Computation of Binary Objects Sides Number using Discrete Geometry, Application to Automatic Pebbles Shape Analysis

Tristan Roussillon,[*] Laure Tougne
Laboratoire LIRIS
Université de Lyon
5 Av Pierre-Mendès France 69676 Bron
{tristan.roussillon, laure.tougne}@liris.cnrs.fr

Isabelle Sivignon
Laboratoire LIRIS
Université de Lyon, CNRS
8 Bd Niels Bohr 69622 Villeurbanne
isabelle.sivignon@liris.cnrs.fr

Abstract

We are working in collaboration with geographers to study a set of pebbles digital images. Among the features geographers want to get automatically, pebble sides number is intuitively defined as shown in Fig 1. It is merely computed after a thick polygonalisation *of pebble contour. We propose an incremental and linear algorithm, free of restrictive hypothesis, which relies on previous works of Debled* et al. *[3, 4, 5]. We give some results on synthetical and real images.*

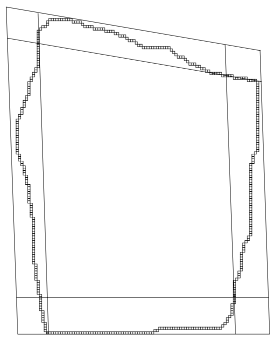


Figure 1. Pebble sides number.

1. Introduction

We are working in collaboration with geographers. Their objective is to establish a set of quantitative characteristics of pebbles sedimented in a river bed. The basic assumption in these studies is that pebbles size and shape are determined by lithology, distance of transport, abrasion, river behavior, etc. The automatic computation of this set of quantitative characteristics by digital image analysis allows a minimization of the time spent on the field and a minimization of the inter- and intra-operator variability. Among the characteristics expected, there is pebble sides number. For geographers, pebble sides number is intuitively defined as shown in Fig 1. We can see that this number results from a *thick polygonalisation* of pebbles contour.

Decomposition of digital curves into piecewise straight segments, splines, quadratic arcs or elliptical arcs is one of the contour-based techniques used in shape analysis, especially for a structural approach. See [9] or [14] for a review of shape representation and description techniques.

Digital straight segments (or discrete segments) are now well known and understood objects. See [8] for a review of the concept of digital straightness. A linear time algorithm for decomposition of digital curves into digital straight segments has been proposed more than ten years ago [6]. Discrete geometry provides effective algorithms, which deal with integers, and permits perfect representations or reconstructions of discrete curves. However, the number of digital straight segments obtained is too large for irregular curves. To cope with this problem, Debled-Rennesson *et al.* [3, 4, 5] have introduced the concept of blurred segment as an extension of the arithmetical approach of Reveilles [12] and derived a polygonalisation algorithm linear in time. Another concept, called α-thick digital line arc, and the polygonalisation algorithm which results from this concept [1], are very similar to Debled *et al.*'s ones. Their algorithms seem to be well adapted, because they produce an edge-based polygonalisation (the goal is to find a sequence of sets of 8-connected points that may stand for the edges), contrary to most of the classical methods which are vertex-based ones (the goal is to find a subset of *dominant points* as stated in [13]). Moreover classical methods that are available in C libraries, like Teh and Chin detection of dominant points or Hough Transform, were used

[*]Supported by a grant from the french DGA (Jacques Blanc-Talon).

for our application and gave more sides than expected.

However the algorithms presented in [1] and [3] assume that the points are added with increasing x-coordinate or y-coordinate. This hypothesis is restrictive if one wants to deal with very noisy curves or if one wants to capture geometry at a coarse scale. In the present paper, we show that we can remove this hypothesis, while keeping the linearity of the algorithm, and as a consequence, deal with any noisy digital curves, with any maximal acceptable width.

The paper is organised as follows. In section 2, we recall some definitions and describe the algorithm presented in [3]. In section 3, we describe and prove our algorithm. Assessments and experiments are given in section 4 with synthetical and real images. The paper ends with some conclusions and future works in section 5.

2. Blurred Segments

2.1. Definitions

The notions of blurred segments and strictly bounding line rely on the arithmetical definition of discrete lines [12].

A set Σ of integer points belongs to the digital straight line $\mathcal{D}(a, b, \mu, \omega)$ with slope $\frac{a}{b}$, lower bound μ and thickness ω (with a, b, μ and ω being integer such that $gcd(a, b) = 1$) if and only if all integer points (x, y) of Σ verify the Diophantine inequalities : $\mu \leq ax - by < \mu + \omega$. The real lines $ax - by = \mu + \omega - 1$ and $ax - by = \mu$ are, respectively, called the *upper* and *lower leaning lines* of $\mathcal{D}(a, b, \mu, \omega)$ [6]. The integer points of the leaning lines are called *leaning points* [6]. If $\omega < max(|a|, |b|)$, the digital straight line is disconnected. If $\omega = max(|a|, |b|)$, the digital straight line is 8-connected (it is named *naive line*). Increasing ω makes the digital straigth line thicker [12]. See Fig. 2 for an example. The digital straight line is depicted with black disks. It is a set of integer points located between two leaning lines (straight lines) passing through some leaning points (circles).

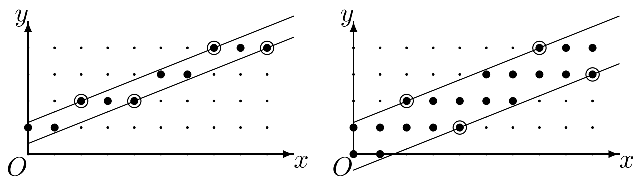


Figure 2. The naive line $\mathcal{D}(2, 5, 0, 5)$ (left) and the thick line $\mathcal{D}(2, 5, 0, 10)$ (right) with their leaning points and leaning lines.

In the following, as a simplification, we assume that $0 \leq y \leq x$. This hypothesis can be done without loss of generality due to symmetries with respect to Ox, Oy and the real line $x = y$.

Definition 1 *[5] Let us consider a set of 8-connected points $\mathcal{S}$. A discrete line $\mathcal{D}(a, b, \mu, \omega)$ is said* bounding, *if all points of $\mathcal{S}$ belong to $\mathcal{D}$.*

Definition 2 *[4] A bounding line of $\mathcal{S}$ is said* optimal *if its vertical distance is minimal, i.e. if the vertical distance of $\mathcal{S}$ equals the vertical distance of its convex hull denoted by $conv(\mathcal{S})$.*

Definition 3 *[4] A set $\mathcal{S}$ is a* blurred segment of width ν *if and only if its optimal bounding line has a vertical distance less than or equal to ν.*

See Fig. 3 for an example. The blurred segment $\mathcal{S}$ is depicted with black disks. The convex hull $conv(\mathcal{S})$ of $\mathcal{S}$ is depicted with dotted lines. The vertical width of $conv(\mathcal{S})$ (double arrow) is computed from the leaning points (circles) and leaning lines (straight lines). The optimal bounding line (which is a thick digital straight line) is drawn with squares on the right.

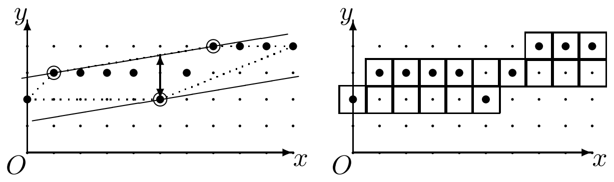


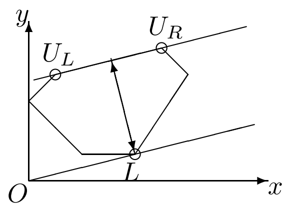
Figure 3. The optimal bounding line $\mathcal{D}(1, 6, 0, 11)$ (squares) of a blurred segment $\mathcal{S}$ (black disks) of width $\nu = 2$.

The recognition of blurred segments with width ν is thus equivalent to the computation of the vertical (or horizontal) width of the convex hull $conv(\mathcal{S})$ [4]. Because of the incremental property of the algorithm, the polygonalisation is very simple: add points until the vertical (or horizontal) width computed is strictly higher than the threshold value ν. The current blurred segment ends and a new blurred segment begins.

It should be noticed that due to symmetry, we only have to know how to compute the vertical width of the convex hull for maintaining both vertical and horizontal width simultaneously and always choose the minimal value as the width of the convex hull [4]. Thus, in the following, as a simplification, we focus on the computation of the vertical width.

It should be noticed that this definition of width is different from the one usually used in Computational Geometry, which is generally not computed with respect to Ox or

Oy [1] (see Fig. 4). However, basic ideas used to compute this width are really close to the ones used in Computational Geometry for the rotating calipers algorithm [11].



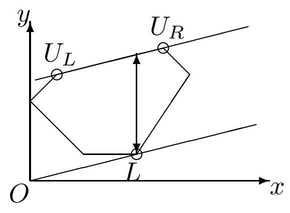


Figure 4. The common (left) and vertical (right) width computed from U_L, U_R and L.

The vertical width of a convex polygon $\mathcal{P}$ is the maximal length of the intersection of $\mathcal{P}$ with a vertical line. The width function $width(i)$ is the length of the intersection of $\mathcal{P}$ with a vertical line given by $x = i$. It is clear that the width function is concave (see Fig. 5).

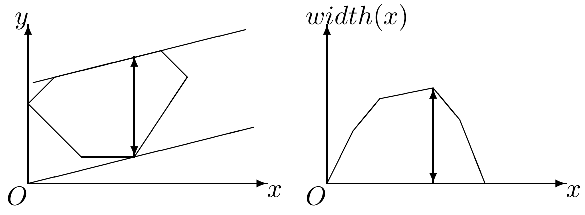


Figure 5. Vertical width and width function.

2.2. Restriction

A linear time incremental algorithm of segmentation into blurred segments with width ν is given in [4]. For each added point, there are two steps.

The first step consists in maintaining the convex hull $conv(\mathcal{S})$ of $\mathcal{S}$ with Melkman's algorithm [10]. This is an incremental and linear algorithm which assumes that added points form a simple polyline (which does not intersect itself). For us, this hypothesis is not restrictive since we deal with binary objects boundaries.

The second step consists in computing the vertical width of the convex hull $conv(\mathcal{S})$. As written in [4], we can consider three cases (edge, vertex), (vertex, edge) and (vertex, vertex) where the first element is located on the lower part of the convex hull and the second element is located on the upper part (see Fig. 4). We call U_L and U_R the left and right extremities of the edge and L the vertex because the two first cases are symmetrics. The third case is comparable with a case (edge-vertex) or (vertex-edge) where the edge is reduced to a single vertex. The edge and its parallel passing through the vertex (called *supporting lines* in Computational Geometry [11]) define exactly the lower and upper leaning lines of an optimal bounding line for $\mathcal{S}$ [4]. Thus, from U_L, U_R and L, we can compute the vertical width of $\mathcal{S}$. Comparing this value to ν, we can then decide if $\mathcal{S}$ is a blurred segment of width ν or not.

The key point consists in updating the position of the leaning points. The algorithm described in [3] is incremental and linear, but it is unusable for a computation of the pebbles sides number because it assumes that the points are added with increasing x-coordinate or y-coordinate. The more the digital curve is noisy and the more ν (the maximal acceptable width) is high, the less this hypothesis is fulfilled. Hence, we obtain too much sides by running this algorithm. In Fig. 6 (which is drawn from figure 7 in [4]), the points are neither added with increasing x-coordinate, nor with increasing y-coordinate. As a consequence, the blurred segment of width $\nu = 3.9$ (the extremities of which are the grey squares) is shorter than expected.

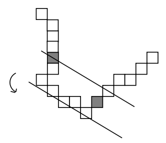


Figure 6. The problem of the algorithm in [4].

3. Our algorithm

3.1. Description

Let us consider that we add a new point M to $\mathcal{S}$, $\mathcal{S}' = \mathcal{S} \cup M$.

At a first level, there are three cases (see Fig. 7).

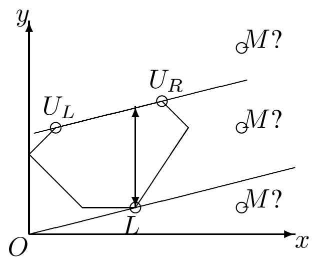


Figure 7. The three cases while adding M.

If M belongs to $\mathcal{D}$, the position of U_L, U_R and L do not change after the application of Melkman's algorithm. So $\mathcal{S}'$ remains a blurred segment with the same vertical width and with the same optimal bounding line $\mathcal{D}$. In the other cases, M is above or below $\mathcal{D}$, the position of U_L, U_R and L change and the vertical width has to be recomputed.

At a second level, for both cases where M does not belong to $\mathcal{D}$, there are several cases again.

Let us called ${U_L}'$, ${U_R}'$ and L', the new leaning points after the update. Given a point P of the convex hull $conv(\mathcal{S})$, we denote $next(P)$ (resp. $prev(P)$) the functions which return the point after (resp. before) P in a trigonometric orientation.

- First, suppose that M is above $\mathcal{D}$.

  Let us suppose that M **is after** L (i.e. the x-coordinate of M is strictly higher than the x-coordinate of L). See case a) in Fig. 8.

  We start by setting ${U_L}'$ and ${U_R}'$ to the two convex hull points that are neighbors of M. In Fig. 8, case a), ${U_L}'$ and ${U_R}'$ are, respectively set to U_L and U_R.

  As M is above $\mathcal{D}$, the slope of $({U_L}'M)$ is greater than the slope of (U_LU_R). As a consequence, the width function increases from ${U_L}'$ to L.

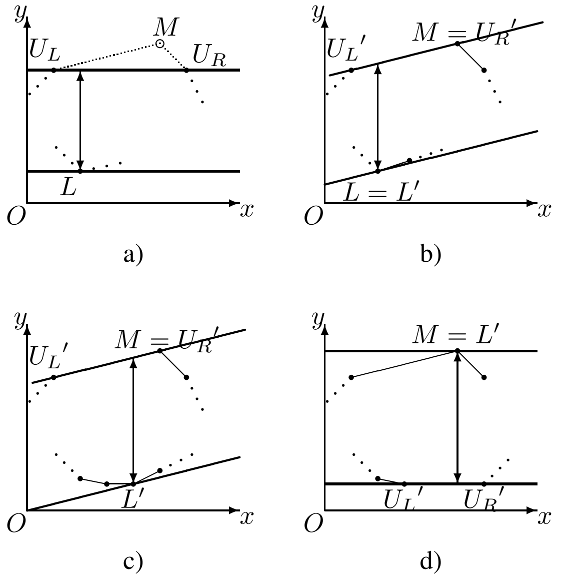


Figure 8. What may happen if M is above (U_LU_R) and after L.

  If the slope of the edge $[L\ next(L)]$ is greater than the slope of $({U_L}'M)$, the width function decreases from L to M (see case b) in Fig. 8). ${U_R}'$ is set to M and L' is set to L. We can compute the vertical width from ${U_L}'$, ${U_R}'$ and L'.

  If the slope of the edge $[L\ next(L)]$ is lower than the slope of $({U_L}'M)$ and if $next(L)$ is before M, the width function increases from L to $next(L)$. In this case, L moves along the lower part of the convex hull. As a matter of fact, L is moving until $next'(L)$ such that $slope(next(L), next'(L)) > slope({U_L}', M)$.

  – If $next'(L)$ is before M, then ${U_R}' = M$ and $L' = next'(L)$ (see case c) in Fig. 8).

  – If $next'(L)$ is after M, then ${U_L}' = next(L)$, $U_R = next'(L)$ and $L' = M$ (see case d) in Fig. 8). The configuration is reversed because the vertical width is given by an edge from the lower part of the convex hull and by a vertex from the upper part of the convex hull.

  The case where M **is before** L is symmetric to the previous case.

  The particular case where M and L have a same x-coordinate is straightforward. Both U_L and U_R move to M. The vertical width is the length of $[U_LL]$ or $[U_RL]$. The slope of $\mathcal{D}$ is equal to zero.

- Next, suppose that M is below $\mathcal{D}$.

  The case where M **is after** U_R is identical to the case where M is above $\mathcal{D}$ and after L, but the configuration is reversed. It is enough to exchange the leaning points to have M above $\mathcal{D}$ and after L.

  The case where M **is before** U_L is symmetric to the previous case, i.e. comparable with the case where M is above $\mathcal{D}$ and after L.

  The particular case where M is between U_L and U_R is straightforward. As the width function increases until M and decreases from M, we set L' to M, ${U_L}'$ to U_L and ${U_R}'$ to U_R.

Given a digital curve which does not intersect itself, there is an incremental algorithm which maintains the convex hull of the digital curve thanks to the Melkman's algorithm and which computes the vertical width of the convex hull thanks to the algorithm above.

3.2. Complexity

To study the algorithm complexity, the idea is to study leaning point displacements. It is enough to focus on the lower leaning point because the upper leaning points are the extremities of the edge on which L project vertically. The algorithm described in the previous section has two interesting properties about the lower leaning point straightforwardly deduced from the different cases studied in section 3.1.

Property 1 *For each added point M, the lower leaning point does not move or moves towards M.*

Property 2 *For each added point M, the lower leaning point never goes beyond M.*

M moves along the digital curve and L always moves towards M without going beyond it. Thus, in case of connected points as input (what is not a restrictive hypothesis for us), L moves are bounded by M moves, i.e. the update of L for the insertion of n connected points do not require more than the test of n points. We can conclude that the overall complexity of the algorithm is $\mathcal{O}(n)$ in that case.

4. Experiments

We implemented our algorithm. It takes as input, chain code, extracted from objects boundary by contour tracking. The starting point is first an extremal point (i.e. with minimal x-coordinate and y-coordinate). We shift then the starting point to the last point of the first blurred segment recognized in order to avoid over-segmentation. In addition to the chain code, our algorithm takes as input, ν, the maximal acceptable width. We studied the behavior of our algorithm both with synthetical images and real images.

4.1. Synthetical images

We drew 48 binary objects with a raster graphics editor. We first drew 27 objects with the *polygon function*. These objects were (convex or concave) polygons (see a) in Fig. 9). Then, we sketched 21 objects with the *pencil function* while trying to imitate the previously drawn polygons. As a result, these objects look like noisy polygons (see b) in Fig. 9). The points number of the set of objects is ranging from 371 to 2870 and the sides number of the set of objects is ranging from 3 to 26.

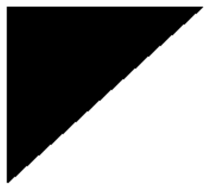
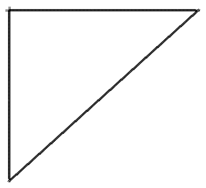
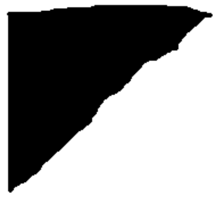
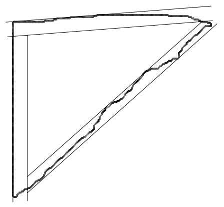
	Binary objects	Thick polygons	ν^*
a)			1.5
b)			13.7

Figure 9. Results for two triangles.

For each object, we performed polygonalisations with width ν ranging from 0.5 to 15, with a step of 0.1. Objects sides number was deduced from the polygonalisations by counting the blurred segments obtained. We focused on the minimal width denoted by ν^* from which we found the true sides number (see Fig. 9). Binary objects drawn with the *polygon function*, called *standard objects*, are used as ground truth.

For standard objects, the minimal width average is 1.87 (ν^* ranging from 1 to 2.7), while for noisy objects, the minimal width average is 7.77 (ν^* ranging from 4.7 to 13.7). See Tab. 1.

objects	min	max	average
standard objects	1	2.7	1.87
noisy objects	4.7	13.7	7.77

Table 1. The minimal width ν^* from which we found the true sides number for each object.

As object sides number is rather stable with respect to the increase of ν beyond ν^*, we reach a perfect prediction rate (i.e. 100%) with $\nu = 3$ for standard objects and with $\nu = 14$ for noisy objects. However, there is no value, which allows to reach a perfect prediction rate for both standard and noisy objects. With $\nu = 14$, we found one side less than expected for 4 standard objects (the predection rate is 44/48). Indeed, even if object sides number is rather stable with respect to the increase of ν, it decreases and tends to 1 for sufficiently large ν (one optimal bounding line $\mathcal{D}$ contains $\mathcal{S}$).

4.2. Real images

We used our algorithm in order to study pebbles digital images. Twenty samples of pebbles taken in the bed of an Indonesian river at various distances from the sea were photographed. We have approximately 1500 pebbles contours of about 350 pixels to analyze.

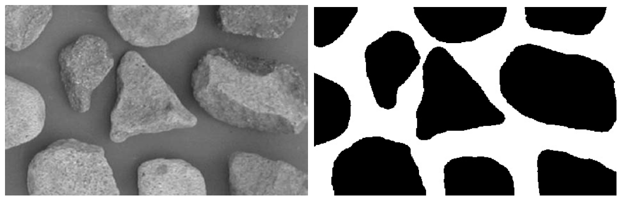


Figure 10. Pebbles images. The 3-sides pebble located in the center of the images was processed in Fig 11.

In a first step we detected pebbles with clustering methods, transforming the original color image into a binary image as shown in Fig. 10. In a second step, we extracted chain codes from pebbles boundaries. In a third step we computed the number of sides. The input parameter ν was set up at 13 by trial and error.

Pebbles size variation is very small in both real world and images. Acquisition process and digitization are identical for all images. According to previous study with synthetical images, it seems that ν^* depends on size and noise. It was thus relevant to set up the value of ν, once for all. Statistical and geographical analysis are under progress. An exemple of polygonalisation is shown in Fig. 11 and some pebbles with their sides number are gathered in Fig. 12.

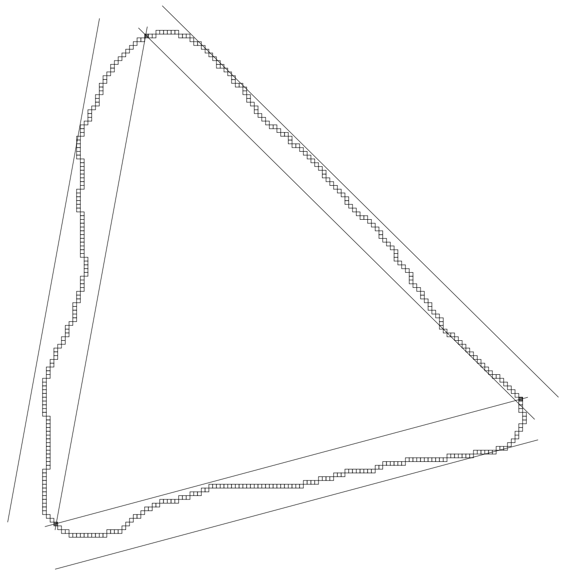


Figure 11. Pebble polygonalisation with $\nu = 13$. The number of sides found is 3.

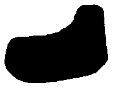
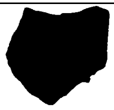
				
4	5	6	7	8

Figure 12. Some pebbles sides number.

5. Conclusion and perspectives

The objective was to compute the number of sides that pebbles have. We opted for an edge-based polygonalisation which is exactly the geographers' intuitive definition (see Fig. 1 and Fig. 11). We proposed an incremental and linear algorithm, free of restrictive hypothesis, contrary to what we can read in the litterature [1, 3, 4, 5]. We assessed our algorithm with synthetical images. It turns out that the input parameter ν controls the polygonalisation according to noise, the objects size and the study scale. We experimented our algorithm with real pebbles images too. Pebble sides number seems to be an interesting feature.

For the future, we think about fixing the input parameter in an automatic way (once for all) or in a dynamic way (tuning it during the process). We need a framework for assessing thick polygonalisations which are different of thoroughly studied polygonal approximations [13]. To end the paper, an interesting way of investigation is the extension to the concept of *blurred tangent* in order to compute robust curvature estimators (see [2] for a first attempt) or to determine the minimal number of necessary blurred segments for a thick polygonalisation (like the min-DSS problem [7]).

References

[1] L. Buzer. An elementary algorithm for digital line recognition in the general case. In *Discrete Geometry in Computer Imagery*, pages 299–310, 2005.

[2] I. Debled-Rennesson. Estimation of tangents to a noisy discrete curve. In *Vision Geometry XII*, pages 117–126, 2004.

[3] I. Debled-Rennesson, F. Feschet, and J. Rouyer-Degli. Optimal blurred segments decomposition in linear time. In *Discrete Geometry in Computer Imagery*, pages 371–382, 2005.

[4] I. Debled-Rennesson, F. Feschet, and J. Rouyer-Degli. Optimal blurred segments decomposition of noisy shapes in linear time. *Computers and Graphics*, 30:30–36, 2006.

[5] I. Debled-Rennesson, J.-L. Rémy, and J. Rouyer-Degli. Linear segmentation of discrete curves into blurred segments. *Discrete Applied Mathematics*, 151:122–137, 2005.

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

[7] F. Feschet and L. Tougne. On the min dss problem of closed discrete curves. *Discrete Applied Mathematics*, 151:138–153, 2005.

[8] R. Klette and A. Rosenfeld. Digital straitghness – a review. *Discrete Applied Mathematics*, 139:197–230, 2004.

[9] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[10] A. A. Melkman. On-line construction of the convex hull of simple polygon. *Information Processing Letters*, 25:11–12, 1987.

[11] F. P. Preparata and M. I. Shamos. *Computational geometry : an introduction*. Springer, 1985.

[12] J.-P. Reveillès. *Géométrie Discrète, calculs en nombres entiers et algorithmique*. thèse d'etat, Université Louis Pasteur, 1991.

[13] P. L. Rosin. Techniques for assessing polygonal approximation of curves. *IEEE Transactions on PAMI*, 19(6):983–1001, 1997.

[14] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, 2004.