A subdivision arrangement algorithm for semi-algebraic curves: an overview

Julien Wintz, Bernard Mourrain

A subdivision arrangement algorithm for semi-algebraic curves: an overview

Julien Wintz
INRIA Sophia-Antipolis
2004, route des lucioles
06902 Sophia-Antipolis, France
jwintz@sophia.inria.fr

Bernard Mourrain
INRIA Sophia-Antipolis
2004, route des lucioles
06902 Sophia-Antipolis, France
mourrain@sophia.inria.fr

Abstract

We overview a new method for computing the arrangement of semi-algebraic curves. A subdivision approach is used to compute the topology of the algebraic objects and to segment the boundary of regions defined by these objects. An efficient insertion technique is described, which detects regions in conflict and updates the underlying arrangement structure. We describe the general framework of this method, the main region insertion operation and the specializations of the key ingredients for the different types of objects: implicit, parametric or piecewise linear curves.

1. Introduction

Arrangements of geometric objects is a field of computational geometry which has been studied for years [1], initially with simple objects such as line segments [3], circular arcs and curves are still investigated [15, 11, 19, 9] and can be used for computing an arrangement of surfaces [17].

While current methods using a sweep approach [3] focus on events, which are critical points for a projection direction, our method, using a subdivision approach, focuses on regularity criteria and regular regions. When using sweep methods, events are treated when the sweep line encounters points of interest where a projection on a line becomes critical, reducing the dimension of the problem but increasing its computational difficulty (for instance by computing resultants and by lifting points in the case of implicit curves).

On the contrary, a subdivision method avoids the analysis at critical values by enclosing the singular points in a domain from which the problem in hand can be solved. Such methods are less sensitive to numerical approximations of objects and of their intersection points. Their application to arrangement computation has recently emerged such as in [5, 12] where interval arithmetic is used to classify cells in the subdivision process. Subdivision methods are also very efficient for isolating the roots of polynomial equations, which appear in geometric problems [18, 6, 10, 16]. They have also been extended for the approximation of one or two dimensional objects [13, 2, 14].

The new method that we describe in this paper for computing arrangements of semi-algebraic curves, aims at exploiting the power of these methods to localize the zero of polynomials. By storing geometric information on the vanishing of the functions which define the algebraic curves in hierarchical structures, we provide an efficient way to localize intersection points of region boundaries. Its originality is to prevent useless computation by stopping the subdivision as soon as the topology of the object is known in a cell of subdivision.

These considerations have lead us to a generic subdivision arrangement algorithm where input objects considered in a given domain are subdivided until being regular in a cell of subdivision, building a quadtree of cells. From this quadtree, we easily obtain regions which are organized within an augmented influence graph to describe the arrangement of objects. It is dynamic in the sense that we can maintain this structure while new objects are inserted or existing objects are removed. These insertions and deletions involve to be able to compute respectively intersections and unions of regions.

Our algorithm brings several contributions. First, the nature of a subdivision scheme allows to focus, if desired, on regions of interest, leading to a multiresolution arrangement algorithm, parameterized by a level of approximation. Another contribution is that thanks to its generic approach, it can be used to com-

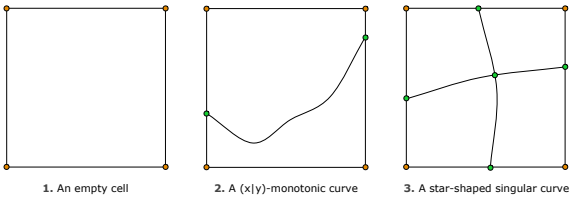

Figure 1. Allowed configurations within a cell.

pute an heterogeneous arrangement, *i.e.* an arrangement of objects having different representations. The resulting regions are also equipped with data structures which lead to efficient operations and are directly usable in a CSG context.

The arrangement algorithm presented in this paper has been completely implemented, tested and validated in an algebraic-geometric modeling environment called Axel[1]. This environment, featuring algebraically defined objects such as implicit or parametric curves or surfaces, allows a visual interaction with computed data structures representing the obtained partition of the space, which can then be queried by various means.

2. Generic arrangement algorithm

The objects that we consider are semi-algebraic curves which representation can either be piecewise linear, parametric or implicit.

The generic arrangement algorithm works incrementally by inserting objects one by one, as follows. For each new object o_k, the regions defined by this new object o_k are computed independently of the others, inserted in the arrangement data structure $\mathcal{A}_{k-1}$ and yields the new data structure $\mathcal{A}_k$.

For more details on the framework of objects, regions, conflicts and associated data structures, see [4].

The *generic* insertion operation of the algorithm makes the assumption that type related functions will be found in a specialization of it. Inserting an object o_k into a structure built for a set $\mathcal{S}_{k-1}$ is handled in four phases.

• **Computing regions.** In this phase, regions are computed from the topology of o_k independently of objects of $\mathcal{S}_{k-1}$.
Regions defined by an object within a given domain cannot be computed directly from its representation. For example, considering the configurations shown in figure 1, we can deduce the topology of regions simply by connecting points of interest of the domain, with points on its border.
When an object is not in one of these configurations, it is subdivided into smaller parts. This process is iterated until one of these configurations is detected. An object in such a configuration is said to be *regular*. The subdivision is then driven by a *regularity test*, which is to be specialized regarding the type of input objects being dependent of the representation of it.
To each object, we associate a quadtree used to keep track of the subdivision process which allows to deduce regions. Once all cells have been processed, its leaves contain sub-regions which union constitutes the regions defined by o_k. To compute this union, these regions are merged traversing the tree from its leaves to its root in a process called *fusion*. The subdivision algorithm ends up with the root node representing the bounding box of o_k containing the regions determined by o_k.

• **Segmenting the boundary.** In this phase, computed regions are equipped with additional data structures to help the introduction in the current arrangement.
To find out the set of regions which conflicts with a new region, we build a structure called the *region segmentation*, a balanced tree data structure in which each node contains the bounding box of an edge defining the boundary of a region, and internal nodes are associated to the union of the boxes of their children.
This structure leads to an efficient algorithm to check whether two regions conflict together and helps dealing with them.

• **Locating conflicts.** In this phase, newly computed regions are checked for conflict with regions defining the current arrangement.
To get the list of conflicts between two regions, we simply compare their segmentations. First, we compare their domain of influence, *i.e.* the roots of their respective segmentations. Second, we compare the boxes associated to internal nodes and proceed to child nodes as long as internal nodes boxes do intersect.
If the resulting list is non empty, intersections of the curve segments are computed using specific intersection methods according to the type of the objects defining the edges in question. If several intersection points are found, we refine the boundary segmentations in order to have at most one intersection per box.

• **Updating regions.** In this phase, conflicts are dealt with, possibly leading to new regions which are inserted in the data structure.
Each conflict between two regions, say r_1 and r_2, refines the arrangement by replacing r_1 and r_2 by $r_1 \cap r_2$, $r_1 \setminus r_2$, and $r_2 \setminus r_1$. We achieve these basic CSG operations by performing a walk-about on the edges of r_1 and r_2, from

[1] http://axel.inria.fr

an intersection point to another, continuing on the left or the right, according to whether we are computing an intersection or a union.
Regions resulting from the intersection are segmented and then inserted in the augmented influence graph as a child nodes of the regions from which they have been computed.

3. Specialization of the algorithm

When computing an arrangement for a given type of curve, we only have to specialize specific functions to meet the generic arrangement algorithm's requirements: computing information on the boundary of a subdivision cell, the subdivision criteria itself, to check if a curve is regular or not within a cell, and computing the intersection of two segments of curves for detecting region conflicts.

We denote by $f_k(x, y) \in \mathbb{R}[x, y]$ the polynomial defining the **implicit curve** corresponding to the object o_k. The specific operations for the arrangement will be performed on the Bernstein representation of f_k (see [8]).

We denote by $x(t)$ and $y(t) \in \mathbb{R}(t)$ the rational functions representing the **parametric curve** corresponding to the object o_k. For sake of simplicity, we will assume that o_k is the image of $[0, 1]$ by the map $\sigma_k : t \mapsto (x(t), y(t))$.

• **Regularity test.** When computing the topology of the regions defined by one given object within a cell, we ensure to be in one of the following configurations: (1) an empty cell (2) a x-monotonic or a y-monotonic cell with exactly two intersection points of the curve with the border of the cell (3) a cell with only one singular point and where the number of branches stemming out from the singular point is exactly the number of intersection points of the curve with the border of the cell (as shown in figure 1).
In the case of **an implicit curve**, we test that the Bernstein coefficients of f_k have no sign change to ensure (1). For (2), we test that these coefficients have at most one sign change on each side of D, and that the coefficients of $\partial_x f$ (resp. $\partial_y f$) have no sign change. For (3), we test that the coefficients of f, $\partial_x f$, $\partial_y f$ both change their sign and that the size of the box is small enough ($< \epsilon$). This step for which we can use a subdivision solver which isolates the singular points of $f(x, y) = 0$ (*e.g.* [16]), allows us to deal safely with "approximate singular points". These tests ensure us that the topology of a curve o_k inside D is uniquely determined from the points of o_k on the boundary of D (see [2] for more details).

In order to obtain such a decomposition for a **parametric curve**, we first partition the interval $[0, 1]$ in intervals where $x'(t) \neq 0$ and $y'(t) \neq 0$ on the interior. On each one of these intervals, the curve is x and y monotonic. The corresponding bounding box of this curve segment is defined by image by σ_k of the end points of the interval. This ensures that (2) is realized. Next, we localize which pairs of bounding boxes of two non-consecutive segments intersect, using the segmentation structure described in Section 2. If we find two such boxes, which are the images of the intervals I and I', we test if there exists $(t, s) \in I \times I'$ with $t \neq s$ such that $x(t) = x(s)$ and $y(t) = y(s)$. This reduces to solving a bivariate polynomial system of polynomial equations. Using subdivision solvers (*e.g.* [16]), we isolate the solution and refine the boundary segmentation by inserting isolating boxes around the image of these points by σ_k. This ensures (3). For (1), we have to test if the cell intersects one of the bounding boxes of one of these boundary curve segments. If so, we test if the curve intersects the boundary of the cell. Otherwise, the cell does not intersect the curve o_k.

• **Regions computation.** To obtain the regions determined by a curve in a regular cell D, we have to isolate their intersection, which determine uniquely the regions defined by o_k in D.
In the case of an **implicit curve**, we use a univariate root solver in the Bernstein basis (*e.g.* [7]), which counts the number of sign changes of the polynomial and subdivides it until this sign variation is 0 or 1, or the size of the interval is smaller than ϵ.
In the case of **a parametric curve**, this can be done easily by solving univariate equations of the form $x(t) = x_0$ or $y(t) = y_0$ using solvers as [7] and by checking that the image by σ_k of these roots is on the border of the cell.
The general scheme when determining a region in a regular cell once these points of o_k on the boundary of the cell have been determined, is to turn around these points on the border in clockwise order while creating edges by connecting these points to points of interest within the cell such as singularities or self-intersection points.

• **Conflict detection.** After the segmentation step, the detection of conflicts reduces either to intersecting regular segments of two different objects or to testing that an endpoint of a regular segment of an object belongs to a given region. To this end, subdivision solvers [18, 6, 16] are used to solve the bivariate polynomial equations f_k, f_l associate to the **implicit curves** in potential conflict in the region of interest.
To compute the intersection points of **two parametric curves** σ_k, σ_l (with $k \neq l$), we solve the bivariate

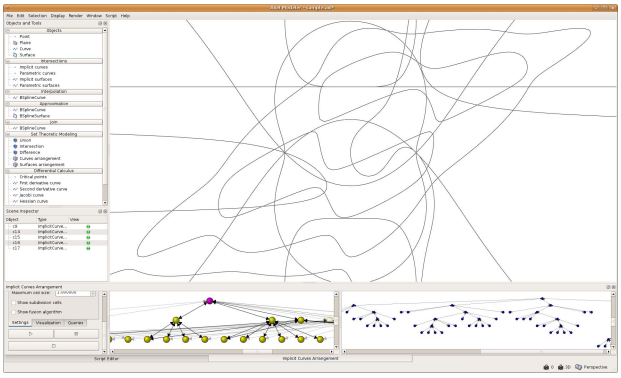

Figure 2. Computing an arrangement in Axel.

system $\sigma_k(t) = \sigma_l(s)$, with t, s in the intervals $\subset [0, 1]$ corresponding to the curve segments. Here again, we can use a subdivision solver like [16] for this purpose.

4. Conclusions

The algorithm presented here has been implemented in the algebraic geometric modeler Axel. We have put a special emphasis on keeping the structure accessible to the user. Using the model-view-controller pattern, data structures such as the augmented influence graph and the various quadtrees can be interactively displayed and queried for point location (see figure 2).

The generic part of the algorithm is currently updated to the higher dimension and specializations are studied to compute arrangements of surfaces with either implicit, parametric or piecewise linear representations.

References

[1] P. Agarwal and M. Sharir. Arrangements and their applications. *Handbook of Computational Geometry (J. Sack, ed.)*, pages 49–119, 2000.

[2] L. Alberti, G. Comte, and B. Mourrain. Meshing implicit algebraic surfaces: the smooth case. In L. S. M. Maehlen, K. Morken, editor, *Mathematical Methods for Curves and Surfaces: Tromso'04*, pages 11–26. Nashboro, 2005.

[3] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, Sept. 1979.

[4] J.-D. Boissonnat and M. Yvinec. *Algorithmic geometry*. Cambridge University Press, New York, NY, USA, 1998.

[5] A. Bowyer, J. Berchtold, D. Eisenthal, I. Voiculescu, and K. Wise. Interval methods in geometric modeling. *gmp*, 00:321, 2000.

[6] G. Elber and M.-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *Proceedings of the sixth ACM Symposium on Solid Modelling and Applications*, pages 1–10. ACM Press, 2001.

[7] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real algebraic numbers: Complexity analysis and experimentations. In *Reliable Implementation of Real Number Algorithms: Theory and Practice*, LNCS. Springer-Verlag, 2007.

[8] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 1993.

[9] E. Fogel, D. Halperin, L. Kettner, M. Teillaud, R. Wein, and N. Wolpert. Arrangements. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 1–66. Springer-Verlag, Mathematics and Visualization, 2006.

[10] J. Garloff and A. P. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. In *Proceedings of the Third World Congress of Nonlinear Analysts, Part 1 (Catania, 2000)*, volume 47, pages 167–178, 2001.

[11] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. CRC Press LLC, Boca Raton, FL, 2004.

[12] Y. Hijazi and T. Breuel. Computing arrangements using subdivision and interval arithmetic. In *To appear in the Proceedings of the Sixth International Conference on Curves and Surfaces Avignon*, 2006.

[13] S. Joon-Kyung, E. Gershon, and K. Myung-Soo. Contouring 1- and 2-Manifolds in Arbitrary Dimensions. In *SMI'05*, pages 218–227, 2005.

[14] C. Liang, B. Mourrain, and J. Pavone. Subdivision methods for 2d and 3d implicit curves. In *Computational Methods for Algebraic Spline Surfaces*. Springer-Verlag, 2006. To appear.

[15] V. Milenkovic and E. Sacks. An approximate arrangement algorithm for semi-algebraic curves. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 237–246, New York, NY, USA, 2006. ACM Press.

[16] B. Mourrain and J.-P. Pavone. Subdivision methods for solving polynomial equations. Technical Report 5658, INRIA Sophia-Antipolis, 2005.

[17] B. Mourrain, J.-P. Técourt, and M. Teillaud. On the computation of an arrangement of quadrics in 3d. *Comput. Geom. Theory Appl.*, (30):145–164, 2005. Special issue, 19th European Workshop on Computational Geometry, Bonn.

[18] E. Sherbrooke and N. Patrikalakis. Computation of the solutions of nonlinear polynomials systems. *Computer Aided Geometric Design*, 10(5):379–405, October 1993.

[19] N. Wolpert. Jacobi curves: Computing the exact topology of arrangements of non-singular algebraic curves. *In ESA 2003, LNCS 2832, pages 532–543, 12*, 2003.