Graphical Models 71 (2009) 113–124

Contents lists available at ScienceDirect

Graphical Models

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

A framework for dynamic implicit curve approximation by an irregular discrete approach

Antoine Vacavant [a,*], David Coeurjolly [b], Laure Tougne [a]

[a] Université de Lyon, CNRS, Université Lyon 2, LIRIS, UMR5205, F-69676, France
[b] Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69622, France

ARTICLE INFO

Article history:
Received 2 July 2007
Received in revised form 8 December 2008
Accepted 28 February 2009
Available online 14 March 2009

Keywords:
Implicit curve approximation
Interval arithmetic
Irregular grid
Discrete geometry

ABSTRACT

The approximation of implicit planar curves by line segments is a very classical problem. Many algorithms use interval analysis to approximate this curve, and to handle the topology of the final reconstruction. In this article, we use discrete geometry tools to build an original geometrical and topological representation of the implicit curve. The polygonal approximation contains few segments, and the Reeb graph permits to sum up efficiently the shape and the topology of the curve. Furthermore, we propose two algorithms to process local cells refinement and local cells grouping schemes. We illustrate these schemes with a global system that efficiently handles manual or automatic fast updates on the global reconstruction, by considering topological or geometrical constraints. We also compare the speed and the quality of our approach with two classical methods.

1. Introduction

In Computer Graphics, the polygonal approximation of an implicit planar curve is a classical problem. Indeed, since such a curve is a very complex object, it becomes hard to plot it, or to apply non-analytical operations on its interior, like boolean operators (union, intersection, *etc.*). The approximation by line segments allows to quickly carry out those tasks. In this case, the problem is to compute a polygonal object $\mathscr{L}$ approximating the curve $\mathscr{C}$, given by the solutions of its associated function f:

$$\mathscr{C} = \{(x,y) \in \Omega : f(x,y) = 0\} \text{ such that } f : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R} \quad (1)$$

To handle this polygonal approximation problem, many tracing algorithms [18,23,24,29,36] use *interval arithmetic* (or *interval analysis*) to approximate $\mathscr{C}$ with a set $\mathscr{E}$ of two-dimensional (2-D) rectangles (or *intervals* for the sake of clarity, instead of *hyper-intervals*). Interval arithmetic, introduced by Moore in the 1960–1970s [20,21], defines classical arithmetic operators and more complex functions by considering intervals containing the real (unknown) value computed. Those intervals are framed by two numbers representable in floating-point. This principle permits to avoid errors in floating-point arithmetic, by framing real quantities we manipulate with intervals. Then, by combining redefined operators and mathematical functions, an *inclusion function* associated with f can be defined. With such a function, we can test if a 2-D interval intersects or not the planar curve $\mathscr{C}$. In general, to compute the set of intervals $\mathscr{E}$, tracing algorithms recursively subdivide an initial large interval I into intervals that intersect $\mathscr{C}$. This loop stops when all the resulting intervals are small enough to approximate at best. Moreover, this refining scheme can be topologically controlled. For example, Snyder [29] proposes to test if an interval is *globally parameterizable* to correctly plot implicit curves and surfaces. Plantinga and Vegter [24] choose to maintain a balanced quadtree [27] and a *small normal variation* criterion (see also [3] for a global overview). Finally, the polygonal approximation $\mathscr{L}$ of $\mathscr{C}$

* Corresponding author.
E-mail addresses: antoine.vacavant@liris.cnrs.fr (A. Vacavant), david.coeurjolly@liris.cnrs.fr (D. Coeurjolly), laure.tougne@liris.cnrs.fr (L. Tougne).
URLs: http://liris.cnrs.fr/antoine.vacavant (A. Vacavant), http://liris.cnrs.fr/david.coeurjolly (D. Coeurjolly), http://liris.cnrs.fr/laure.tougne (L. Tougne).

1524-0703/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2009.02.001

is performed by tracing one or more segments in each interval I, by computing the points where $\mathscr{C}$ intersects the boundaries of the rectangle associated to I.

In this article, we propose a system to build the polygonal approximation and to represent the topology of $\mathscr{C}$ thanks to discrete geometry tools. More precisely, we handle the description of $\mathscr{C}$ within an irregular partition of the plane, or *irregular isothetic grid* ($\mathbb{I}$-grid for short) [6]. In our framework, the *cells* of this grid are defined by variable sizes and positions. We also propose a method to update the polygonal and the topological interpretations of $\mathscr{C}$ with local refinement and local grouping schemes. Thanks to the three algorithms we present here, we could implement a two-pass algorithm that first refines intervals while a given geometrical or topological criterion is not respected, then groups them. We can draw a parallel between this approach and the split-and-merge image segmentation algorithm [15], where the image is subdivided into small regions, that are then grouped together if they are homogeneous enough.

We first give details about interval arithmetic and link it with discrete geometry by recalling some definitions and introducing the concept of irregular discrete arcs. Then, we present the extended supercover model on an $\mathbb{I}$-grid. We also recall the invertible reconstruction of arcs into line segments described in [7]. In Section 3, we describe our topological and geometrical reconstruction of an implicit curve based on the Reeb graph [26] that we have proposed in [32]. Then, we present in the next section two algorithms to handle local refinement and grouping schemes. We also give an example of application of our system for manual and interactive approximations of an implicit curve by lines. We finally compare the speed and the quality of our approach with two relevant methods [18,29], before discussing about possible extensions of our system.

2. Preliminaries

2.1. Interval arithmetic

Interval arithmetic (or interval analysis) has been introduced by R.E. Moore in the 1960s [20,21] (see also [9,16,30,35]). In this theory, a real quantity x is represented by an interval $\bar{x} = [\bar{x}.inf, \bar{x}.sup]$ of representable numbers in floating-point. Classical arithmetic operators are defined in such a way that the obtained results are guaranteed to contain the real quantity x. We can sum up them as follows:

$$\begin{aligned}
&\bar{x} \oplus \bar{y} = [\bar{x}.inf + \bar{y}.inf, \bar{x}.sup + \bar{y}.sup] \\
&\bar{x} \ominus \bar{y} = [\bar{x}.inf - \bar{y}.inf, \bar{x}.sup - \bar{y}.sup] \\
&\bar{x} \otimes \bar{y} = [\min(\bar{x}.inf \times \bar{y}.inf, \bar{x}.inf \times \bar{y}.sup, \bar{x}.sup \times \bar{y}.inf, \bar{x}.sup \times \bar{y}.sup), \\
&\qquad \max(\bar{x}.inf \times \bar{y}.inf, \bar{x}.inf \times \bar{y}.sup, \bar{x}.sup \times \bar{y}.inf, \bar{x}.sup \times \bar{y}.sup)] \\
&\bar{x} \oslash \bar{y} = \left[\min\left(\frac{\bar{x}.inf}{\bar{y}.inf}, \frac{\bar{x}.inf}{\bar{y}.sup}, \frac{\bar{x}.sup}{\bar{y}.sup}, \frac{\bar{x}.sup}{\bar{y}.sup}\right), \right. \\
&\qquad \left. \max\left(\frac{\bar{x}.inf}{\bar{y}.inf}, \frac{\bar{x}.inf}{\bar{y}.sup}, \frac{\bar{x}.sup}{\bar{y}.sup}, \frac{\bar{x}.sup}{\bar{y}.sup}\right)\right], \quad 0 \notin \bar{y}
\end{aligned} \tag{2}$$

The operators $\oplus, \ominus, \otimes$ and $\oslash$ are also named *inclusion functions* of the arithmetical operators $+, -, \times, /$. We can define by the same manner mathematical functions, like $\sin \bar{x}, \cos \bar{x}, \log \bar{x}, \exp \bar{x}$ with intervals. In a more general way, an inclusion function $\square f$, associated to the planar function f, permits to estimate the set of values taken by f on a 2-D domain I. In two dimensions, I is the cartesian product between two one-dimensional (1-D) intervals, *i.e.* $I = \bar{x} \times \bar{y} = [\bar{x}.inf, \bar{x}.sup] \times [\bar{y}.inf, \bar{y}.sup]$.

Definition 2.1 (*Inclusion function, general case*). Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a given function. $\square f$ is said an inclusion function of f if it computes, for a m-dimensional interval I, the n-dimensional interval $\square f(I)$ such that:

$$x \in I \Rightarrow f(x) \in \square f(I) \tag{3}$$

To build a complex inclusion function $\square f$, we could combine known inclusion functions, since $\square f \circ g = \square f(\square g)$. Another approach can be to develop $\square f$ thanks to Taylor series, in order to reduce the error induced by the inclusion function [21]. We can now easily implement an *absence oracle*. Let $\mathscr{C}$ be an implicit curve given by the planar function $f : \mathbb{R}^2 \to \mathbb{R}$, such that $f(x, y) = 0$. If $0 \notin \square f(I)$, then $0 \notin f(I)$, *i.e.* $f(x, y) \neq 0$ for all points (x, y) in I. Clearly, I does not intersect $\mathscr{C}$.

To find the solutions of f in the input interval I, we could use an analytical Newton like method that considers the derivatives of f. But, Snyder [29] has proposed a simple recursive algorithm to build a set of 2-D intervals $\mathscr{E}$ that bounds the function f, and conserves the topology of $\mathscr{C}$. By considering an inclusion function $\square f$, and an initial interval $I \subset \mathbb{R}^2$ such that $\mathscr{C} \cap I \neq \emptyset$, it first evaluates $\square f$ on I. Then, by applying the absence oracle, we can determine if I is a *feasible region*, that is, $0 \in \square f(I)$ or not. In the first case, I is the solution of this algorithm. We then apply recursively the absence oracle on I and its sub-intervals. Thus, we have to choose a *solution acceptance set constraint*, specifying when a region may be finally accepted. For example, I should satisfy $w(\square f(I)) < \delta$, where $\delta \in \mathbb{R}$ may be a small value and w is the width of I. This condition permits to refine the solution by subdividing I into several sub-regions. Then, the algorithm keeps on recursively treating intervals and subdividing them by generating a quadtree [27] for example, until all the solution intervals respect the acceptance constraint. Lopes et al. [18] propose to carry the first derivative of f in the stopping criterion. By constructing an interval $\square f'(I)$ with the two components of the normalized gradient of f, one have to test if I satisfy $w(\square f(I)) < \delta \vee w(\square f\prime(I)) < \beta$, for some $\beta \in \mathbb{R}$. Whatever the criterion chosen to stop this recursive algorithm, it ensures that the set of intervals $\mathscr{E}$ contains the curve $\mathscr{C}$ and geometrically approximates it. This article deals with subdivision techniques based on interval arithmetic, but we could study other tools [19], like Bernstein coefficients [1,11], spline functions [10], *etc.* as long as they build a decomposition of the plane into isothetic rectangles.

2.2. Irregular discrete geometry tools

Since the birth of digital plotters on matrix screens, representing discrete objects (lines, circles, *etc.*) has stood as a classical and still open problem. For example, Bresenham has dealt with tracing lines [4] and circles [5] on a discrete regular grid in the 1960–1970s. "Regular" means that all the pixels of the grid have the same size, and can be easily

indexed. In discrete geometry, the supercover model [8,17] has become a classical model to characterize a discrete object $\mathscr{S}$ belonging to a regular grid, in relation to $\mathscr{O}$, the underlying real object it represents. It mainly guarantees that $\mathscr{O}$ is contained into $\mathscr{S}$. In the continuity of this theory, we present here the supercover model on *irregular isothetic grids*. Indeed, those grids can be clearly compared with 2-D intervals given by an arithmetic analysis for spatial approximation of planar curves.

We first define an irregular isothetic grid, denoted $\mathbb{I}$, as a tiling of the plane with isothetic rectangles. Each rectangle R (also called *cell*) of $\mathbb{I}$ is defined by its center $(x_R, y_R) \in \mathbb{R}^2$ and a size $(l_R^x, l_R^y) \in \mathbb{R}^2$. In our framework, adjacency relation is an important feature that we depict through the following definitions.

Definition 2.2 (*Adjacency*). Let R_1, R_2 be two cells. R_1 and R_2 are adjacent if:

$$or \begin{cases} |x_{R_1} - x_{R_2}| = \frac{l_{R_1}^x + l_{R_2}^x}{2} \text{ and } |y_{R_1} - y_{R_2}| \leqslant \frac{l_{R_1}^y + l_{R_2}^y}{2} \\ |y_{R_1} - y_{R_2}| = \frac{l_{R_1}^y + l_{R_2}^y}{2} \text{ and } |x_{R_1} - x_{R_2}| \leqslant \frac{l_{R_1}^x + l_{R_2}^x}{2} \end{cases} \quad (4)$$

Definition 2.3 (*Arc*). Let $\mathscr{S}$ be a set of non-overlapping cells, $\mathscr{S}$ is an arc if and only if for each element of $\mathscr{S} = \{R_i, i \in \{1, \ldots, n\}\}, R_i$ has exactly two adjacent cells in $\mathscr{S}$, except R_1 and R_n which are called extremities of the arc.

We now consider the extension of the supercover model from [8] on irregular isothetic grids [6] to digitize Euclidean objects on $\mathbb{I}$. We also present some interesting properties of this model.

Definition 2.4 (*Supercover on irregular isothetic grids*). Let F be an Euclidean object in $\mathbb{R}^2$. The supercover $\mathbb{S}(F)$ is defined on an irregular isothetic grid $\mathbb{I}$ by:

$$\begin{aligned} \mathbb{S}(F) &= \{R \in \mathbb{I} | \mathbb{B}^\infty(R) \cap F \neq \emptyset\} \\ &= \left\{ R \in \mathbb{I} | \exists (x,y) \in F, |x_R - x| \leqslant \frac{l_R^x}{2} \text{ and } |y_R - y| \leqslant \frac{l_R^y}{2} \right\} \end{aligned} \quad (5)$$

where $\mathbb{B}^\infty(R)$ is the rectangle centered in (x_R, y_R) of size (l_R^x, l_R^y) (if $l_R^x = l_R^y, \mathbb{B}^\infty(R)$ is the ball centered in (x_R, y_R) of size l_R^x for the L_∞ norm).

Proposition 2.1 (Proof in [6]). *Let F, G be two Euclidean objects in $\mathbb{R}^2$, and an $\mathbb{I}$-grid, we have:*

$$\begin{aligned} &\mathbb{S}(F \cup G) = \mathbb{S}(F) \cup \mathbb{S}(G) \\ &\mathbb{S}(F \cap G) \subseteq \mathbb{S}(F) \cap \mathbb{S}(G) \\ &\text{if } F \subseteq G \text{ then } \mathbb{S}(F) \subseteq \mathbb{S}(G) \end{aligned} \quad (6)$$

We can notice that the last inequality can be interpreted as an inclusion function, in respect to the interval arithmetic definition we have given before. Moreover, we can draw a parallel between the 2-D irregular object $\mathscr{S}$ and the set of 2-D intervals $\mathscr{E}$ given by Snyder's algorithm. In the following, we denote any irregular object $\mathscr{E}$.

We now present the arc reconstruction algorithm we use in our implicit curve approximation algorithm (Section 3). This approach also respects the supercover model we have just presented. The algorithm proposed in [7] permits to decompose an irregular arc into segments. The algorithm inspired from [28] principally uses the construction and update procedures of a *visibility cone* [25], and can be sketched as follows. We first fix the extremity p_0 of the first segment such that $p_0 \in R_0$. We note e_0 the Euclidean segment shared by R_0 and R_1, and we consider the first cone $C_0(p_0, s, t)$ such that s and t coincide with the extremities of e_0, and $\{p_0, s, t\}$ are sorted counterclockwise. Then, for each cell R_i, we consider the shared segment e_i between R_{i-1} and R_i, and the current cone $C_j(p_j, s, t)$ is updated. When the update procedure fails, the visibility cone vanishes, and a new cone $C_{j+1}(p_{j+1}, s, t)$ is set up, and we add the point p_{j+1} to the reconstruction: to compute the new cone, authors of [7] consider the bisector of the cone and define p_{j+1} as the midpoint of the intersection between the bisector and the pixel R_{i-1}. Fig. 1 illustrates the construction of cones in an arc, and the resulting segmentation into lines.

3. Topological and geometrical representations of an implicit planar curve

To represent the shape of a set of cells $\mathscr{E}$ built by an interval analysis algorithm like in [29], we have chosen in [32] an incremental directional approach to build its associated Reeb graph $\mathscr{G}$, as in continuous space. It is an interesting structure introduced by Reeb [26] based on the Morse theory [13]. This graph is also used in many applications for surface and curve description [14,31,34]. The Reeb graph $\mathscr{G} = (V, E)$ is associated to a *height function* h defined on $\mathscr{E}$, and the nodes of $\mathscr{G}$ represent the critical points of h. More exactly, we denote them *begin* (b), *merge* (m), *split* (s) and *end* (e) nodes (see Fig. 2 and Algorithm 1). Without loss of generality, we can suppose that we choose, as the height function h, the left-to-right orientation along X axis, *i.e.* the height function h is defined along X axis. Moreover, we want to represent an edge between two nodes by an irregular arc. To respect this constraint, we use a simple procedure to update and recode an arc with one or more adjacent cells (see Fig. 3 and [32] for more details). In Algorithm 1, we present the main stages of our reconstruction algorithm of complex irregular objects. As an arc A is associated to an edge $n_1 - n_2 \in E$, $n_1, n_2 \in \{b, e, m, s\}$, and since we have chosen a left-to-right orientation for $\mathscr{G}$, we denote in this algorithm $inf(A) = n_1$ and $sup(A) = n_2$. For the sake of simplicity, we say that an arc A and a cell R are adjacent if there exists a cell R' in A such that R and R' are adjacent. Finally, we denote the set of irregular arcs $\mathscr{A}$ and the set of line segments $\mathscr{L}$, outputs of Algorithm 1.

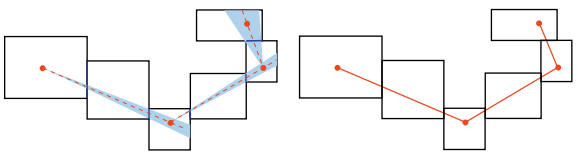

Fig. 1. An example of the construction of cones in an arc (left), and the reconstruction into segments we obtain (right).

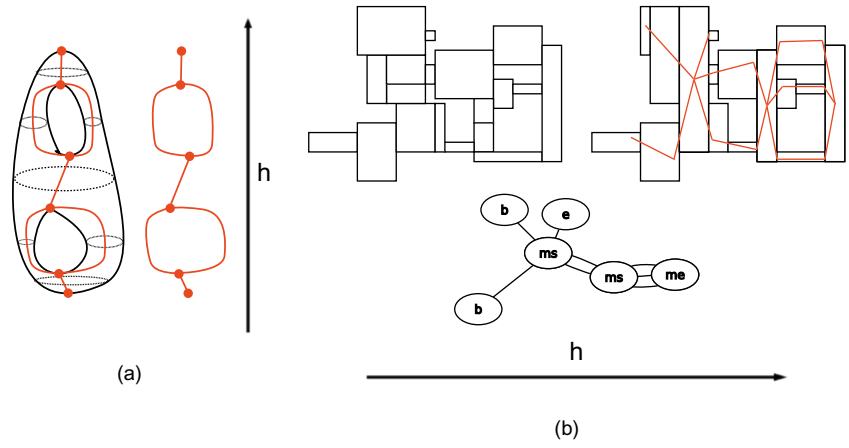

Fig. 2. (a) An example of the Reeb graph $\mathscr{G}$ of a continuous object $\mathscr{E}$. The node of G represent the critical points of h (maxima, minima, inflection points), and an edge is a connected component of $\mathscr{E}$ between two critical points. (b) An example of an irregular object $\mathscr{E}$ (left), the final recoded structure with arcs, the obtained polygonalization *(right)* and the Reeb graph associated to the height function f defined on $\mathscr{E}$ (bottom). Finally, a node $n_1n_2,\ n_1, n_2 \in \{b, e, m, s\}$ represents two nodes n_1 and n_2 at the same place.

Algorithm 1. Reconstruction of an object $\mathscr{E} \in \mathbb{I}$

Input: $\mathscr{E} = \{R_i\}_{i=1,\ldots,n}$, a set of n 2-D cells in $\mathbb{I}$.
Output: $\mathscr{G} = (V, E)$, the Reeb graph associated to $\mathscr{E}$
$\mathscr{L}$, the reconstruction by lines of $\mathscr{E}$
$\mathscr{A} = \{A_i\}_{i=1,\ldots,n_a}$, the set of n_a arcs recoding $\mathscr{E}$

1. **begin**
2. Add an edge $b - e$ in $\mathscr{G}\{V = \{b, e\},\ E = \{b - e\}\}$
3. Create the arc A_1 with $inf(A_1) = b$ and $sup(A_1) = e\{n_a = 1\}$
4. **for all** cells $R_i, 2 \leqslant i \leqslant n$ **do**
5. **for all** arcs $A_j, 1 \leqslant j \leqslant n_a$ **do**
6. **if** one cell R_i is adjacent with one arc A_j **then**
7. Update A_j with R_i
8. **if** p cells $R_i, R_{i+1}, \ldots, R_{i+p}$ are adjacent with A_j **then**
9. Update A_j with $R_i, R_{i+1}, \ldots, R_{i+p}$
10. Add a split node s in $\mathscr{G}\{deg(s) = p + 1\}$
11. $sup(A_j) = s$
12. Link p new arcs $A_{n_a+1} = R_i, \ldots, A_{n_a+p} = R_{i+p}$ with A_j
13. $inf(A_{n_a+1}) = \cdots = inf(A_{n_a+p}) = s$
14. **if** one cell R_i is adjacent with p arcs $A_j, A_{j+1}, \ldots, A_{j+p}$ **then**
15. Update $A_j, A_{j+1}, \ldots, A_{j+p}$ with R_i
16. Add a merge node m in $\mathscr{G}\{deg(m) = p + 1\}$
17. $sup(A_j) = \cdots = sup(A_{j+p}) = m$
18. Link a new arc $A_{n_a+1} = R_i$ with $A_j, A_{j+1}, \ldots, A_{j+p}$
19. $inf(A_{n_a+1}) = m$
20. **if** a cell R_i is adjacent with any arc A_j **then**
21. Add an edge $b - e$ in $\mathscr{G}$
22. Create an arc A_{n_a+1} with $inf(A_{n_a+1}) = b$ and $sup(A_{n_a+1}) = e$
23. Perform the polygonalization over the set of arcs $\{A_j\}_{j=1,\ldots,n_a}$ considering $\mathscr{G}$
24. **end**

Indeed, our method builds a complete topological representation of $\mathscr{E}$ with the Reeb graph $\mathscr{G}$ by recognizing and linking critical nodes. At the end of our approach, we use the arc reconstruction algorithm (see Section 2) to build a polygonal description of $\mathscr{E}$. We process the geometrical reconstruction of $\mathscr{E}$ from *merge* and *split* nodes (internal nodes of our structure) to the others. Thus, all the *merge* or *split* nodes in $\mathscr{G}$ are represented by a single point. With this choice, we force the computed points to be correctly placed in the cells associated to those important nodes. Moreover, a left-to-right polygonal reconstruction would not be able to respect the symmetry of the object $\mathscr{E}$. The complexity in the worst case of this algorithm is $\mathscr{O}(n_a \times n)$: for all R_i, we test all the n_a arcs A_j to find which cell R_i is adjacent with A_j. In Fig. 4, we present the Reeb graph and the polygonalization obtained by our approach on a simple example of a cubic curve reconstruction.

In Fig. 4, we can notice that the topology is always correctly represented in $\mathscr{G}$. If $\delta = 0.5$ (second case), the Reeb graph contains a hole. This "error" could stand for many implicit curve representation algorithm, since the interval analysis stage build a hole in the set of cells $\mathscr{E}$. The quality of geometrical and topological representations mainly depends on the choice of the stopping criterion during this part of our system. Thanks to the arc reconstruction algorithm (see Section 2), the number of line segments is very competitive, in comparison with the classical methods of implicit curve approximation, where a segment is built in each cell of $\mathscr{E}$. Indeed, thanks to the arc reconstruction algorithm we have presented in Section 2, we build less segments than the cells contained in $\mathscr{E}$. For example, in Fig. 4, if $\delta = 1.0$, we have $|\mathscr{E}| = 24$, $|\mathscr{L}| = 5$, and a classical technique of approximation like the one of Snyder would achieve a set of about 14 segments. In Section 5.1, we present more experiments about the quality of our polygonal approximation algorithm. Furthermore, we can notice that we propose a polygonal reconstruction of $\mathscr{C}$, even if the set of intervals is not topologically correct.

4. Dynamic reconstruction of an implicit planar curve

In this section, we want to efficiently update our reconstruction of implicit curves, considering one or more local refining and grouping operations (for the refining scheme, see also [33]). We present here two efficient algorithms to locally change the Reeb graph and the polygonal approximation we compute in Algorithm 1. $\mathscr{G}, \mathscr{L}$ and $\mathscr{A}$ represent

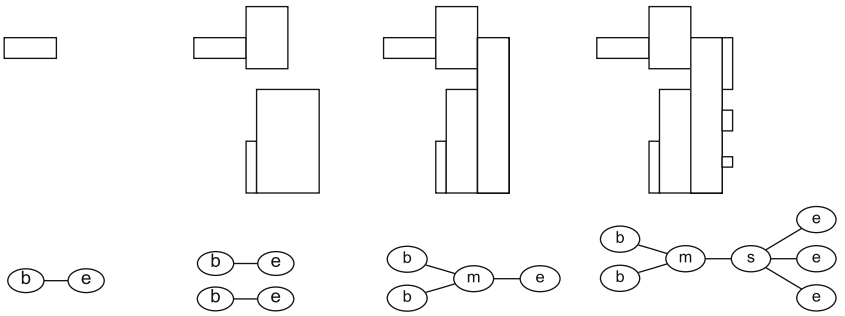

Fig. 3. The recognized irregular arcs and the associated Reeb graph for some iterations of our algorithm. First, we initialize an arc with the cell with the smallest left border. Then, we progressively update and recode arcs. The third and fourth images present *merge* and *split* phases. We can notice that in one hand the recoding stage is not detailed in this figure, and in the other hand the edges $m - s$ represent an irregular arc with one cell in this example.

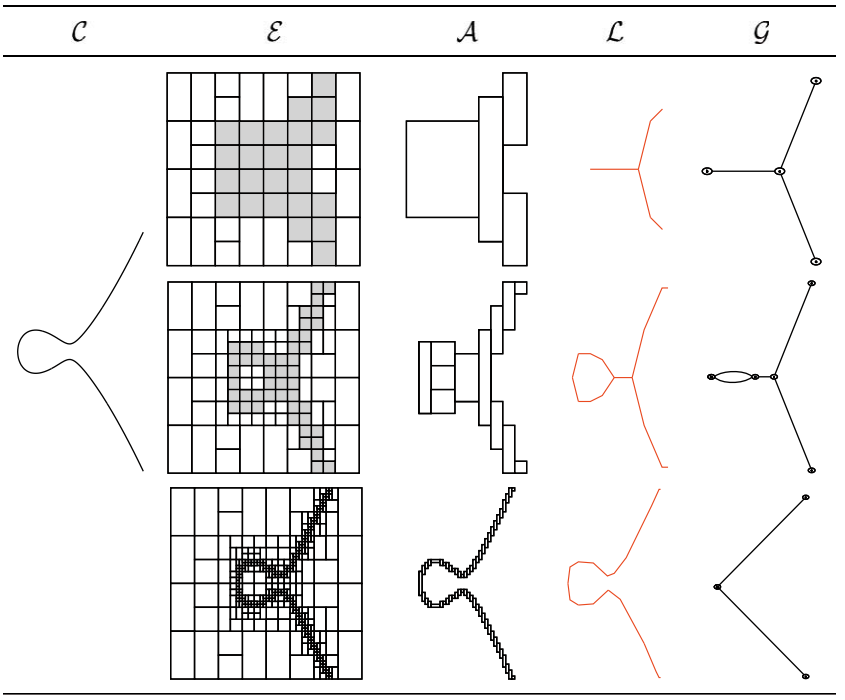

Fig. 4. Some results for our implicit curve reconstruction algorithm. We have chosen the cubic function $f(x,y) = y^2 - x^3 + x - 0.5$ in the interval $[-4.0; 4.0] \times [-4.0; 4.0]$. In the three cases, δ, the stopping criterion for the interval analysis part, represents the width of the rectangles in $\mathscr{E}$. We have fixed it at 1.0, 0.5 and 0.1, respectively.

the elements computed by Algorithm 1 on the original object $\mathscr{E}$. Finally, for each arc $A_i \in \mathscr{A}$, its associated edge in $\mathscr{G}$ is denoted in a lower case a_i.

4.1. Local refining scheme by inclusion

Let $R \in \mathscr{E}$ be a cell where we want to have a more precise decomposition, *e.g.* a local quadtree refinement from level t to $t+1$. We denote the function RS the refining scheme performed on R. Thus, $\mathrm{RS}(R)$ is the set of cells in the domain bounded by R obtained by the process RS. Thus, with those notations, our problem consists in updating $\mathscr{G}, \mathscr{L}$ and $\mathscr{A}$, by considering $\mathrm{RS}(R)$. First, we consider all the irregular arcs in E that contain R. Then, we replace R with the smaller cells $\mathrm{RS}(R)$, and perform Algorithm 1 over those pixels to have a local topological and geometrical representation of them. Then, we branch the local reconstructed arcs represented in $\mathscr{G}_R, \mathscr{L}_R$ and $\mathscr{A}_R$ (associated to $\mathrm{RS}(R)$) in $\mathscr{G}, \mathscr{L}$ and $\mathscr{A}$, respectively. In Fig. 5, we illustrate the behaviour of our algorithm for the case where R belongs to one arc ($n_a = 1$ in Algorithm 2), and when $n_a > 1$, *i.e.* R belongs to several arcs.

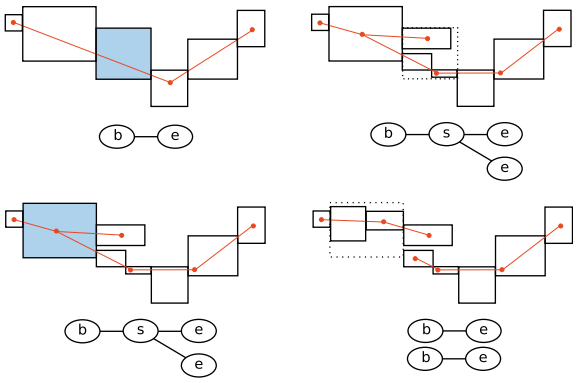

Fig. 5. Examples of results for Algorithm 2. The first reconstruction (top-left) contains one arc. Then, we choose the cell in dark and decompose it into two arcs (top-right). This cell belongs to one arc, thus $n_a = 1$ and the edge $b - e$ has to be divided into two before adding the node s. The next refinement (bottom) concerns three arcs, since we select to split the cell in dark ($n_a > 1$). This cell and its associated node (s) are removed before inserting the new edges.

Algorithm 2. Refine a cell in the reconstruction of an object $\mathscr{E} \in \mathbb{I}$

Input: An irregular decomposition $\mathscr{E}$, and a cell $R \in \mathscr{E}$
$\mathtt{RS}(R)$, the local decomposition of $R \in \mathscr{E}$ by $\mathtt{RS}$
$\mathscr{G}$, the Reeb graph associated to $\mathscr{E}$
$\mathscr{L}$, the reconstruction by lines of $\mathscr{E}$
$\mathscr{A}$, the set of arcs recoding $\mathscr{E}$
Output: $\mathscr{G}, \mathscr{L}$ and $\mathscr{A}$ are updated
1. **begin**
2. Find the n_a arcs $A_k, \ldots, A_{k+n_a} \in \mathscr{A}$ so that $R \in A_k, \ldots, A_{k+n_a}$
3. **if** $n_a > 1$ **then** [R represents a split or merge node in $\mathscr{G}$]
4. Remove the node $n \in \mathscr{G}$ associated to R
5. **else** [R is in one arc]
6. Split a_k into two edges a_k, a_{k+1}
7. Remove the polylines of $\mathscr{L}$ associated to the arcs $A_k, \ldots, A_{k+n_a}$
8. Remove R from $A_k, \ldots, A_{k+n_a}$
9. Algorithm 1 computes $\mathscr{G}_R, \mathscr{L}_R$ and $\mathscr{A}_R$ associated to $\mathtt{RS}(R)$
10. **for all** edges $a'_i \in \mathscr{G}_R$ **do**
11. **for all** edges $a_j \in \mathscr{G}, j = k, \ldots, k + n_a$ **do**
12. **if** A_j and A'_i are adjacent **then**
13. Branch a'_i to a_j
14. Merge the irregular arcs A_j and A'_i
15. Add all polylines from $\mathscr{L}_R$ to $\mathscr{L}$
16. **for all** arcs $A_j, j = k, \ldots, k + n_a$ **do**
17. Compute arc reconstruction on A_j
18. **end**

To find the n_a arcs required by the first operation of Algorithm 2, we use a pointer between the cell R and an arc A_k such that $R \in A_k$. If R is inside A_k, it is clear that $n_a = 1$, and so only one arc is considered. When R stands on one of the extremities of A_k, we have to consider all the arcs linked with A_k. Thanks to its associated edge a_k in $\mathscr{G}$, and the direct access to the edges linked of $\mathscr{G}$ with a_k, this search operation is computed in constant time $\mathscr{O}(1)$. Then, the next steps change the Reeb graph $\mathscr{G}$ in the set of n_a edges $\{a_j\}_{j=k,k+n_a}$ and impact the n_a irregular arcs $\{A_j\}_{j=k,k+n_a}$ found in the first operation. The topology of the irregular object $\mathscr{E}$ is locally updated. The rest of the graph does not change. In the last phase of the algorithm, the polygonal reconstruction is first computed in the arcs belonging to the local refinement $\mathscr{A}_R$. Then, this reconstruction may be propagated to the n_a global arcs in $\mathscr{A}$, as in Fig. 5 (top). Indeed, several strategies may be adopted. We could also link fine local line segments to global ones. We have preferred to propagate the reconstruction to adjacent arcs, to get an homogeneous polygonal structure. In the worst case, the update of this structure is locally performed on $\mathscr{A}_R$ and $\mathscr{A}$, and does not modify the rest of the reconstruction. The time complexity of this refining algorithm is in $\mathscr{O}(|\mathtt{RS}(R)| + \Delta n_a)$, where $\Delta = |\mathscr{G}_R| + \max_{k=1,\ n_a} |A_k|$ represents (1) the size of the local Reeb graph that impacts the connection step (Algorithm 2, lines 10–14) and (2) the maximum size of the neighbor arcs, since we process the polygonal reconstruction over all those arcs (lines 16–17). Finally, we can notice that our algorithm locally impacts the topological and geometrical reconstruction we globally compute for $\mathscr{E}$. To perform our algorithm on several refined cells, we first update the Reeb graph with the local graphs using the same approach. Then, we perform only once the final polygonalization over the concerned arcs.

4.2. Local grouping scheme by inclusion

Let $R_1, R_2, \ldots, R_n$ in $\mathscr{E}$ be a set of cells we want to group together (*e.g.*, a local merge of four nodes in a quadtree). In an interactive application, the selection of $\{R_i\}_{i=1,n}$ should be performed by "drawing" a rectangle in the plane. In this case, we force the user to choose packs of adjacent cells in (maybe several) arcs of $\mathscr{A}$. As in the previous algorithm, we first search all the arcs in $\mathscr{E}$ that contain a cell of $\{R_i\}_{i=1,n}$. We remove the cells of $\{R_i\}_{i=1,n}$ in those arcs and update the associated edges in $\mathscr{G}$. As in Algorithm 2, we have to consider if a cell R_i such that $R_i \in A_j$ stands or not on the extremities of A_j. Then, we replace those cells with the minimal bounding cell $B = \mathtt{GS}(\{R_i\}_{i=1,n})$ such that $R_1, R_2, \ldots, R_n \subseteq B$. The local reconstruction of this cell is a unique edge $b - e$ in $\mathscr{G}_B$ and a single point in $\mathscr{L}_B$. We just have to branch those simple local informations to global ones, and perform an arc reconstruction over the updated arcs $\mathscr{A}$. In Fig. 6, we illustrate two cases of our grouping cells algorithm.

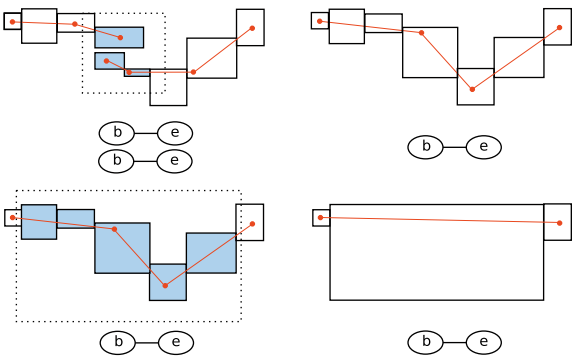

Fig. 6. Examples of results for Algorithm 3. The first update *(top)* permits to merge two edges in the Reeb graph. The second one *(bottom)* does not change the topology, but it builds a simpler polygonal reconstruction (one segment) and a very coarse arc description. The selection drawn by the user is depicted with dotted lines.

Algorithm 3. Group cells in the reconstruction of an object $\mathscr{E} \in \mathbb{I}$

Input: An irregular decomposition $\mathscr{E}$ and a set of cells $\{R_i\}_{i=1,n}$
$\mathtt{GS}(\{R_i\}_{i=1,n}) = B$, the bounding cell of $\{R_i\}_{i=1,n} \in \mathscr{E}$ by $\mathtt{GS}$
$\mathscr{G}$, the Reeb graph associated to $\mathscr{E}$
$\mathscr{L}$, the reconstruction by lines of $\mathscr{E}$
$\mathscr{A}$, the set of arcs recoding $\mathscr{E}$
Output: $\mathscr{G}, \mathscr{L}$ and $\mathscr{A}$ are updated

```
1. begin
2. Find the n_a arcs A_k,...,A_{k+n_a} ∈ A so that R_i ∈ A_k,...,A_{k+n_a}, i = 1,...,n
3. for all arcs A_j, j = k,...,k+n_a do
4.   for all cells R_i, i = 1,...,n do
5.     if R_i ∈ A_j then
6.       Remove R_i from A_j
7.       Update the edge a_j in G
8. Algorithm 1 computes G_B, L_B and A_B associated to B
9. for all edges a_j, j = k,...,k+n_a do
10.   if A_j and A_1 are adjacent then [A_B contains only one element]
11.     Branch a_1 to a_j
12.     Merge the arcs A_j and A_1
13. for all arcs A_j, j = k,...,k+n_a do
14.   Compute arc reconstruction on A_j
15. end
```

In Algorithm 3, the first search operation is still in constant time $\mathscr{O}(1)$ par cell R_i, as in the previous algorithm. The next operation has to be performed on all the cells of $\{R_i\}_{i=1,n}$ and all the arcs $A_k, \ldots, A_{k+n_a}$. Thus, it only modifies those n_a arcs and their associated edges in the Reeb graph. Then, the local reconstruction by Algorithm 1 is very fast (in $\mathscr{O}(1)$), since we only consider the cell B. The merging phase only impacts global arcs and edges, and does not change the rest of our representations. Finally, the arc reconstruction is still propagated locally in $\mathscr{A}$. The time complexity of Algorithm 3 is in $\mathscr{O}(\Gamma \times n_a)$, where we have denoted $\Gamma = |\{R_i\}_{i=1,n}| + \max_{k=1,n_a}|A_k|$. Those operations are due to: (1) we treat the set of cells $\{R_i\}$ and (2) we perform again the polygonal reconstruction over the n_a adjacent arcs of B.

4.3. Summary

We have proposed two fast algorithms that locally update the topological information computed by Algorithm 1 by splitting or linking nodes in the Reeb graph $\mathscr{G}$. Then, they build a new correct and efficient polygonal reconstruction $\mathscr{L}$ by only considering the n_a adjacent arcs concerned by the update. We can notice that we always respect the inclusion rule implied by the extended supercover model.

5. Experiments

The algorithms we have presented until now permit to define a global system for implicit curve approximation, that refines or groups the interval analysis by considering one or several processes: (1) like other tracing algorithms, we can control the topology, and update intervals only if they respect a given topological constraint; (2) contrary to those algorithms, we may also estimate the quality of the resulting polygonal approximation. For example, the local curvature could be a relevant feature to improve the approximation and (3) refinement and grouping processes can be finally managed by a supervised manner. Indeed, any user could manually choose the cells to refine or to group. In this section, we illustrate our general system with two applications for topological and geometrical representations of implicit curves. We first compare the quality and the speed of our method with classical methodologies and show examples of manual refining and grouping processes. Then, we present an automatic refining program based on the local curvature of the polygonal reconstruction of the curve.

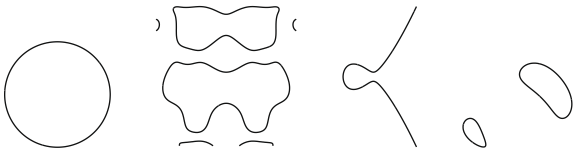

Fig. 7. A small overview of the four curves tested in this section. From left to right: *circle, trigo, cubic* and *two blobs*.

5.1. Static polygonal reconstruction of implicit curves

Here, we compare the speed and the quality of our method with relevant implicit curve tracing algorithms [18,29]. We have chosen several functions f that we approximate on a domain $\Omega = [-\omega; \omega] \times [-\omega; \omega] \subset \mathbb{R}^2$. We first consider two versions for the rectangles decomposition step. In the *spatially adaptive decomposition* [18,29], we only test if the size of a subdivided cell I is small enough for a good approximation: $w(\square f(I)) < \delta;\ \delta \in \mathbb{R}$ stands as our acceptation constraint. The *geometrically adaptive decomposition* [18] adds the first derivative to build less cells in $\mathscr{E}$. In this case, we have to check if a cell I verify $w(\square f(I)) < \delta \vee w(\square f'(I)) < \beta$, $\delta, \beta \in \mathbb{R}$. In our experiments, we have chosen $\delta = \frac{R}{2^{10}}$ and a varying β. We recall that $w(\square f'(I))$ is constructed with the two components $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ of the normalized gradient of f. Given a set of cells $\mathscr{E}$, we also propose to build a polygonal approximation with the algorithm of Snyder [29], where the intersections between the curve $\mathscr{C}$ and a cell R are computed thanks to interval arithmetic. We thus present here the comparison (accuracy and execution time) between four methods. $\mathtt{IA}_s$ [29] is the original algorithm of Snyder, using only interval analysis to approximate $\mathscr{C}$. $\mathtt{IA}_g$ [18] also considers the first derivative of f during the cell decomposition stage. We denote the use of Algorithm 1 on the cells built by those methods $\mathtt{DG}_s$ and $\mathtt{DG}_g$, respectively. In the list above are presented the equation $f(x, y) = 0$ of the curves we have chosen for those experiments, with the width of the 2-D domain ω (see also Fig. 7 for the simple plot of those curves):

Circle	$x^2 + y^2 - 1 = 0,\ \omega = 1.30$
Trigo	$x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2) - 1 = 0,\ \omega = 1.1$ (from [29])
Cubic	$y^2 - x^3 + x - 0.5 = 0,\ \omega = 5.2$ (from [18])
Two blobs	$\frac{601}{9} - \frac{872}{3}x + 544x^2 - 512x^3 + 256x^4 - \frac{2728}{9}y + \frac{2384}{3}xy - 768x^2y + \frac{5104}{9}y^2 - \frac{2432}{3}xy^2 + 768x^2y^2 - 512y^3 + 256y^4 = 0,\ \omega = 0.5$ (from [19])

We present in Table 1 the results of our comparison between the four methods. The level value k is considered in

Table 1
The comparison of our contribution with classical methodologies. The mean distance D between $\mathscr{L}$ and $\mathscr{C}$ is estimated thanks to interval arithmetic. The time T is presented in milli-seconds.

f	k	$\#\mathscr{L}$	T	D
IA $_s$ [29]				
Circle	4	52	21	6.38×10^{-4}
	6	164	100	1.69×10^{-4}
	8	620	252	4.23×10^{-5}
	10	2384	951	1.02×10^{-5}
Trigo	4	128	133	5.25×10^{-4}
	6	354	367	1.62×10^{-4}
	8	1482	1627	3.02×10^{-5}
	10	4778	5249	1.02×10^{-5}
Cubic	4	30	41	2.62×10^{-3}
	6	120	90	6.67×10^{-4}
	8	422	185	1.63×10^{-4}
	10	1642	722	4.20×10^{-5}
2 blobs	4	52	282	1.72×10^{-4}
	6	148	675	4.31×10^{-5}
	8	518	1580	1.08×10^{-5}
	10	1866	4978	2.69×10^{-6}
DG $_s$				
Circle	4	10	1	1.02×10^{-1}
	6	20	2	1.11×10^{-2}
	8	56	13	2.85×10^{-3}
	10	182	199	7.77×10^{-4}
Trigo	4	44	4	4.08×10^{-2}
	6	58	6	1.28×10^{-2}
	8	143	84	2.55×10^{-3}
	10	366	928	7.31×10^{-4}
Cubic	4	11	1	1.45×10^{-1}
	6	13	1	5.37×10^{-2}
	8	29	6	1.16×10^{-2}
	10	80	81	2.69×10^{-3}
2 blobs	4	4	1	1.40×10^{-2}
	6	6	5	5.88×10^{-3}
	8	30	19	1.19×10^{-3}
	10	72	157	4.29×10^{-4}
IA $_g$ [18]				
Circle	0.05	196	76	3.67×10^{-5}
	0.10	100	41	6.79×10^{-5}
	0.20	52	19	2.56×10^{-4}
	0.40	28	12	2.07×10^{-4}
Trigo	0.20	2800	3032	1.20×10^{-5}
	0.25	2604	2808	1.21×10^{-5}
	0.30	2374	2573	1.23×10^{-5}
	0.35	2110	2284	1.27×10^{-5}
Cubic	0.20	1298	604	1.12×10^{-5}
	0.25	1248	581	1.24×10^{-5}
	0.30	1214	563	1.40×10^{-5}
	0.35	1150	533	1.45×10^{-5}
2 blobs	0.20	1262	3460	8.31×10^{-6}
	0.25	1093	3073	8.32×10^{-6}
	0.30	975	2547	8.43×10^{-6}
	0.35	739	2009	8.48×10^{-6}
DG $_g$				
Circle	0.05	26	4	0.98×10^{-3}
	0.10	20	1	1.57×10^{-2}
	0.20	12	< 1	4.53×10^{-2}
	0.40	6	< 1	1.11×10^{-1}
Trigo	0.20	304	521	1.17×10^{-3}
	0.25	278	385	1.31×10^{-3}
	0.30	224	265	1.70×10^{-3}
	0.35	220	183	1.99×10^{-3}
Cubic	0.20	101	48	2.11×10^{-3}
	0.25	98	45	2.60×10^{-3}
	0.30	99	40	3.83×10^{-3}
	0.35	81	35	7.42×10^{-3}
2 blobs	0.20	70	85	4.13×10^{-3}
	0.25	75	71	4.80×10^{-3}
	0.30	68	56	5.49×10^{-3}
	0.35	61	40	6.20×10^{-3}

the computation of δ: $\delta = \frac{R}{2^k}$. For each test, we compute a set of line segments $\mathscr{L}$. We give the number of segments $\#\mathscr{L}$ to point out the complexity of the polygonal approximation, and thus the time required to plot it. The execution time T does not take into account the interval analysis step, and is depicted in milli-seconds. To rate the quality of each method, we have chosen to estimate the distance between a point $p_i = (x_i, y_i) \in \mathscr{L}$ and $\mathscr{C}$ with interval arithmetic. By construction, we know that the distance between p_i and $\mathscr{C}$ is in the worst case $\delta\sqrt{2}$. Thus, we compute a very fine interval decomposition in the domain $\left[x_i + \delta\sqrt{2}, x_i - \delta\sqrt{2}\right] \times \left[y_i - \delta\sqrt{2}, y_i + \delta\sqrt{2}\right]$. Then, we search for the nearest 2-D interval $\bar{\epsilon}_i$ from p_i. We consider the center of $\bar{\epsilon}_i$, that we denote $\bar{\epsilon}_i.mid$. Once we have computed the Euclidean distance $d(p_i, \bar{\epsilon}_i.mid)$ for each point p_i, we calculate the mean error of the polygonal approximation over all the n_p points in $\mathscr{L}$

$$D = \frac{1}{n_p} \sum_{i=1}^{i=n_p} d(p_i,\ \bar{\epsilon}_i.mid). \tag{7}$$

As we can see in Fig. 8, our method is very competitive for fine description of the curve (*e.g.* $k = 10$ with DG $_s$) because it approximates efficiently $\mathscr{C}$ with less segments (and thus a reduced execution time) than the

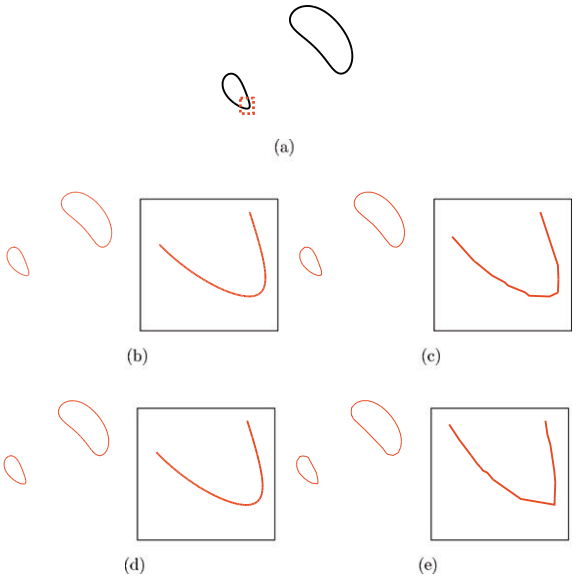

Fig. 8. The polygonal approximation of the *two blobs* curve (from [19]) obtained with the four methods we have tested. We have depicted a zoomed region with the dashed rectangle in (a), that is then illustrated for each case. (b) and (c) correspond to IA $_s$ and DG $_s$ with level $k = 10$, while (d) and (e) were plotted thanks to IA $_g$ and DG $_g$ defined by $\beta = 0.20$. In this region, we have observed that the number of segments is 204 for IA $_s$, 115 for IA $_g$ and 14 for DG $_s$ and DG $_s$.

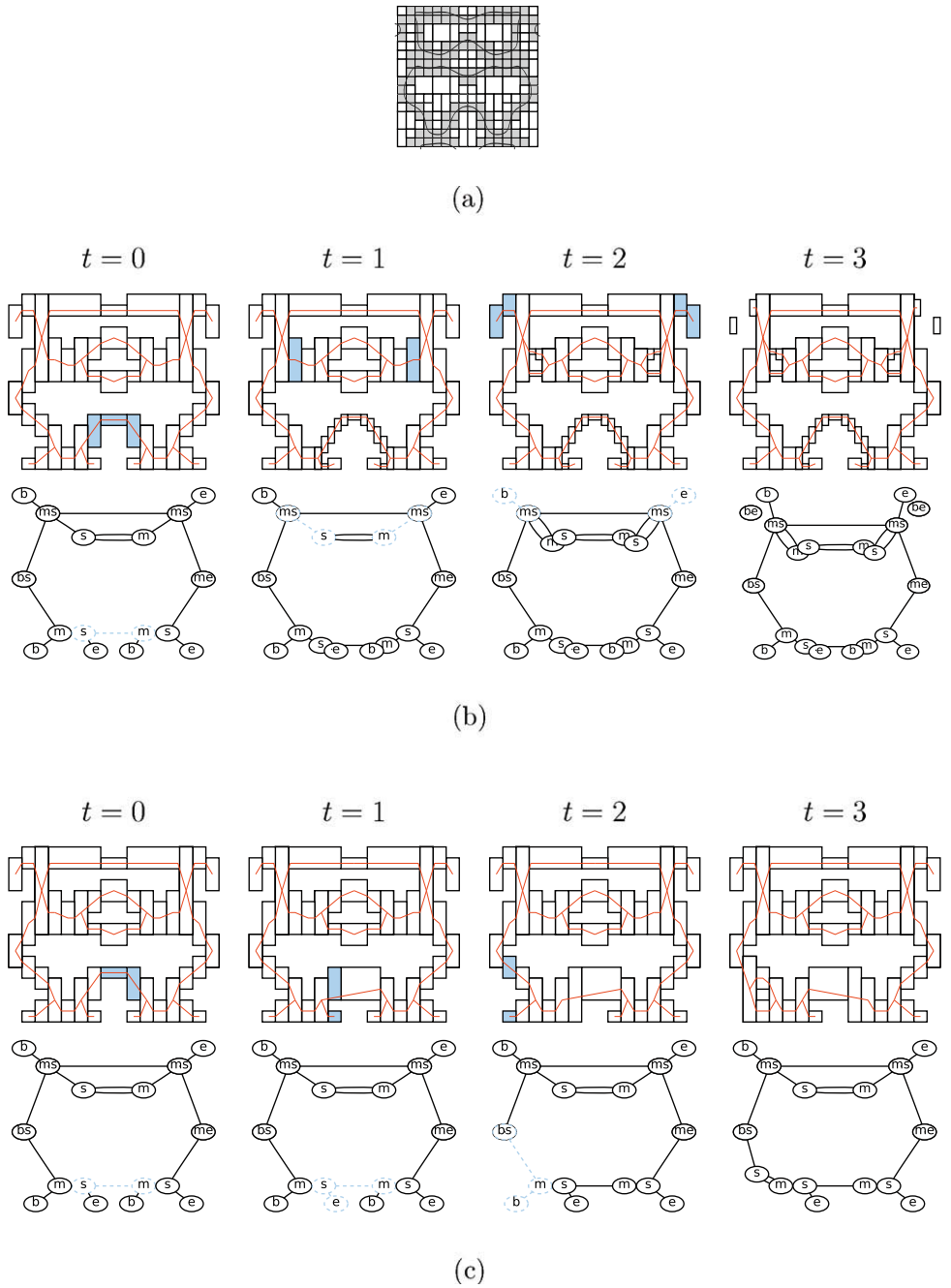

Fig. 9. Refining and grouping schemes results. The solution of $x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2) = 1$ on $[-1.1; 1.1] \times [-1.1; 1.1]$ (top) discretized in a set of cells by an algorithm described in [29]. For each update procedure, we show in the corresponding table the recoded set of cells and the polygonalization (middle). The selected cells $R_1, R_2, \ldots, R_n$ for the next iteration are presented in dark, and the considered arcs in the Reeb Graph are illustrated in dashed lines *(bottom)*.

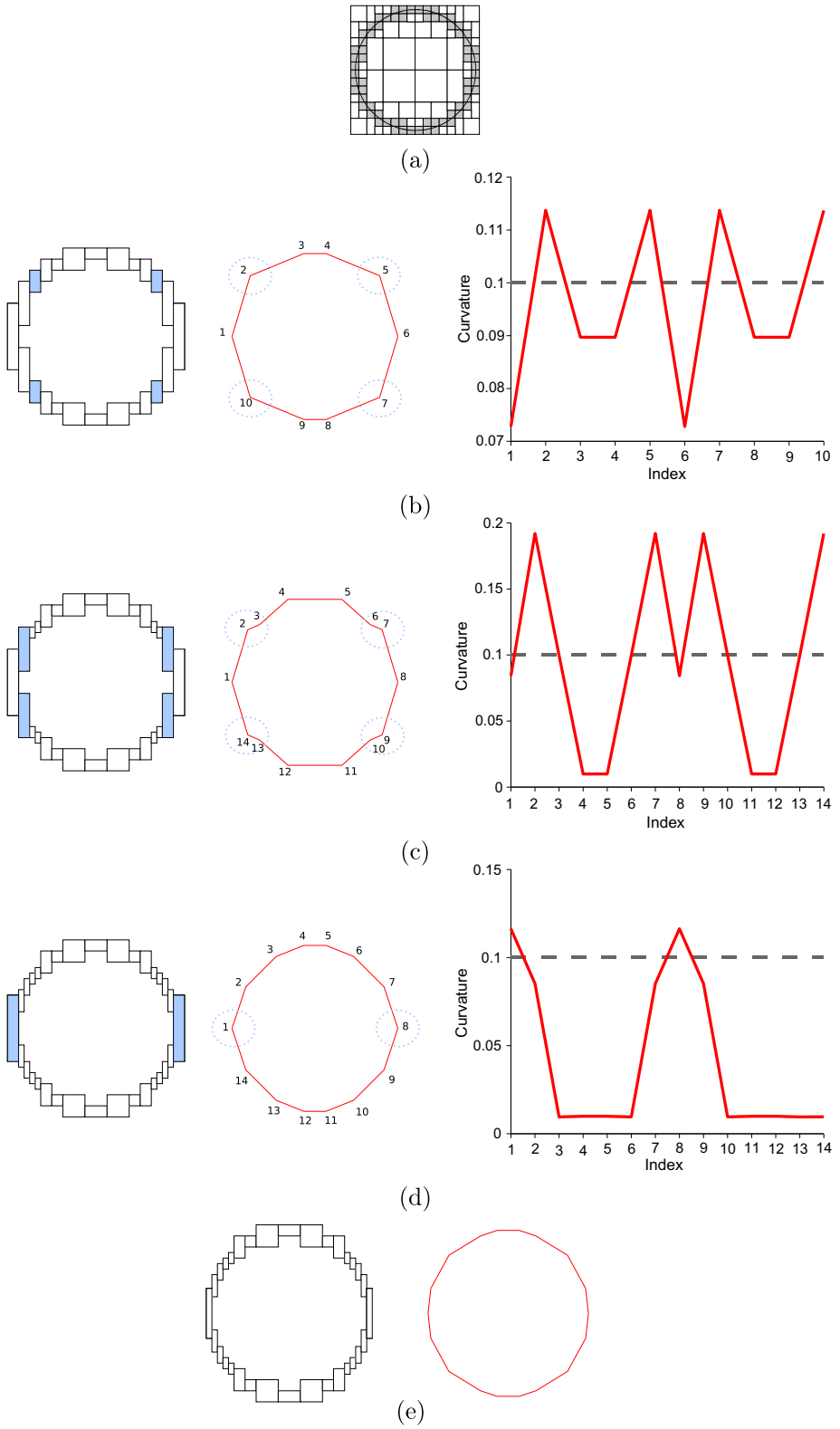

Fig. 10. Some iterations for the function $f : (x, y) \rightarrow (x^2 + y^2) - 1.0$. We first illustrate the interval analysis output (*top*). We have fixed here $\bar{\sigma} = 0.1$ and the successive discretization steps $\delta_1 = 0.1, \delta_2 = 0.05$. We depict the arc reconstruction (set $\mathscr{A}$) and the polygonal approximation (set $\mathscr{L}$). The curvature graph is illustrated, where we dot the curvature σ of each point in the counterclockwise order, and the max curvature $\bar{\sigma}$ is drawn in dashed lines. Finally, the refined cells and the high curvature points are highlighted.

classical method $\mathtt{IA}_s$. This behaviour can be observed even for a very complex function like *two blobs*: we plot the curve in 0.1 s with an error about 4.0×10^{-4}. Since we only consider the cells to compute the polygonal approximation, a coarser tiling of the plane with two variable rectangles significantly spoils the reconstruction. For instance, lower levels of decomposition k and high values of β make the set of segments computed with $\mathtt{DG}_s$ less correct than the one deduced from the approach $\mathtt{IA}_g$. The advantage of building larger cells where the derivative is small makes this classical algorithm more convenient than ours, as we can see in Fig. 8. Finally, we can notice that our contribution $\mathtt{DG}_s$ can be a very good approximation in respect to $\mathtt{IA}_g$. In fact, Lopes et al. have chosen this criterion to compute less segments in $\mathscr{L}$. Our proposed method is then an efficient polygonal reconstruction, since we quickly get few segments and a correct approximation of the curve.

5.2. Some examples of update operations on an implicit curve reconstruction

In this section, we have chosen to study *trigo*, the function f on the interval $\Omega = [-1.1; 1.1] \times [-1.1; 1.1]$, such that $f(x,y) = x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2)$ from [29]. As we have presented during this article, we first discretize this function on the set $\mathscr{E}$ thanks to the approach described in [29]. The width δ of the final computed ranges is fixed: $\delta = 0.1$ (see Fig. 9a). We now present some experiments for the refining process (Algorithm 2). Indeed, we successively update the elements computed by Algorithm 1 with local refinements by inclusion. We select several cells at each time, and use Algorithm 2. The new cells contained in the refinement are deduced from the same interval computation algorithm, with a smaller range width ($\delta = 0.05$). In the Fig. 9b, we show four times of update. The first update procedure, with the input presented in $t = 0$, does not modify the Reeb graph ($\mathscr{G}_0 = \mathscr{G}_1$) while the geometrical reconstruction ($\mathscr{A}_1$ and $\mathscr{L}_1$) is more precise. At time $t = 1$, we choose two cells that modify the Reeb graph $\mathscr{G}_1$ by splitting the edges $m - m$ and $s - s$. Finally, selecting the four cells at time permits to create two new edges in $\mathscr{G}_3$. For the grouping processes (Algorithm 3), we show in Fig. 9c the results for three updates of our reconstruction of f. The first one does not change the Reeb graph, the next one allows to build a simpler Reeb graph by deleting an $b - m$ edge, and by connecting the two neighbor s nodes. The last modification creates a split node by connecting two edges.

5.3. An automatic refining scheme

Here, we propose to study the initial polygonalization $\mathscr{L}$ and to refine the global reconstruction when a point in $\mathscr{L}$ has a high radius of curvature. The radius of curvature σ_q of a point $q \in \mathscr{L}$ is computed by considering the three consecutive points $p, q, r \in \mathscr{L}$ and the circle $C_{(p,q,r)}$ passing through p, q and r: $\sigma_q = \frac{1}{radius(C_{(p,q,r)})}$. With this process, we illustrate the use of an automatic refining criterion. More precisely, we first fix a maximum radius $\bar{\sigma}$, and we find for each arc of $\mathscr{A}$ the cell R where is located a point q with a radius $\sigma_q > \bar{\sigma}$. Indeed, we suppose that the underlying real curve $\mathscr{C}$ is smooth and with bounded curvature. Finally, we refine this cell and update the reconstruction (Algorithm 2). We could iteratively refine $\mathscr{L}$ until the polygonalization respects this inequality. In Fig. 10, we show some iterations of this auto refining program for the circle function $x^2 + y^2 - 1.0 = 0$.

6. Conclusion and future work

In this article, we have proposed a global framework for implicit curve approximation by line segments. It includes three efficient and fast algorithms to build and update the geometrical and topological representations of the curve. Moreover, thanks to an original irregular discrete approach, we build a very simple polygonal approximation. We have observed that our approach is very competitive with respect to classical methodologies. We currently work on an hybrid method based on the visibility cone tool and an interval analysis to guide the construction of the cone. We finally illustrate the use of various criterions (local curvature, interval analysis precision, user's queries, *etc.*) that we would like to include in our system. In particular we have presented an automatic polygonal refining program, based on the local curvature of the reconstructed line segments.

We would like to extend our system to three-dimensions (3-D) irregular objects recognition and representation. In this case, a 3-D set of irregular cells should be approximated with parts of planes. However, a hard work has to be driven to build a correct Reeb graph associated to such an object. Indeed, we shall study Forman's discrete adaptation of the Morse theory [2,12]. There exists many methods to compute the Reeb graph of any triangular objects [22], but we would like to avoid the use of a temporary triangular mesh to deduce the Reeb graph of the original voxelized object like in the common methods [34].

References

[1] L. Alberti, B. Mourrain, Visualisation of implicit algebraic curves, in: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications (PG'07), 2007, pp. 303–312.

[2] J.-D. Boissonnat, D. Cohen-Steiner, G. Vegter, Meshing implicit surfaces with certified topology, Technical Report, INRIA, Number 4930, 2003.

[3] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, G. Vegter, Meshing of surfaces, in: J.-D. Boissonnat, M. Teillaud (Eds.), Effective Computational Geometry for Curves and Surfaces, Mathematics and Visualization, Springer-Verlag, 2006.

[4] J. Bresenham, Algorithm for computer control of digital plotter, IBM System Journal 4 (1965) 25–30.

[5] J. Bresenham, A Linear algorithm for incremental display of circular arcs, Communications of the ACM 20 (2) (1977) 100–106.

[6] D. Coeurjolly, Supercover model and digital straight line recognition on irregular isothetic grids, in: Proceedings of the 12th International Conference on Discrete Geometry for Computer Imagery (DGCI 2005), LNCS, vol. 3429, 2005, pp. 311–322.

[7] D. Coeurjolly, L. Zerarga, Supercover model, digital straight line recognition and curve reconstruction on the irregular isothetic grids, Computer and Graphics 30 (1) (2006) 46–53.

[8] D. Cohen-Or, A. Kaufman, Fundamentals of surface voxelization, Graphical Models and Image Processing: GMIP 57 (6) (1995) 453–461.

[9] L.H. de Figueiredo, J. Stolfi, Self-validated Numerical Methods and Applications Brazilian Mathematics Colloquium Monograph, IMPA, Rio de Janeiro, Brazil, 1997.
[10] G. Elberand, M. Kim, Geometric constraint solver using multivariate rational spline functions, in: Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications (SMA'01), New York, NY, USA, 2001, pp. 1–10.
[11] G. Farin, Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, third ed., Academic Press Professional, Inc., San Diego, USA, 1993.
[12] R. Forman, Morse theory for cell-complexes, Advances in Mathematics 134 (1998) 90–145.
[13] A. Gramain, Topologie des Surfaces, Presses Universitaires Françaises, 1971.
[14] F. Hétroy, Méthodes de Partitionnement de Surfaces, PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 2003.
[15] S.L. Horowitz, T. Pavlidis, Picture segmentation by a directed split-and-merge procedure, in: The SPHINX-II Speech Recognition System: An Overview, Computer Speech and Language, vol. 2, 1974, pp. 137–148.
[16] R.B. Kearfott, Interval computations: introduction, uses, and resources, Euromath Bulletin 2 (1) (1996) 95–112.
[17] R. Klette, A. Rosenfeld, Digital Geometry, Elsevier, San Fransisco, USA, 2004.
[18] H. Lopes, J. Oliveira, L.H. de Figueiredo, Robust adaptive approximation of implicit curves, in: Proceedings of the XIV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'01), 2001.
[19] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, G. Wang, Comparison of interval methods for plotting algebraic curves, Computer Aided Geometric Design 19 (7) (2002) 553–587.
[20] R.E. Moore, C.T. Yang, Interval Analysis I, Technical Report Space Div. Report LMSD285875, Lockheed Missiles and Space Co., 1959.
[21] R.E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1966.
[22] V. Pascucci, G. Scorzelli, P. Bremer, A. Mascarenhas, Robust on-line computation of Reeb graphs: simplicity and speed, ACM Transactions on Graphics 26 (3) (2007).
[23] S. Plantinga, G. Vegter, Isotopic approximation of implicit curves and surfaces, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP'04), Nice, France, 2004, pp. 245–254.
[24] S. Plantinga, G. Vegter, Isotopic meshing of implicit surfaces, Visual Computer 23 (2007) 45–58.
[25] F.P. Preparata, M.I. Shamos, Computational Geometry – An Introduction, Springer, 1985.
[26] G. Reeb, Sur les Points Singuliers d'une Forme de Pfaff Complément Intégrable ou d'une Fonction Numérique, Comptes Rendus de L'Académie des Sciences, Paris 222 (1946) 847–849.
[27] H. Samet, Hierarchical spatial data structures, Proceedings of Design and Implementation of Large Spatial Databases, First Symposium (SSD'89), vol. 409, Springer, Santa Barbara, California, 1989, pp. 193–212.
[28] I. Sivignon, R. Breton, F. Dupont, E. Andres, Discrete analytical curve reconstruction without patches, Image and Vision Computing 23 (2) (2005) 191–202.
[29] J.M. Snyder, Interval analysis for computer graphics, Computer Graphics 26 (2) (1992) 121–130.
[30] J. Stolfi, L.H. de Figueiredo, An introduction to affine arithmetic, TEMA Tendências em Matemática Aplicada e Compututacional 4 (3) (2003) 297–312.
[31] T. Tung, Indexation 3D de Bases de Données d'Objets 3D par Graphes de Reeb Améliorés, PhD thesis, Telecom Paris, ENST/TIC, Paris, France, 2005.
[32] A. Vacavant, D. Coeurjolly, L. Tougne, Topological and geometrical reconstruction of complex objects on irregular isothetic grids, in: Proceedings of the 13th International Conference on Discrete Geometry for Computer Imagery (DGCI 2006), Szeged, Hungary, LNCS, vol. 4245, 2006, pp. 470–481.
[33] A. Vacavant, D. Coeurjolly, L. Tougne, Dynamic reconstruction of complex planar objects on irregular isothetic grids, in: Proceedings of the 2nd International Symposium on Visual Computing (ISVC 2006), Lake Tahoe, Nevada, LNCS, vol. 4292, 2, 2006, pp. 205–214.
[34] Y. Xiao, P. Siebert, N. Werghi, A discrete Reeb graph approach for the segmentation of human body scans, in: Proceedings of the 4th International Conference on 3D Digital Imaging and Modeling, Banff, Canada, 2003, pp. 378–385.
[35] S. Xu, X. Yang, A review on interval computation—software and applications, International Journal of Computational and Numerical Analysis and Applications 1 (2002) 149–162.
[36] Y. Zheng-sheng, C. Yao-zhi, O. Min-jae, K. Tae-wan, P. Qun-sheng, An efficient method for tracing planar implicit curves, Journal of Zhejiang University 7 (7) (2006) 1115–1123.