A Hierarchical Structure for Locating Intersections in a Large Sets of B-spline Curves

Eric Guilbert, Eric Saux, Marc Daniel

A Hierarchical Structure for Locating Intersections in Large Sets of B-spline Curves

Eric Guilbert, Eric Saux, and Marc Daniel

Abstract. The automation of maritime chart production at various scales requires at least two important processes: line modeling and generalization. During these processes, line intersections, self-intersections and neighboring lines can be encountered. Such drawbacks must be first detected and finally corrected. As thousands of curves are drawn, efficient approaches must be developed. This paper deals with a method to detect and locate these spatial conflicts for a large set of lines modeled by B-spline curves.

§1. Introduction

Maritime chart production tends to be completely automated: quality, efficiency and ability for producing charts at various scales are expected. The global process is currently a sequence of separate processes which must be respected. From a set of soundings (i.e. depth points), a set of more than one thousand isobathymetric lines (equal depth lines) is obtained. Each line is described by up to two or three thousand points. Our first studies focused on smoothing and compression of these lines with cubic B-splines [11], but a further step is necessary. As a matter of fact, the mapping associated with each given scale of a chart produces non-relevant results as illustrated by Figure 1 on the left. Some curves must be suppressed, aggregated together or simplified so that the final result offers a chart suitable for navigation (Figure 1, right). This very important stage in cartography is called generalization. Several constraints must be respected according to the constraint classification established by Beard (application, graphic, structural, procedural) [2]. Most of them are application

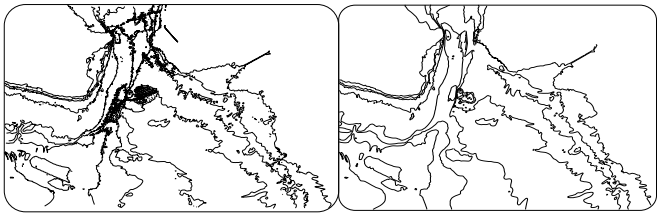

Fig. 1. Isobathymetric lines before (left) and after generalization.

constraints (specific to maritime cartography and navigation safety). In particular, a curve can only be shifted towards higher depths. For our problem, there also exists a very important graphic constraint: the final chart must be legible and a minimal distance (called the legibility distance) should be warranted in between the lines. This distance corresponds to the thickness of a pencil mark (0.2 mm).

An important feature of generalization algorithms is the detection and removal of spatial conflicts. Three types of conflicts can initially be proposed:

- two curves intersect, which is possible since curves are independently smoothed,
- a curve has a self-intersection corresponding to an artifact introduced by its parameterization used for smoothing,
- some curves are too close from each other at the given scale so that they cannot be visually discriminated (graphic constraint of legibility). These "intersections" can be called "visual intersections".

Section 2 proposes a more accurate description of spatial conflicts. They must be detected in large sets of B-spline curves and must be located on the curves (determination of the parametric intervals concerned) so that a final process can remove them by shifting control points or suppressing lines. Thus, a method based on a hierarchical decomposition of the plane is proposed. The curves are shared in different cells in order to locate the possible conflicts as detailed in Section 3. Then, Section 4 explains how the intersections are detected in each cell of the structure. A subdivision scheme is used to compute approximations of the curve segments. Due to the large number of curves, our main objective is to reduce as much as possible the number of computations in order to save time and avoid numerical errors. Results are given in Section 5. Finally, conclusions are reported and we discuss further directions of our forthcoming studies.

§2. Type of Intersections

We assume that isobathymetric lines are obtained from data previously cleaned (superfluous or absurd data are suppressed). The lines are independently smoothed and compressed [11]. Generalization algorithms

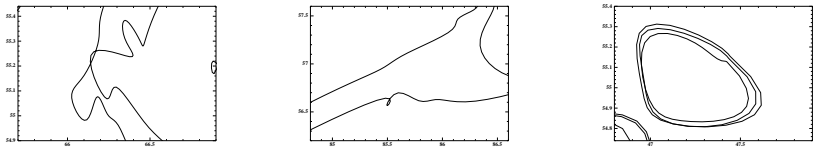

Fig. 2. Isobathymetric lines intersection: contact, micro-loop, closeness.

are then applied on the set of curves to suppress, simplify or aggregate together so that the final result offers good legibility at the given scale of the chart (see Figure 1). These different processes can produce three types of intersection which must be completely removed for a correct generalization because either they are impossible in practice or they produce black spots or thick curves on the map. These types of intersection are illustrated in Figure 2.

The first type of intersection corresponds to a transversal intersection between two curves. Its theoretical limit cases (tangency or local overlapping) are also possible. It is well known that there is a local numerical indetermination: are the two curves tangent (or with a contact of higher order), intersecting more than once or very close together? It can nevertheless be claimed that, within a given tolerance, a local phenomena identified as an intersection has been detected. For our purpose, it corresponds to a drawback on the chart.

The second type of intersection is the occurrence of a self-intersection. This local phenomenon is sometimes encountered when smoothing and is due to the choice of the nodes (chord length, centripetal, etc. [6]). No choice always produces *the* best result even if different solutions have been proposed in the literature. The problem of self-intersection corresponds in fact to the previous type of intersection: the curve segment is split if its control polygon turns through a total angle greater than π [1]. The detection is then processed between the different pieces. The local shape of the cubic curve can also be analyzed through the shape of its control polygon [5].

The third case is the most usual one. Segments of neighboring curves are too close to each other at the given scale so that they cannot be visually discriminated, and thus produce a thick curve. The solution is to suppress some curves, or to locally shift parts of curves according to the application constraint of safety.

A classical problem of intersection implies that both Cartesian and parametric coordinates of intersection points are determined. But our requirement is completely different since an intersection must only be detected in order to be removed. A transversal intersection implies that two curve segments are too close at one point. Moreover, the first and third types of intersection differ in practice from the considered tolerance. Our problem is finally equivalent to detecting curve segments closer than a

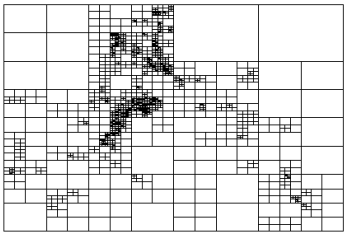

Fig. 3. Segmentation of the initial chart of Figure 1 using a quadtree.

given tolerance. The latter is expressed so that it corresponds to the legibility distance. The following sections present the segmentation process to efficiently apply a test of closeness and how the distances are evaluated.

§3. Hierarchical Decomposition

The hierarchical decomposition principle is based on two simultaneous processes. The first one is a space partitioning method which enables us to define areas where there are potential intersections according to certain criteria. The second one is applied after each partitioning step, and splits the B-spline curve segments into smaller segments related to the new created cells.

3.1. Space partitioning

One can apply either hierarchical uniform or non-uniform segmentations in order to determine areas where there are potential intersections of B-spline curves. The choice of a uniform partitioning like a quadtree [10] depends on the two main objectives of reducing the computation cost and the memory space.

If we assume that the chart containing the initial set of B-spline curves is the root of the quadtree structure, the general step in the segmentation process consists in splitting a cell into four uniform "children" cells according to the following criterion: if a cell contains no more than one curve segment, no intersection can occur in this cell and the segmentation process stops. Otherwise, there is (a) potential intersection(s) and the cell is split. To take into account the visual intersections which might occur on the common edge of two cells, the latter are dilated so that they have a common area. The band width is equal to the legibility distance.

The segmentation process is also stopped when:

- a cell minimal size is reached. The smaller the cell, the more precise the location. However, a cell must always be larger than the legibility distance.
- the curve segments are defined with k (order of the curves) control points from the initial curves. Indeed, if a segment has k control points, it cannot be shared in smaller cells (see subsection 3.2).

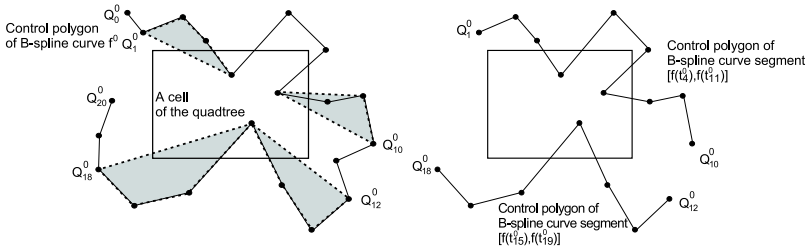

Fig. 4. Illustration of the curve splitting method. Left: gray convex hulls identify the tips of the two curve segments. Right: the control polygons of two B-spline curve segments related to the cell and their corresponding parametric intervals.

This technique has been applied to the initial chart of Figure 1 and leads to the result in Figure 3. The following subsection explains how the B-spline curves are split as a result of the space partitioning.

3.2. Curve splitting

Let us first introduce the following notations. The B-spline curves of the chart are denoted by f^i with $0 \leq i \leq N$ and are of order k. A curve f^i has $m^i + 1$ control points Q^i_j with $0 \leq j \leq m^i$. Its knots are denoted by $(t^i_j)_{j=0}^{j=m^i+k}$. The first and last knots have multiplicity k (clamped knot vector).

Let c be a cell of the quadtree. Let $I \subset \{0, \ldots, N\}$ such that if $i \in I$, then $f^i \cap c \neq \emptyset$. Each of these subscripts $i \in I$ is linked to a list of pairs of indexes $(first, last)$ that identifies the parametric intervals or curve segments $f^i([t_{first}, t_{last}])$ which intersect cell c.

The method is based on a removal of parametric intervals [4]. The interest in this method is that no point is computed on the curve, and no new knot is inserted. Our purpose is to find the smaller parametric interval $[t_{first}, t_{last}]$ that defines the part of the curve intersecting the cell.

Depending on the convex hull property on interval $[t^i_j, t^i_{j+1}]$, curve segment $f^i([t^i_j, t^i_{j+1}])$ is included in the convex hull CH of control points $Q^i_{j-k+1}, \ldots, Q^i_j$. On one hand, if the convex hull CH does not intersect cell c, then $f^i([t^i_j, t^i_{j+1}])$ is outside c and the parametric interval $[t^i_j, t^i_{j+1}]$ can be removed. On the other hand, CH and c intersect, and the control points can be both inside and outside cell c.

Figure 4 illustrates the principle on a cubic B-spline curve f^0 defined on parametric interval $[t^0_0, t^0_{24}]$. The convex hulls in gray identify the tips of the two curve segments which might intersect the cell. The latter are defined on intervals $[t^0_4, t^0_{11}]$ and $[t^0_{15}, t^0_{19}]$. Thus, the cell of Figure 4 is linked to set $I = \{0\}$ (i.e. the index of the B-spline curve), where 0 is related to the list $\{(4, 11), (15, 19)\}$.

As a result, if we assume that:

1) the initial chart is the root cell c,
2) I contains the indexes of all initial B-spline curves f^i,
3) each index $i \in I$ is linked to a pair of indexes $(k-1, m^i+1)$ (i.e. f^i being defined on parametric interval $[t^i_{k-1}, t^i_{m^i+1}]$),

the space partitioning method described in Section 3.1 can be coupled with this curve splitting method.

At the end of this hierarchical decomposition process, a quadtree is defined, where each cell contains a list (empty or not) of B-spline curve segments. There are potential intersections when there are at least two curve segments in the same cell. The method described in the next section is then applied.

§4. Detection of Intersections

The aim is to detect the intersections between different curve segments in a cell. We propose an efficient and reliable method suitable for the types of intersections described in Section 2 using the fewest number of computations. Our approach is based on geometrical methods instead of numerical methods [12]. The method is divided into two parts. Firstly, the curve segments are approximated by polygonal lines. Secondly, the intersections between these lines are computed. In the following, we consider cubic B-spline curves with uniform knot vectors with clamped ends for simplicity. This study has been extended with more general knot vectors and different subdivision schemes.

4.1. Approximation of curve segments

The approximation algorithm consists in building a tight linear envelope for the curve segments by applying a subdivision scheme [8]. Then, for each segment, an approximating polygonal line is deduced from this envelope [9]. It is defined from control points and points on the segment to be approximated. Let

$$\Delta_2 Q_j = Q_{j-1} - 2Q_j + Q_{j+1}, \quad 0 < j < m, \tag{1}$$

be the second differences of control polygon Q of curve f, and $l(\zeta)$ be the piecewise linear interpolant of the control points at Greville abscissae $\zeta_j = j + \frac{k}{2}$, i.e. $l(\zeta_j) = Q_j$. On an interval $[\zeta_j; \zeta_{j+1}]$, Lutterkort and Peters [8] prove that the difference between $l(\zeta)$ and $f(\zeta)$ is bounded by

$$\|f - l\| \leq \frac{k}{24} \max_{j-\frac{k-1}{2}+1 \leq i \leq j+\frac{k-1}{2}-1} \|\Delta_2 Q_i\|_2. \tag{2}$$

For cubic curves, we have the equality at each Greville abscissa ζ_i, so that $f(\zeta_j) = l(\zeta_j) + \frac{1}{6}\Delta_2 Q_j$. Thus, a tight envelope is built with points $f(\zeta_j)$ and $l(\zeta_j)$ (see Figure 5) [8].

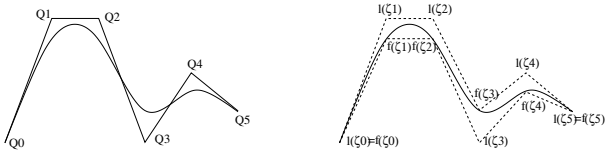

Fig. 5. Tight envelope of a B-spline curve.

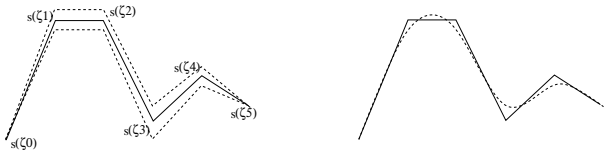

Fig. 6. Midpath of an envelope.

A more accurate envelope can be computed by applying a subdivision scheme. In our case, the following scheme is used [8]:

$$Q_{2j}^{(n+1)} = \frac{1}{2}(Q_j^{(n)} + Q_{j+1}^{(n)}), \qquad Q_{2j+1}^{(n+1)} = \frac{1}{8}Q_j^{(n)} + \frac{3}{4}Q_{j+1}^{(n)} + \frac{1}{8}Q_{j+2}^{(n)}, \quad (3)$$

where n is the subdivision step. At each step of the subdivision scheme, the second differences of the new polygon are bounded by the second differences of the previous polygon:

$$\max_j \|\Delta_2 Q_j^{(n+1)}\|_2 = \frac{1}{4} \max_j \|\Delta_2 Q_j^{(n)}\|_2. \quad (4)$$

Equation (5) yields an a priori estimate on the number of subdivisions needed to bring the curve segment and control polygon within a given distance ϵ [8]:

$$n = \log_4 \frac{k \max_j \|\Delta_2 Q_j\|_2}{24\epsilon}. \quad (5)$$

4.2. Locating intersections

Based on the previous method, an envelope is computed for each curve in a cell, and the intersections between these envelopes are located. To reduce the number of computations, we construct polygonal lines called "midpaths" [9] passing through the middle of each envelope (see Figure 6) and finally compare these lines. The piecewise linear function $s(\zeta)$ defining a midpath is obtained with the values $s(\zeta_j) = \frac{1}{2}(f(\zeta_j) + l(\zeta_j))$, $0 \le j \le m$.

The midpath is used as an approximation of the curve segment. The error ϵ_{num} is half the width of the envelope. Two curve segments intersect if the distance between their envelopes is less than the legibility

distance ϵ_{vis}. So these segments intersect if Hausdorff distance ϵ_{mid} between their midpaths is less than $\epsilon_{vis} + 2\epsilon_{num}$. The error ϵ_{num} must be small compared to the legibility distance ϵ_{vis}; otherwise, we will have too large approximations of the segments, and too many intersections.

When an intersection is detected, one has to find the corresponding parametric intervals of each segment in order to remove the conflict. As the number of steps in the subdivision scheme is known, the control points of original curve segments which are involved in the conflict can be identified.

§5. Results

The method has been implemented and tested (Pentium IV 1GHz) on several data sets containing isobathymetric lines. The legibility distance ϵ_{vis} is set to 0.2 mm. We focus here on two charts: chart of Figure 1 (900 curves with 30000 points) and a second chart (2000 curves with 80000 points). Table 1 proposes results of processing time and number of intersections depending on the quadtree depth. Percentages of time spent in detecting the intersections (i.e. process of Section 4) compared with the whole processing time (i.e. processes of Sections 3 and 4) are given.

	Chart 1			Chart 2		
Depth	Time (s)	Ratio	Inters.	Time (s)	Ratio	Inters.
4	12.36	94%	996	20.28	90%	562
5	8.65	85%	1046	15.12	77%	585
6	8.16	75%	1239	15.69	62%	629
7	9.17	68%	1556	20.80	52%	714
8	11.73	62%	2241	30.71	49%	828

Tab. 1. Results for different quadtree depths ($\epsilon_{num} = 0.02$ mm).

	Chart 1			Chart 2		
ϵ_{num} (mm)	Time (s)	Ratio	Inters.	Time (s)	Ratio	Inters.
2.10^{-3}	41.61	95%	1032	57.55	90%	511
5.10^{-3}	17.20	88%	1089	29.00	79%	557
10^{-2}	12.86	84%	1149	15.95	73%	597
2.10^{-2}	8.16	75%	1239	15.69	62%	629
5.10^{-2}	6.06	66%	1414	12.65	51%	729

Tab. 2. Results for different approximation errors ϵ_{num} (tree depth=6).

We notice that the number of intersections increases when the tree is deeper. Best times are obtained at levels 5 and 6. On one hand, shallow trees (e.g. level 4) produce large curve segments: much time is spent in the detection process. On the other hand, deeper trees (e.g. levels 7 or 8) lead to smaller cells: computations in the detection process are reduced, but redundant curve segments are created.

For a given tree depth (e.g. 6), Table 2 illustrates the effect of ϵ_{num} on time and number of intersections. With a higher accuracy, less visual intersections are detected. However, more subdivisions are done and segments are defined with more points entailing more computations.

As a result, a tuning of the quadtree depth and the accuracy ϵ_{num} must be found to detect all the conflicts in a reasonable time. Contrary to ϵ_{num}, the depth variation has more effect on the number of intersections than on computing time. Further results are required to propose a global solution: computing times for merging the parametric intervals corresponding to an intersection and for correcting the curves should be quantified. Nevertheless, choosing a large ϵ_{num} reduces time but overestimates the number of conflicts and a thin segmentation of the plane involves an accurate location of each conflict and of its environment.

§6. Conclusion and Forthcoming Studies

In this paper, we have introduced a new method for locating intersections in large sets of B-spline curves. It can deal with different types of intersections such as transversal intersections, overlapping and also "visual intersections". The detection requires two stages. The first one segments the chart using a quadtree and the curve segments are divided into different cells. The second stage computes intersections by approximating the curve segments with polygonal lines.

The interest in our approach is twofold. Firstly, the number of computations is reduced. The curves are not modified and the segmentation is processed by comparing the point coordinates. Therefore, the method is fast and can be applied to large set of curves. Secondly, intersections are deduced from the distance between the segments so that all kinds of intersections are detected with the same process. Thus, the method can be used for complex problems such as map generalization.

After the detection stage described here, our next study will concern their correction by locally shifting the control points. For that purpose, one has to take into account the neighborhood of the curves and the cartographic constraints. In order to identify the neighboring segments, the quadtree and the links between the cells are needed.

The strategies used to deform the curves are mainly based on energy minimization. Internal energies are defined from the derivatives, whereas external energies depend on the proximity of other curves. The system is balanced when its energy is minimal. Snakes [3] and mechanical deformations [7] can be used. The difficulty is to control the displacement of the curves: it should avoid creating new conflicts in the neighboring areas.

References

1. Andersson L. E., T. J. Peters and N. F. Stewart, Self-intersection of composite curves and surfaces, Comput. Aided Geom. Design **15** (1998), 507–527.

2. Beard K., Constraints on rule formation, in *Map Generalization*, B. Buttenfield and R. B. McMaster (eds.), 1991, 121–135.
3. Burghardt D. and S. Meier, Cartographic displacement using the snakes concept, in *Semantic Modeling for the Acquisition of Topographic Information from Images and Maps*, W. Förstner and L. Plümer (eds.), 1997, 59–71.
4. Daniel M., A curve intersection algorithm with processing of singular cases: introduction of a clipping technique, in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche and L. L. Schumaker (eds.), Academic Press, New York, 1992, 161–170.
5. Kantorowitz E. and Y. Schechner, Managing the shape of planar splines by their control polygons, Computer-Aided Design **25** (1993), 355–364.
6. Lee E. T. Y., Choosing nodes in parametric curve interpolation, Computer-Aided Design **21** (1989), 363–370.
7. Léon J. C. and P. Trompette, A new approach towards free-form surfaces control, Comput. Aided Geom. Design **12** (1995), 395–416.
8. Lutterkort D. and J. Peters, Linear envelopes for uniform B-spline curves, in *Curve and Surface Fitting: Saint-Malo 1999*, Albert Cohen, Christophe Rabut, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000, 239–246.
9. Peters J. and X. Wu, Optimised refinable surface enclosures, Technical report, University of Florida, 2001.
10. Samet H., *Application of spatial data structures*, Addison Wesley, 1990.
11. Saux E. and M. Daniel, Data reduction of polygonal curves using B-splines, Computer-Aided Design **31** (1999), 507–515.
12. Sederberg T. and S. R. Parry, Comparison of three curve intersection algorithms, Computer-Aided Design **18** (1986), 58–64.

Eric Guilbert and Eric Saux
Institut de Recherche de l'Ecole Navale, Groupe SIG
Ecole Navale, Lanvéoc-Poulmic, B.P. 600, 29240 Brest-Naval, France
{guilbert;saux}@ecole-navale.fr

Marc Daniel
Laboratoire des Sciences de l'Information et des Systèmes
ESIL, Campus de Luminy, case 925, 13288 Marseille cedex 9, France
Marc.Daniel@esil.univ-mrs.fr