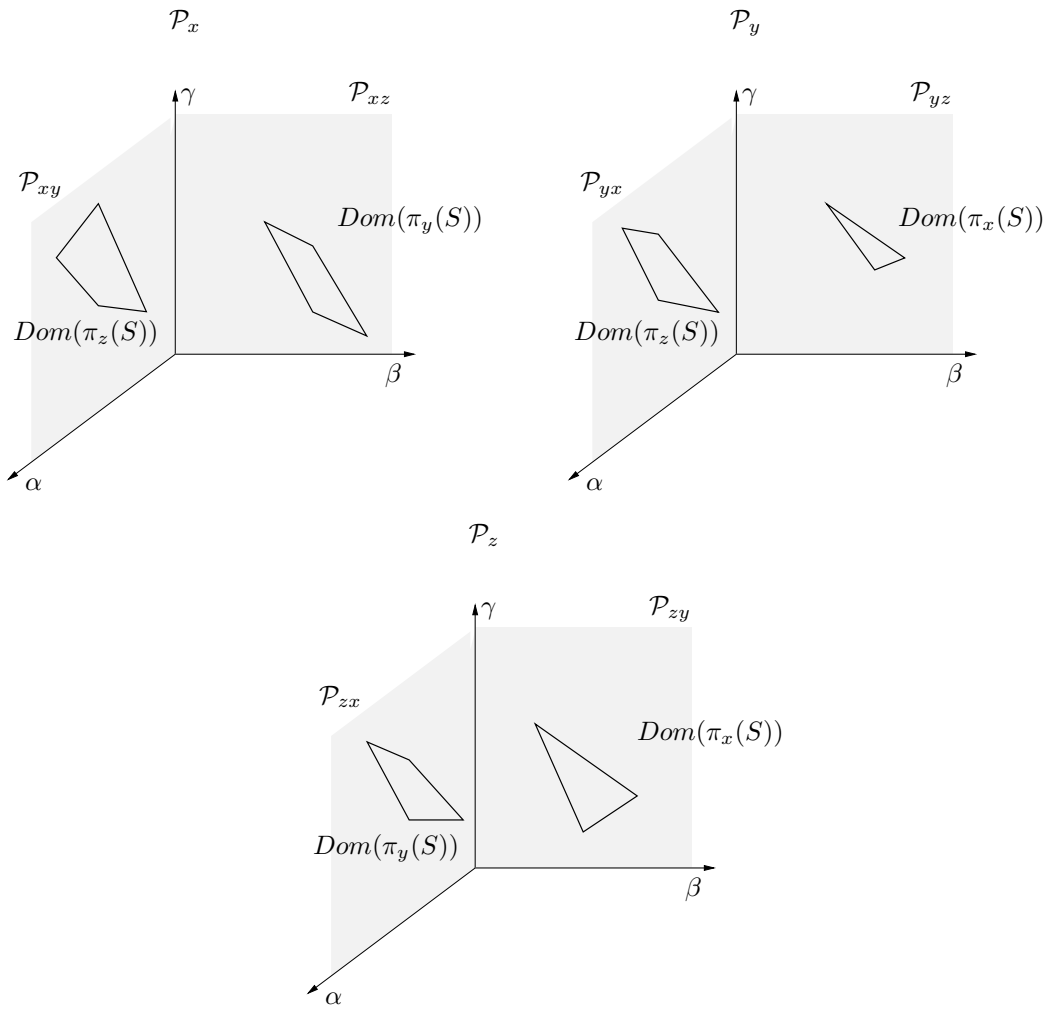


Fig. 9. Representation of the projections' domains of S in the three parameter spaces $\mathcal{P}_x$, $\mathcal{P}_y$ and $\mathcal{P}_z$.

Proposition 6 *Let $L_\mathcal{C}$ be a straight line in the Cartesian space of direction (a, b, c), with $b \neq 0$ and $c \neq 0$, and let $L_\mathcal{P}$ be the image of $L_\mathcal{C}$ in the parameter space $\mathcal{P}_x$ ($L_\mathcal{P} = C^{-1}(L_\mathcal{C})$). Then the point $L_\mathcal{P} \cap (\alpha = 0)$ maps to the line $p_y(L_\mathcal{C})$, and the point $L_\mathcal{P} \cap (\beta = 0)$ maps to the line $p_z(L_\mathcal{C})$.*

PROOF. Let be $L_\mathcal{C}$ the 3D straight line defined by $L_\mathcal{C} = \{(x, y, z) \in \mathbb{R}^3 \mid \exists t \in \mathbb{R},\ (x, y, z) = (x_0, y_0, z_0) + t(a, b, c)\}$, with $b \neq 0$ and $c \neq 0$. Then the projection of $L_\mathcal{C}$ onto the plane $(0xy)$ is the line $p_z(L_\mathcal{C}) :\ bx - ay + ay_0 - bx_0 = 0$. Similarly, its projection onto the plane $(0xz)$ is the line $p_y(L_\mathcal{C}) :\ cx - az + az_0 - cx_0 = 0$. In the parameter space $\mathcal{P}_x$, those lines map to the two points $m_z = (-\frac{a}{b}, 0, \frac{a}{b}y_0 - x_0)$ and $m_y = (0, -\frac{a}{c}, \frac{a}{c}z_0 - x_0)$ respectively. The two equations of $p_z(L_\mathcal{C})$ and $p_y(L_\mathcal{C})$ are also the equation of two planes in which the 3D line $L_\mathcal{C}$ lies. Since the domain of the 3D straight line $L_\mathcal{C}$ is the 3D straight line $L_\mathcal{P}$ representing all the planes containing $L_\mathcal{C}$, the two points m_y and m_z lie on $L_\mathcal{P}$. Finally, we

have $m_y = L_{\mathcal{P}} \cap (\alpha = 0)$ and $m_z = L_{\mathcal{P}} \cap (\beta = 0)$, which ends the proof. □

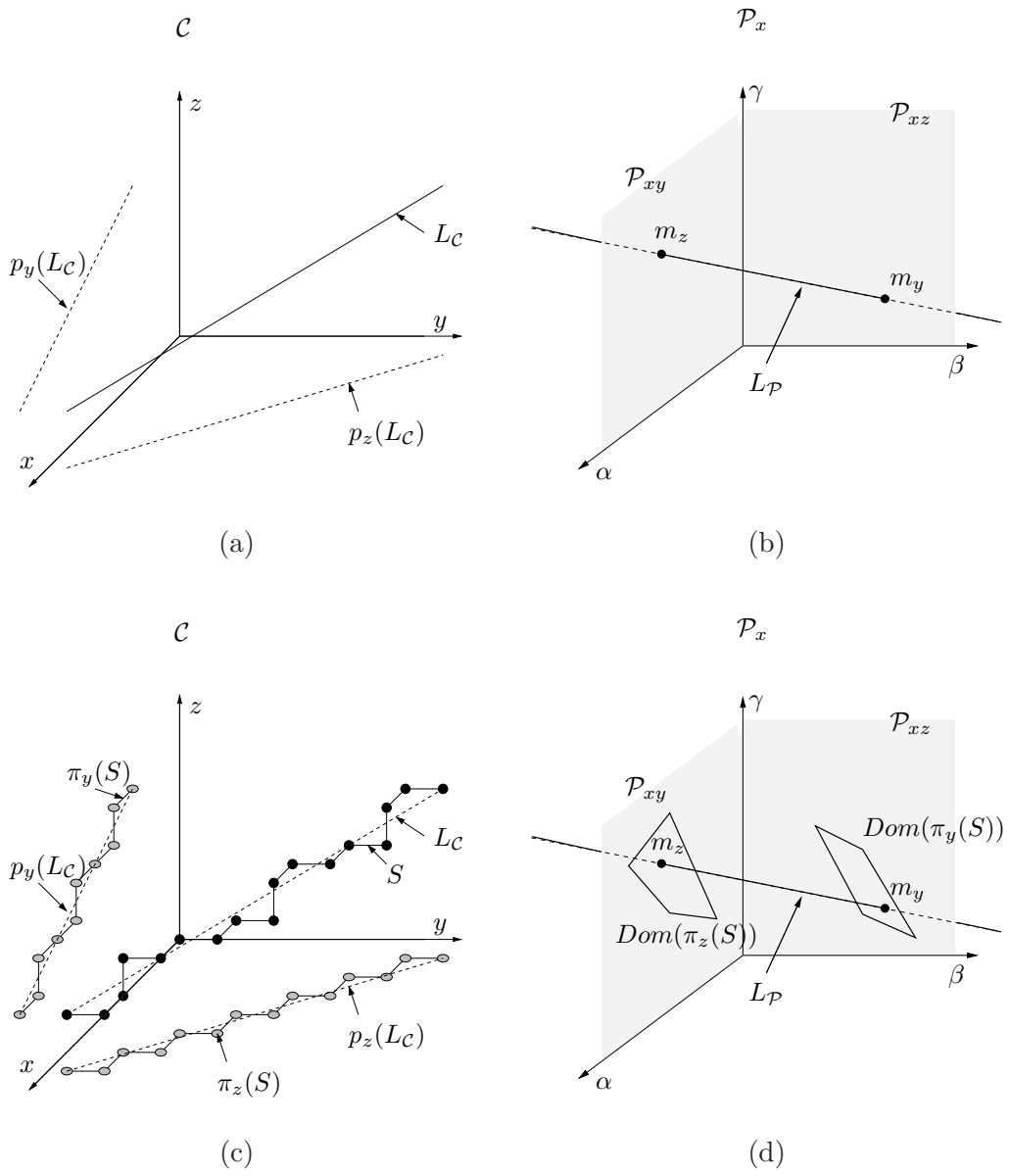


Fig. 10. Illustration of Propositions 6 and 7: (a) a straight line $L_{\mathcal{C}}$ in the Cartesian space and its projections; (b) representation of $L_{\mathcal{C}}$ and its projections in the parameter space; (c) same as (a) plus a discrete segment S, part of $L_{\mathcal{C}}$ standard digitization, and its projections; (d) the points m_y and m_z which represent the projections of $L_{\mathcal{C}}$ belong to $Dom(\pi_y(S))$ and $Dom(\pi_z(S))$ respectively.

This Proposition is illustrated in Figure 10 (a) and (b). We can now give the following result :

Proposition 7 *Let S be a 3D discrete segment. Then any line in the parameter space $\mathcal{P}_x$ that crosses both $Dom(\pi_z(S))$ and $Dom(\pi_y(S))$ represents a*

3D line $\mathcal{L}$ in the Cartesian space such that $p_z(\mathcal{L})$ belongs to $Dom(\pi_z(S))$ and $p_y(\mathcal{L})$ belongs to $Dom(\pi_y(S))$.

The proof of this proposition is straightforward using Proposition 6. Figure 10 (c) and (d) illustrates this result.

Then, according to Proposition 7, any 3D line which crosses the domains of the projections of S in the parameter space maps to a 3D line $L_{\mathcal{C}}$ such that two out of its three projections are solutions for the projections of S. But to ensure the compatibility of the domains, $p_x(L_{\mathcal{C}})$ must also belong to $Dom(\pi_x(S))$. It is not straightforward to check this condition algorithmically in the parameter spaces we have presented by now. Indeed, such a verification implies to check every couple of points in $Dom(\pi_z(S))$ and $Dom(\pi_y(S))$ - this defines a line in the Cartesian space -, then to compute the parameters of the third projection of this line and finally to check whether those parameters belong to $Dom(\pi_x(S))$ or not. If the answer is "yes" for one couple, then the domains are compatible. Otherwise, S is not a 3D standard line segment. This checking has to be exhaustive to give the exact result.

In the next part, we see how the compatibility checking can actually be simplified in a 3D curve segmentation process.

4.4 Polygonalization of a 3D curve

The segmentation process aims at decomposing the 3D discrete curve into 3D discrete segments s_k with corresponding straight line segments r_k of extremal points r_k^1 and r_k^2. As for the 2D case, the main idea is to fix the first extremity of each real segment r_k to a given real point v_k. In the parameter spaces, a point v is represented by a plane as illustrated in Figure 11. Seeing that, for a given segment s_k, all the solution 3D lines have to contain v_k, this condition is transposed in the parameter space saying that we only consider the lines included in $C_x^{-1}(v_k)$. Consequently, the domains are no more polygons but simply segments (see Figure 11).

Consider the two domains $Dom(\pi_z(S))$ and $Dom(\pi_y(S))$. As said previously, any couple of points in those domains defines a Cartesian line $L_{\mathcal{C}}$. To ensure the compatibility, we have to check that there exist two points m_1 and m_2 in the parameter space, the first one in $Dom(\pi_z(S))$ and the second one in $Dom(\pi_y(S))$ such that the third projection of the line $L_{\mathcal{C}}$ defined by m_1 and m_2 lies in $Dom(\pi_x(S))$. The computation of the third projection's parameters from the two points m_1 and m_2 is given by the following property :

Proposition 8 *Let $L_{\mathcal{C}}$ be a Cartesian line, and let $m_1 = (\alpha_1, \beta_1)$ (resp. $m_2 = (\alpha_2, \beta_2)$) be the point in $\mathcal{P}_{xy}$ (resp. $\mathcal{P}_{xz}$) associated to $p_z(L_{\mathcal{C}})$ (resp. $p_y(L_{\mathcal{C}})$).*

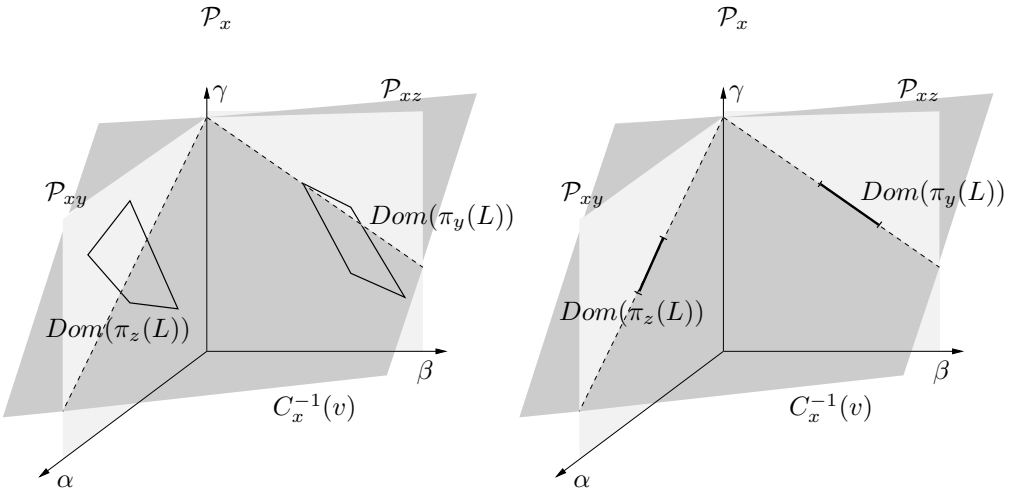


Fig. 11. Representation of what happens in the parameter space when a point v is fixed: on the right, the domains we consider are now straight line segments.

Then, the projection $p_x(L_\mathcal{C})$ maps to the point $(-\frac{\alpha_2}{\alpha_1}, \frac{\beta_1-\beta_2}{\alpha_1})$ in the parameter space $\mathcal{P}_{yz}$.

PROOF. The line $L_\mathcal{C}$ is defined by the two points m_1 and m_2, which correspond to the system of equations $\begin{cases} x + \alpha_1 y + \beta_1 = 0 \\ x + \alpha_2 z + \beta_2 = 0 \end{cases}$. Then we get $\alpha_1 y - \alpha_2 z + \beta_1 - \beta_2 = 0$. If $\alpha_1 \neq 0$, we can divide by α_1, and this equation becomes $y - \frac{\alpha_2}{\alpha_1} z + \frac{\beta_1-\beta_2}{\alpha_1} = 0$, which corresponds to the point $(-\frac{\alpha_2}{\alpha_1}, \frac{\beta_1-\beta_2}{\alpha_1})$ in $\mathcal{P}_{yz}$. Otherwise, if $\alpha_1 = 0$ and $\alpha_2 \neq 0$, we can divide by α_2 and get the same result in the parameter space $\mathcal{P}_{zy}$. Finally, the case $\alpha_1 = 0$ and $\alpha_2 = 0$ is a degenerate case which can be handled algorithmically using infinite domains. □

Since all the domains are embedded in a plane, all the parameters of the third projection computed from any m_1 and m_2 in $\mathcal{P}_{xy}$ and $\mathcal{P}_{xz}$ lie on the same line in $\mathcal{P}_{yz}$. Moreover, this plane is not perpendicular to the $(0\alpha\beta)$ plane. Then, all the domains, which are segments, can be defined by the α-coordinates of their extremities. For instance, $Dom(\pi_z(S))$ which is equal to the segment $[A_z, B_z]$ in $\mathcal{P}_{xy}$ is defined by the interval $I_z = [\alpha_{zA}, \alpha_{zB}]$ where α_{zA} is the α-coordinate of point A_z and α_{zB} is the α-coordinate of point B_z. In the same way, if $Dom(\pi_y(S)) = [A_y, B_y]$, we denote it by the interval $I_y = [\alpha_{yA}, \alpha_{yB}]$ in $\mathcal{P}_{xz}$. Then we have the following property:

Proposition 9 *Consider a discrete segment S and the domains of its projections in $\mathcal{P}_{xy}$, $\mathcal{P}_{xz}$ and $\mathcal{P}_{yz}$ denoted I_z, I_y and I_x. Let f be the function defined as: $f(\alpha, \alpha') = -\frac{\alpha'}{\alpha}$. Then the domains are compatible if and only if $I_x \cap [min(f(\alpha_1, \alpha_2)), max(f(\alpha_1, \alpha_2))] \neq \emptyset$ where $\alpha_1 = \alpha_{zA}$ or $\alpha_1 = \alpha_{zB}$ and*

$\alpha_2 = \alpha_{yA}$ *or* $\alpha_2 = \alpha_{yB}$.

The proof of this proposition is straightforward seeing that the function f, which is part of the function defined in Proposition 8, is continuous and monotonic over positive or negative intervals. If I_z or I_y contains both positive and negative values, then it is simply split into one negative and one positive intervals.

Algorithm 2 describes the global polygonalization algorithm for a 3D curve taking into account all the elements presented so far. This algorithm requires an ordered set of voxels denoted $V = \{V_1, \dots V_n\}$ as input parameter. This curve may be open or closed. The output of this algorithm is a set of ordered discrete segments s_k together with a corresponding real segment r_k which standard digitization is s_k.

Let us give some precisions and explanations on this algorithm.

In the first line, two variables are set: i is used to track all the voxels of the curve and k counts the number of segments found by the algorithm. On line 2, the first extremity of the first real segment is chosen: this real point v_1 has to belong to the voxel V_1.

The "while" loop that begins in line 3 and ends in line 29 is the global tracking of the voxels of the curve. Inside this loop, the discrete segment s_k is initialized with the current voxel V_i (line 4). The extremity of the first real segment r_k is set to the current fixed real point v_k (line 5). The last initializations are those of the domains: as illustrated in Figure 11, those domains are first lines and then segments after the first voxel addition (lines 6 to 8).

While those domains are not empty, the parameters are compatible and the end of the curve is not reached, the following voxel of the curve is added to the current discrete segment s_k (line 12). The projections of the new voxel induce some reductions of the three domains (line 13). Next, the image of the two intervals I_z and I_y is computed and intersected with I_x in order to check the compatibility of the domains (lines 14 to 19). When one condition of the while loop (line 10) is no more fulfilled, if V_i is not the last voxel of the curve, then the voxel V_{i-1} is defined as the last voxel of the segment s_k (line 22). One solution l_k is chosen for the 3D line and the real point v_{k+1} (line 26) is both r_k last point and r_{k+1} first point (lines 25 to 27).

Finally, the recognition process starts all over from line 3 with V_{i-1} as first voxel of the next segment s_{k+1} and v_{k+1} as first extremity of the real segment r_{k+1}.

In the case of closed curves, the same trick as for the 2D case can be used for the end of the recognition process. Indeed, if the discrete curve is closed,

Algorithm 2 Polygonalization of a 3D 6-connected curve

POLYGONALIZATION_3DCURVE(ordered set of voxels V)

1: $i \leftarrow 1$, $k \leftarrow 1$
2: Choose a real point v_1 belonging to V_1.
3: **while** $(i \leq n)$ **do**
4: $s_k \leftarrow \{V_i\}$
5: $r_k^1 \leftarrow v_k$
6: $Dom(\pi_x(s_k)) = \mathcal{P}_{yz} \cap C_y^{-1}(v_k) = I_x$
7: $Dom(\pi_y(s_k)) = \mathcal{P}_{xz} \cap C_x^{-1}(v_k) = I_y$
8: $Dom(\pi_z(s_k)) = \mathcal{P}_{xy} \cap C_x^{-1}(v_k) = I_z$
9: $compatible \leftarrow true$
10: **while** ($I_x \neq \emptyset$ and $I_y \neq \emptyset$ and $I_z \neq \emptyset$ and $compatible = true$ and $i \leq n$) **do**
11: i ← i+1
12: $s_k \leftarrow s_k \cup \{V_i\}$
13: Compute the reductions of the three intervals according to the constraints related to $\pi_x(V_i)$, $\pi_y(V_i)$ and $\pi_z(V_i)$.
14: **if** ($I_x \neq \emptyset$ and $I_y \neq \emptyset$ and $I_z \neq \emptyset$) **then**
15: Compute $f(I_z, I_y)$.
16: **if** ($I_x \cap f(I_z, I_y) = \emptyset$) **then**
17: $compatible \leftarrow false$
18: **end if**
19: **end if**
20: **end while**
21: **if** ($I_x = \emptyset$ or $I_y = \emptyset$ or $I_z = \emptyset$ or $compatible = false$) **then**
22: $s_k \leftarrow s_k - \{V_i\}$ and retrieve I_x, I_y and I_z before the addition of V_i.
23: $i \leftarrow i - 1$
24: **end if**
25: Choose a solution l_k in the domains.
26: Choose a real point v_{k+1} such that $v_{k+1} \in l_k$ and $v_{k+1} \in V_i$.
27: $r_k^2 = v_{k+1}$
28: $k \leftarrow k + 1$
29: **end while**

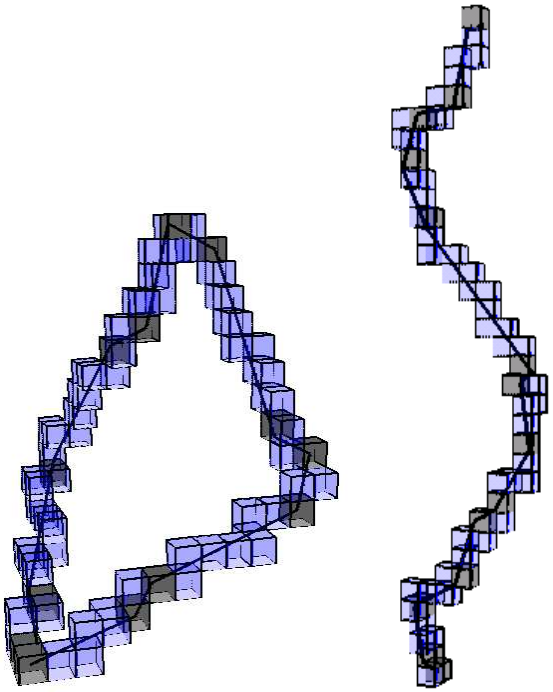
the Euclidean curve reconstructed should also be a closed curve. This can be achieved fixing not one extremity but the two extremities of the maybe last real segment and check if its discretization contains the set of last pixels (see section 3.3.2 for details).

Finally, let us study the complexity of this algorithm. All the operations on lines 2, 4 to 9 and 11 to 12 can be done in constant time. The reductions of the intervals on line 13 are done either in constant time with a simple line/line intersection computation, or in logarithmic time if the same trick as in [18] is used. In practice, this second method is often the fastest mean to compute the intersection of two rational lines. Operations on lines 14 to 19 can also

be done in constant time since $f(I_z, I_y)$ is an interval. In line 25, the choice of the solution is also done in constant time taking the middle point of the solution interval. In line 26, the new real point is computed as the middle of the segment $l_k \cap V_i$ (V_i is the last voxel of the discrete segment) which is done in constant time. At last, the global complexity of this algorithm is $\mathcal{O}(n \log(n))$ where n is the length of the curve.

4.5 Results

Algorithm 2 has been implemented in C using the multi-precision arithmetic library called GMP [21]. Using this library enables to work with rational numbers of any precision without rounding any value. The Figures 12 and 13 present some results on synthetic curves.

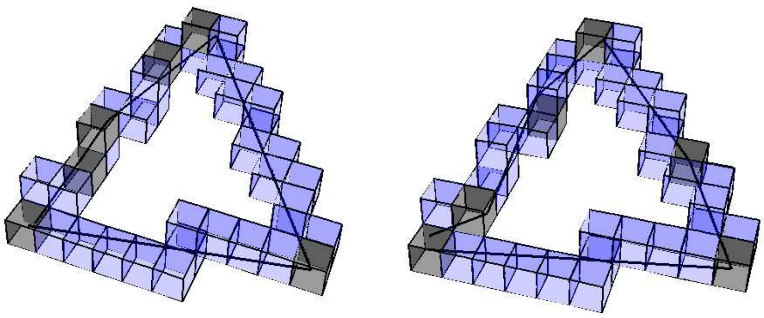


(a) (b)

Fig. 12. Result of the polygonalization algorithm for 3D 6-connected curves.

The algorithm decomposes the curves into discrete and Cartesian segments, one Cartesian segment for one discrete segment. The polygonal line computed is represented by a polygonal dark line, and its digitization is exactly the input discrete curve. The darkest voxels are the extremities of the discrete segments recognized. Remark that each dark voxel contains one extremity of a Cartesian segment.

The curve of Figure 12 (a) and (b) are respectively decomposed into 12 and 13 segments and the curve of Figure 13 (a) is divided into 6 segments.



(a) (b)

Fig. 13. Influence of the fixed real points chosen.

Note that the curve of Figure 13 could theoretically be decomposed into 3 discrete segments only. Indeed, the choice of a fixed real point extremity for each Cartesian segment (used to ensure the reversibility) constraints also the discrete segments. In Figure 13 (a), the first real point chosen is the point of coordinates (x_0, y_0, z_0) which are the coordinates of the first voxel center, whereas in Figure 13 (a), the first real point chosen is the point of coordinates $(x_0 + \frac{1}{4}, y_0 - \frac{1}{4}, z_0 + \frac{1}{4})$. Even if the number of segments computed is the same for the two cases here (it could be different), note that the extremities of the segments are different.

5 Conclusions and future work

In this paper we described a framework to find a polygonal curve from a discrete curve with an invertible method in dimension two and three. In 2D, the method proposed in [7] has been improved and leads to a new algorithm to vectorize a discrete curve without post-treatment patches. We use the Vittone's algorithm for line recognition and force the vertices of the reconstructed polygon to be inside the curve setting one extremity of each real segment to a chosen point before the recognition process. This ensures the reversibility of the reconstruction.

Then, we extended this algorithm for 3D curves. As in 2D, one extremity of each segment of the reconstructed curve is set before the recognition process. This process is composed of three 2D discrete segments recognitions which are done simultaneously. Moreover, a new constraint, called compatibility constraint, ensures that those three recognitions can lead to a solution 3D line.

The results for the polygonalization of 2D curves are good, but we can notice that the number of real segments obtained is greater than what we got with the algorithm presented in [7]. Nevertheless this could be improve by relaxing the constraint of a fix extremity for each real segment while keeping the reversibility property: for instance, we may force each real segment to go through one part of the previous segment which is inside the curve instead of one given point. This should give better results.

In 3D, we presented first results for this problem. We can consider implementing the same improvements as in the 2D case in order to reduce the number of segments found. Moreover, in the context of surface polygonalization, this algorithm may be adapted to polygonalize the border of a discrete plane (coplanar curve). From a discrete surface segmentation into discrete plane pieces, such an algorithm would give an analytical modeling of a discrete surface.

References

[1] W. Lorensen, H. Cline, Marching cubes : a high resolution 3D surface construction algorithm, in: SIGGRAPH '87, Computer Graphics J., Vol. 21, Anaheim, USA, 1987, pp. 163–169.

[2] D. Cœurjolly, Algorithmique et géométrie discrète pour la caractérisation des courbes et des surfaces, Ph.D. thesis, Université Lumière, Lyon 2, France (December 2002).

[3] I. Debled-Rennesson, Étude et reconnaissance des droites et plans discrets, Ph.D. thesis, Universit Louis Pasteur, Strasbourg, France (December 1995).

[4] R. Klette, H. J. Sun, Digital planar segment based polyhedrization for surface area estimation, in: C. Arcelli, L. P. Cordella, G. S. di Baja (Eds.), International Workshop on Visual Form 4, Vol. 2059 of Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 356–366.

[5] J. Vittone, J.-M. Chassery, Recognition of digital naive planes and polyhedrization, in: Discrete Geometry for Computer Imagery, Vol. 1953 of Lecture Notes in Computer Science, Springer-Verlag, 2000, pp. 296–307.

[6] E. Andrès, Discrete linear objects in dimension n: the standard model, Graphical Models 65 (1–3) (2003) 92–111, Special Issue DGCI 2002.

[7] R. Breton, I. Sivignon, F. Dupont, E. Andrès, Towards an invertible euclidean reconstruction of a discrete object, in: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.), Discrete Geometry for Computer Imagery, Vol. 2886 of Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 246–256.

[8] E. Andrès, Defining discrete objects for polygonalization : the standard model, in: A. Braquelaire, J.-O. Lachaud, A. Vialard (Eds.), Discrete Geometry for

Computer Imagery 2002, Vol. 2301 of Lecture Notes in Computer Science, Springer-Verlag, Bordeaux, France, 2002, pp. 313–325.

[9] P. V. C. Hough, Method and means for recognizing complex patterns., United States Patent, n3, 069, 654 (December 1962).

[10] H. Maître, Un panorama de la transformation de Hough, Traitement du Signal 2 (4) (1985) 305–317.

[11] R. O. Duda, P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures., Communications of the ACM 15 (1) (1972) 11–15.

[12] A. Rosenfeld, I. Weiss, A convex polygon is determined by its Hough transform, Pattern Recognition Letters 16 (1995) 305–306.

[13] P. Milanfar, On the Hough transform of a polygon, Pattern Recognition Letters 17 (1996) 209–210.

[14] A. Jonas, N. Kiryati, Digital representation schemes for 3D curves, Pattern Recognition 30 (11) (1997) 1803–1816.

[15] L. Dorst, A. N. M. Smeulders, Discrete representation of straight lines, IEEE Trans. on Pattern Analysis and Machine Intelligence 6 (1984) 450–463.

[16] M. D. McIlroy, A note on discrete representation of lines, AT&T Technical Journal 64 (2) (1985) 481–490.

[17] M. Lindenbaum, A. Bruckstein, On recursive, $\mathcal{O}(n)$ partitioning of a digitized curve into digital straight segments, IEEE Trans. on Pattern Analysis and Machine Intelligence 15 (9) (1993) 949–953.

[18] J. Vittone, Caractérisation et reconnaissance de droites et de plans en géométrie discrète., Ph.D. thesis, Université Joseph Fourier (December 1999).

[19] P. Veelaert, Geometric constructions in the digital plane, Journal of Mathematical Imaging and Vision 11 (1999) 99–118.

[20] H. Freeman, Boundary encoding and processing, Picture Processing and Psychopictorics (1970) 241–266.

[21] GMP library.
URL `http://www.swox.com/gmp/`