Sweeping and Maintaining Two-Dimensional Arrangements on Surfaces*

Eric Berberich† Efi Fogel‡ Dan Halperin‡ Ron Wein‡

Abstract

We introduce a general framework for processing a set of curves defined on a continuous two-dimensional parametric surface, while sweeping the parameter space. A major goal of our work is to maximize code reuse in implementing algorithms that employ the prevalent sweep-line paradigm, and consequently to minimize the effort needed to extend the implementation of the paradigm to various surfaces and families of curves embedded on them. We show how the sweep-line paradigm is used to construct an arrangement of curves embedded on an orientable parametric surface, and explain how the arrangement package of CGAL, which previously handled only arrangements of bounded planar curves, is extended to handle curves embedded on a general surface. To the best of our knowledge, this is the first software implementation of generic algorithms that can handle arrangements on general parametric surfaces.

1 Introduction

We are given a surface S in $\mathbb{R}^3$ and a set $\mathcal{C}$ of curves that all lie on this surface. The *arrangement* of the curves of $\mathcal{C}$, denoted $\mathcal{A}(\mathcal{C})$ is the subdivision these curves induce on the surface S into cells of dimension 0 (*vertices*), 1 (*edges*) and 2 (*faces*).

CGAL, the Computational Geometry Algorithms Library,[1] is the product of a collaborative effort of several sites in Europe and Israel, aiming to provide a generic and robust, yet efficient, implementation of widely used geometric data structures and algorithms. The arrangement package [9] included in the latest public release of CGAL (Version 3.2) is capable of constructing and maintaining planar arrangements of bounded curves. That is, the surface S is the xy-plane, and all curves in $\mathcal{C}$ are bounded. When working with unbounded curves, users are required to properly clip them as a preprocessing step, so that no essential information about the arrangements (e.g., a finite intersection point) is lost. However, this solution is insufficient for some applications. For example, it is possible to represent the minimization diagram of a set of surfaces in $\mathbb{R}^3$ as a planar arrangement, where each face is labeled with the surface that induces the lower envelope over that face [8]. As an arrangement of bounded curves has only a single unbounded face, it is impossible to represent the minimization diagram of a set of unbounded surfaces, where several unbounded faces might be required.

We have recently enhanced the arrangement package to support planar arrangements of unbounded curves. This extension will be included in the forthcoming release of CGAL (Version 3.3). The same principles we used for handling unbounded curves, or more precisely curve-ends that lie at infinity, can be nicely generalized for the case of a set of curves embedded on a parametric surface. An orientable *parametric surface* S is a surface defined by parametric equations involving two parameters u and v, namely: $f_S(u,v) = (x(u,v), y(u,v), z(u,v))$. Thus, $f_S : \mathbb{P} \longrightarrow \mathbb{R}^3$ and $S = f_S(\mathbb{P})$, where $\mathbb{P}$ is a continuous and simply connected two-dimensional parameter space. The general case is currently implemented as a prototypical package in CGAL.

Related work: Effective algorithms for manipulating arrangements of curves have been a topic of considerable interest in recent years, with an emphasis on exactness and efficiency of implementation [5]. Mehlhorn and Seel [7] propose a general framework for extending the sweep-line algorithm to handle unbounded curves. Note that they do not address the case of surfaces other than the plane. Andrade and Stolfi [1] develop exact algorithms for manipulating circular arcs on a sphere. Cazals and Loriot [4] compute exact arrangements of circles on a sphere. Halperin and Shelton [6] incrementally construct arrangements of circles on a sphere, using floating-point arithmetic and assuming general position. The latter two works are motivated by molecular modeling.

2 Sweeping on Surfaces

Recall that the main idea behind the Bentley-Ottmann sweep-line algorithm [2] is to sweep a vertical line starting from $x = -\infty$ onward and maintain the set of x-monotone curves that intersect it. These curves are ordered according to the y-coordinate of their intersection with the vertical line and stored in a balanced search tree named the *status structure*. The

*This work has been supported in part by the IST Programme of the EU as Shared-cost RTD (FET Open) Project under Contract No IST-006413 (ACS - Algorithms for Complex Shapes), by the Israel Science Foundation (grant no. 236/06), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

†Max-Planck-Institut für Informatik, Saarbrücken, Germany, eric@mpi-inf.mpg.de .

‡School of Computer Science, Tel-Aviv University, Israel, {efif,danha,wein}@post.tau.ac.il .

[1] http://www.cgal.org/ .

contents of the status line change only at a finite number of *event points*, where an event point may correspond to a curve endpoint or to an intersection of two curves. The event points are sorted in ascending xy-lexicographic order and stored in an *event queue*. This event queue is initialized with all curve endpoints, and is updated dynamically during the sweep process as new intersection points are discovered.

2.1 Augmenting the Parameter Space

Our goal is to study the subdivision induced on a parametric surface by sweeping over its parameter space. However, to conveniently do so, we must consider a subspace of $\mathbb{P}$. Sweeping over the entire parameter space raises, in general, several difficulties either when the parameter space is unbounded, or when there is no inverse mapping from the surface to the parameter space. We eliminate these difficulties by cutting out portions of the parameter space and symbolically keeping track of these modifications.

We next formally define three aspects that require special attention when generalizing the sweep procedure. In all cases, S is a parametric surface defined over $\mathbb{P}$ in the uv-plane. We give the definitions using the u-parameter; the definitions with respect to the v-parameter are similar.

Definition 1 (Infinite boundary:) *Let $\hat{u}$ be one of the values defining the u-range of $\mathbb{P}$ ($\hat{u}$ may be finite or $\hat{u} = \pm\infty$). We say that the surface has an* infinite boundary *in u if: $\forall v \ \lim_{u \to \hat{u}} f_S(u, v) = \pm\infty$.*

Definition 2 (Curve of discontinuity:) *If u is defined over a bounded parameter range $[u_{\min}, u_{\max})$ such that: $\forall v \ \lim_{u \to u_{\max}} f_S(u, v) = f_S(u_{\min}, v)$, then the curve defined by $f_S(u_{\min}, v)$ forms a* curve of discontinuity *in u on the surface S.*

Definition 3 (Singularity point:) *A point $p_0 = f_S(u_0, v_0) \in S$ is a* singularity point *in u if u_0 is either $u_{\min}$ or $u_{\max}$, and for each $\delta > 0$ we have: $\forall v \ \exists u \ \|f_S(u, v) - p_0\| < \delta$.*

The xy-plane (see Fig. 1(a)), for example, has an infinite boundary in the minimal and the maximal values of u *and* in the minimal and maximal values of v. A canonical 3D cylinder of radius r (see Fig. 1(b)), parameterized for $\mathbb{P} = [-\pi, \pi) \times \mathbb{R}$ such that $f_S(u, v) = (r \cos u, r \sin u, v)$, contains a line of discontinuity that is parallel to the z-axis and passes through $(-r, 0, 0)$. The unit sphere (see Fig. 1(c)), parameterized over $\mathbb{P} = [-\pi, \pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ using $f_S(u, v) = (\cos u \sin v, \sin u \sin v, \cos v)$, contains a semicircle of discontinuity that connects the two poles $(0, 0, -1)$ and $(0, 0, 1)$ through $(-1, 0, 0)$. In addition, the two poles are singularity points in v.

Given a surface containing curves of discontinuity and singularity points we modify the parameter space as follows: In case of discontinuity in u, we consider the open u-range $(u_{\min} + \varepsilon, u_{\max} - \varepsilon)$ for an infinitesimally small $\varepsilon > 0$. In case of a singularity point in $u_{\min}$ we augment the u-parameter range to be lower bounded by $u_{\min} + \varepsilon$ (or upper bounded by $u_{\max} - \varepsilon$ in case of a singularity point in $u_{\max}$), for an infinitesimally small $\varepsilon > 0$. We handle singularities in v in a similar fashion. As a result, we obtain an augmented parameter space $\tilde{\mathbb{P}}$, for which it is possible to define the inverse mapping $f_S^{-1} : \mathbb{R}^3 \longrightarrow \tilde{\mathbb{P}}$.

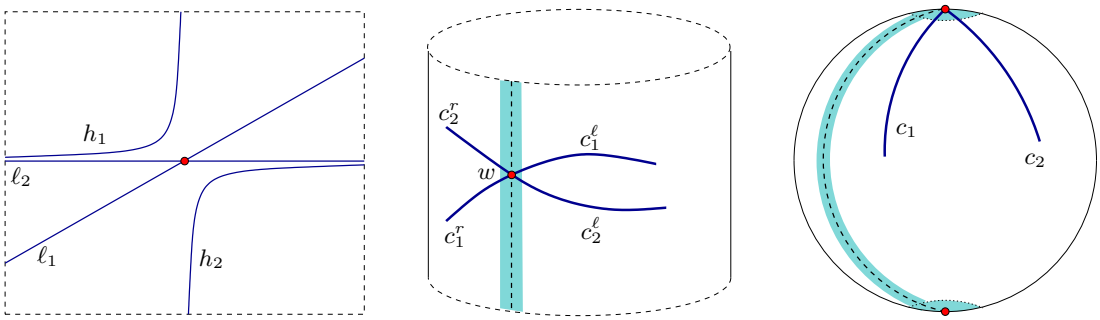
It is now possible to apply an augmented sweep-line algorithm to our parametric surface, where we actually perform a plane sweep over $\tilde{\mathbb{P}}$. Let $\tilde{S}$ denote the image of the augmented parameter space, namely $f_S(\tilde{\mathbb{P}})$. Given a set $\mathcal{C}$ of curves defined on S, we start by computing $C' = C \cap \tilde{S}$ for each $C \in \mathcal{C}$, and by subdividing C' into u-monotone subcurves. We refer to the resulting subcurves as *sweepable curves*. Note that in particular, the interior of a sweepable curve cannot intersect a curve of discontinuity or contain a singularity point. However, the curve-ends may be incident to the modified surface boundaries.

We start the sweep with the curve $f_S(u_0, v)$, for some initial fixed u-value u_0 (e.g., $u_0 = u_{\min} + \varepsilon$ in the example of the cylinder). We now sweep the curve over the surface $\tilde{S}$. For each u-value u', a subset of the sweepable curves induced by $\mathcal{C}$ intersect the sweep-curve $f_S(u', v)$, at the points $p_1, \ldots, p_k \in S$. The status structure stores these curves ordered in ascending v of $f_S^{-1}(p_1), \ldots, f_S^{-1}(p_k)$. Similarly, when we detect an intersection point p, we insert it into the event queue, considering the lexicographic uv-order of $f_S^{-1}(p)$. The event queue must contain events associated with curve-ends incident to the surface boundaries for its proper maintenance.

2.2 Sweeping Unbounded Curves

The main difficulty in adapting the Bentley-Ottmann sweep algorithm [2] to the unbounded case, lies in handling its initialization step, and symmetrically in completing the sweep after all the finite event points were encountered. Let us revise the terminology used so far. Instead of considering the endpoints of a curve, we refer to the two *curve-ends*. A curve-end may be unbounded or bounded, and only in the latter case we have a valid endpoint. In order to perform the sweep-line procedure, we require the two following comparisons involving unbounded curve-ends (in addition to the operations listed in [5] for bounded curves): (i) determine the relative vertical position of two curve-ends defined at $x = \pm\infty$,[2] and (ii) determine the relative horizontal positions of two curve-ends with finite x-coordinates that lie at $y = \pm\infty$.

[2] For two lines this amounts to comparing their slopes, and in case of equality we can compare their vertical position at $x = 0$. Other curves may require more careful analysis.



(a) (b) (c)

Figure 1: Comparing curve-ends with boundary conditions: (a) Comparing at infinity. (b) Comparing near the line of discontinuity. (c) Comparing near a singularity point.

Having defined the geometric primitives, we are ready to modify the sweep-line algorithm to handle infinite curves. First, we store extra information with the events: an event may be associated with a (finite) point, or it may be associated with an unbounded curve-end at $x = \pm\infty$ or at $y = \pm\infty$. We begin the sweep process by constructing events that represent all unbounded and bounded curve-ends. To sort these events we use a simple procedure based on the two primitive comparison operations listed above: if one event lies at $x = -\infty$ and the other is a (finite) point, then the first event is obviously smaller; if both events lie at $x = -\infty$ we compare their associated curve-ends there, etc. For instance, in Fig. 1(a) we have $\ell_1 < \ell_2 < h_1$ when the sweep is initialized. We omit further details of the process in this abstract, and remark that whenever infinity in x or in y is involved, barely any geometric operations are required.

2.3 Sweeping on General Surfaces

We can now generalize the sweep-line procedure for sweeping over curves embedded on a surface in $\mathbb{R}^3$. So far we swept over the parameter space $\mathbb{P} = \mathbb{R}^2$, and treated curve-ends that coincide with the infinite boundaries symbolically. We can use the same set of geometric primitives for sweeping over a set of curves on a surface. However, we have to re-interpret the geometric predicates as if they are given on the uv-plane. For instance, instead of comparing two points p_1 and p_2 by their xy-lexicographic order, we compare $f_S^{-1}(p_1)$ and $f_S^{-1}(p_2)$ according to their uv-lexicographic order in $\tilde{\mathbb{P}}$.

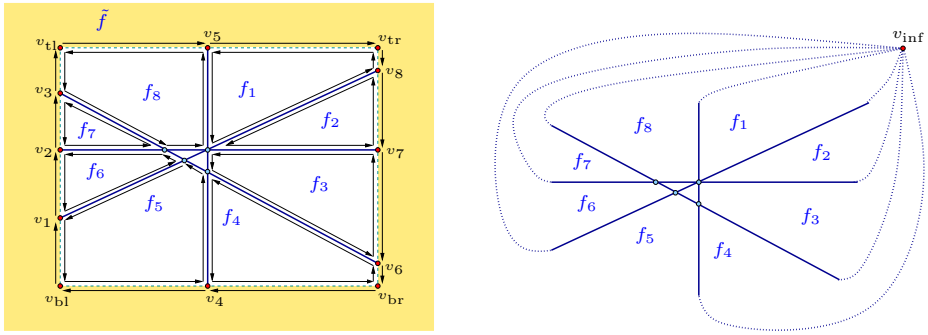
A sweepable curve-end may have *boundary conditions*. In the previous subsection we have already encountered curves with unbounded ends, and we say that the boundary condition of such an end in x (or in y) is of type *minus infinity* or *plus infinity*. In the general case, we may also encounter curve-ends whose boundary condition is *leaving discontinuity* (or *approaching discontinuity*), or *leaving singularity* (or *approaching singularity*). For instance, in the example depicted in Fig. 1(b), all sweepable curve-ends may start right after the line of discontinuity or may end right before this line, as we have removed the line of discontinuity from $\tilde{S}$. The two curves C_1 and C_2 are split at the line of discontinuity, forming the sweepable curves c_1^ℓ, c_1^r and c_2^ℓ, c_2^r, respectively. Yet when we compare the curve-ends we consider an ε-neighborhood around the line of discontinuity (shaded). Thus, c_1^ℓ is above c_2^ℓ after the line of discontinuity, when the sweep starts. Fig. 1(c) shows how we symbolically handle curve-ends that are incident to a singularity point (the north pole of a sphere in this case): c_1 lies to the left of c_2, as we compare the ends of sweepable curves in an ε-neighborhood below the north pole (shaded). Note that this means that we have a different event for every curve-end that coincides with a pole.

3 Constructing Arrangements on Surfaces

Constructing an arrangement of curves on a parametric surface boils down to properly handling the subcurves the sweep-line procedure detects and inserting them into the doubly-connected edge-list (DCEL for short) that represents the arrangement; see, e.g., [3, Chap. 2]. As the only modification of the sweep-line algorithm involves curve-ends with boundary conditions, we have to augment the curve-insertion procedures to properly handle such curve-ends.

Already when moving to unbounded curves we should consider a representation of the arrangement that caters for more than one unbounded face. Fig. 2(a) demonstrates one possibility, where we use an implicit bounding rectangle embedded in the DCEL structure using *fictitious* edges that are not associated with any concrete planar curve. It is also possible to choose a different representation of a planar arrangement of bounded curves that uses a single vertex at infinity v_{inf}, such that all unbounded curve-ends are incident to this vertex; see illustration in Fig. 2(b).

Aiming for modularity, we wish to decouple the implementation of the basic arrangement operations (e.g., inserting a new edge associated with a subcurve, removing an edge, etc.)



(a) (b)

Figure 2: Possible DCEL representations of an arrangement of four lines in the plane.

from the actual representation of the arrangement. We do this by introducing the class-template `Arrangement_on_surface_2<GeomTraits, TopTraits>`, which should be instantiated by two types. The first is the *geometry-traits class*, which defines the family of curves that induce the arrangement, and encapsulates all primitive geometric predicates and constructions (e.g, comparing two points by their uv-lexicographic order, computing intersection points, etc.) on curves of this family. The second type is a *topology-traits class*, which encapsulates the topology of the surface on which the arrangement is embedded, and determines the underlying DCEL representation of the arrangement. It does so by supplying predicates and operations related to curve-ends with boundary conditions. For example, it is responsible for initializing a DCEL structure that represents an empty arrangement, and for locating the DCEL feature that represents a given curve-end (this feature may be a fictitious edge as in Fig. 2(a), a vertex at infinity as in Fig. 2(b), etc.). Using the topology-traits primitives, we can use the sweep-line procedure to construct the arrangement of a set of curves on a surface: When we detect a subcurve with boundary conditions, we query the topology-traits class to obtain the DCEL feature containing the curve-end, then insert the subcurve accordingly. For example, if we sweep over the cylinder depicted in Fig. 1(b), a vertex w is created on the line of discontinuity when we insert c_1^ℓ into the arrangement. The topology-traits class keeps track of this vertex, so it will associate w as the minimal end of c_2^ℓ and as the maximal end of c_1^r and c_2^r. Similarly, in the example shown in Fig. 1(c), the north pole will eventually be represented as a single DCEL vertex, with c_1 and c_2 incident to it.

We have already implemented a topology-traits class for handling unbounded curves on the plane, along with geometry-traits classes for handling lines and rays, and with EXACUS[3] based geometry-traits classes for algebraic curves. We have also designed and implemented two other topology-traits classes along with corresponding geometry-traits classes, that define curves on surfaces: the first maintains arrangements of arcs of great circles embedded on a sphere, and the other constructs arrangements of intersection curves between quadric surfaces embedded on a quadric surface. For lack of space, we omit the implementation details.

[3] `http://www.mpi-sb.mpg.de/projects/EXACUS/` .

References

[1] M. V. A. Andrade and J. Stolfi. Exact algorithms for circles on the sphere. *Internat. J. Comp. Geom. Appl.*, 11(3):267–290, 2001.

[2] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. on Computers*, 28(9):643–647, 1979.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, Germany, 2nd edition, 2000.

[4] F. Cazals and S. Loriot. Computing the exact arrangement of circles on a sphere, with applications in structural biology. Technical Report 6049, INRIA Sophia-Antipolis, 2006.

[5] 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*, chap. 1, pages 1–66. Spinger, 2006.

[6] D. Halperin and C. R. Shelton. A perturbation scheme for spherical arrangements with application to molecular modeling. *Comput. Geom. Theory Appl.*, 10:273–287, 1998.

[7] K. Mehlhorn and M. Seel. Infimaximal frames: A technique for making lines look like segments. *Internat. J. Comp. Geom. Appl.*, 13(3):241–255, 2003.

[8] M. Meyerovitch. Robust, generic and efficient construction of envelopes of surfaces in three-dimensional space. In *Proc. 14th Europ. Sympos. Alg.*, pages 792–803, 2006.

[9] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. Advanced programming techniques applied to CGAL's arrangement package. *Comp. Geom. Theory Appl.* To appear.