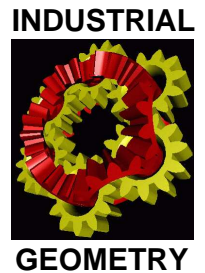


Forschungsschwerpunkt S92

# Industrial Geometry

<http://www.ig.jku.at>



FSP Report No. 63

## On Computing the Convex Hull of (Piecewise) Spherical Objects

F. Aurenhammer, B. Jüttler

Dezember 2007

---

**FWF**

Der Wissenschaftsfonds.



**TU**  
Graz  
Graz University  
of Technology

# On Computing the Convex Hull of (Piecewise) Spherical Objects\*

FRANZ AURENHAMMER

*Institute for Theoretical Computer Science*  
*University of Technology, Graz, Austria*  
auren@igi.tugraz.at

BERT JÜTTLER

*Institute for Applied Geometry*  
*Johannes Kepler University, Linz, Austria*  
bert.juettler@jku.at

## Abstract

We utilize support functions to transform the problem of constructing the convex hull of a finite set of spherical objects into the problem of computing the upper envelope of piecewise linear functions. This approach is particularly suited if the objects are (possibly intersecting) circular arcs in the plane or spheres in three-space.

## 1 Introduction

Computing the convex hull of a set of geometric objects is a basic task in computational geometry, with far-reaching applications. Maybe the most fascinating aspect of the convex hull problem is that many different construction techniques can be used for its solution. In the case of finite point sets in the Euclidean plane  $\mathbb{R}^2$ , practically all standard algorithmic paradigms lead to convex hull computations that are efficient in one or the other respect. We refrain from bibliographic details as many of the papers cited below contain introductory surveys on this topic. In particular, the article [12] also gives references on lower bounds for the computational complexity of the planar convex hull problem.

Despite its importance in practical applications, the convex hull problem for objects more general than points remained less investigated. Clearly, if the objects are polygonal, then applying to their total set of vertices any convex hull algorithm for finite point sets will do. Curved objects, however, have to be treated differently. Their convex hull will, in general, contain vertices that have not been part of the input. In fact, even for relatively simple objects in the plane, the number of vertices of their convex hull may be superlinear in the number of objects.

As one possible approach, the Voronoi diagram of the  $n$  given objects can be computed in a first step, and the convex hull then extracted from those among the objects whose Voronoi regions are unbounded. This approach is efficient only for interior-disjoint planar objects, where  $O(n \log n)$  time algorithms for computing such Voronoi diagrams exist [23, 3], and even then is inherently involved. A theoretically interesting alternative that also works for intersecting objects is the output-sensitive  $O(nf \log h)$  algorithm in [15], based on the marriage-before-conquest paradigm. Here  $h$  denotes the size (number of vertices) of the convex hull, and  $f$  is a slowly growing function depending on  $h$  and the size of the convex hull of object pairs. For inputs forming the boundary of a single planar shape (a simply connected object in the

---

\*Research supported by the FWF Joint Research Project 'Industrial Geometry', S9205-N12, S9202-N12.

plane), several  $O(n)$  time algorithms have been devised, for piecewise smooth Jordan curves [20], algebraic curves [5], and circular arcs [7, 2].

The case of circular arcs may deserve particular attention. First of all, more general curved objects can always be approximated, mostly in an easy way, with arbitrary accuracy by circular arcs [13, 16]. Also, to achieve a given accuracy, their number is significantly less than in any straight line segment approximation (faceting); this advantage is made use of in a systematic way in [2]. Finally, circular arcs are still reasonably easy to deal with algorithmically.

In view of these facts, algorithms that specialize on computing the convex hull of a finite set of circular arcs in the plane are surprisingly rare. In the paper [2], Melkman's  $O(n)$  time algorithm [14] is generalized for circular arcs, but the method applies only to situations where the input forms a single noncrossing chain. Generalizing Jarvis' march [11] to sets of circular arcs is proposed and detailed in [22]. Their algorithm, though intended for noncrossing arcs, also works in the general case. It runs in  $O(nh)$  time and thus is theoretically inferior to the output-sensitive  $O(nf \log h)$  algorithm in [15] which, in turn, is much less practical. Another fact that comes as a surprise is that the maximum output size,  $h_m$ , is still unsettled in the case of circular arcs.

In this note, we describe a simple  $O(h_m \log n)$  method for computing the convex hull of  $n$  circular arcs in  $\mathbb{R}^2$ . No restrictions are drawn on the placement of these arcs. The input thus may stem from approximating any set of (possibly intersecting) curved objects in the plane, be they shapes or curves. We also show that  $h_m = O(\lambda_4(n))$  and  $h_m = \Omega(\lambda_3(n))$ , where  $\lambda_i(n)$  denotes the maximum length of an  $n$ -symbol Davenport-Schinzel sequence [21] of order  $i$ . As an algorithmic byproduct, the intersection of two circularly bounded convex shapes can be computed in a simple way in  $O(n \log n)$  time (without using data structure consuming sweep techniques.) We use support functions to transform the problem into one easier to handle, exploiting the fact that support functions of circular objects are linearly embeddable. The method in principle works for spherical objects in higher dimensions. For example, it leads to a worst-case optimal algorithm for computing the convex hull of  $n$  spheres in  $\mathbb{R}^3$ , in a way more direct than the method used in [6]. Support functions are a standard concept in convex geometry [10], and a versatile tool in computer vision, graphics, and image processing [8]. For example, they find application in the reconstruction of convex shapes [17]. They did not receive much attention in computational geometry yet, however, except for probing of convex planar sets [19], and indirectly in polygon matching [1] in the definition of the Steiner point of a convex polygon.

## 2 Support functions

We start with recalling the standard definition of support functions for curves and surfaces. Consider a curve in  $\mathbb{R}^2$  (or a surface in  $\mathbb{R}^3$ ) that allows a  $C^2$  parameter description

$$\mathbf{g} : \Omega \rightarrow \mathbb{R}^d, \quad t \rightarrow \mathbf{g}(t)$$

where parametrization is taken over the domain  $\Omega$  (for  $d = 2, 3$ ). We denote with  $\mathbf{n}(t)$  the unit normal vector of  $\mathbf{g}$  at point  $\mathbf{g}(t)$ . For each  $t \in \Omega$ , the function

$$\delta(t) = \mathbf{n}(t) \cdot \mathbf{g}(t)$$

expresses the (signed) distance from the origin to the tangent line (plane) at  $\mathbf{g}(t)$ . Let us assume that  $\mathbf{g}$  is such that the mapping

$$\mathbf{n} : \Omega \rightarrow \mathbf{n}(\Omega) \subseteq \mathbb{S}^{d-1}$$

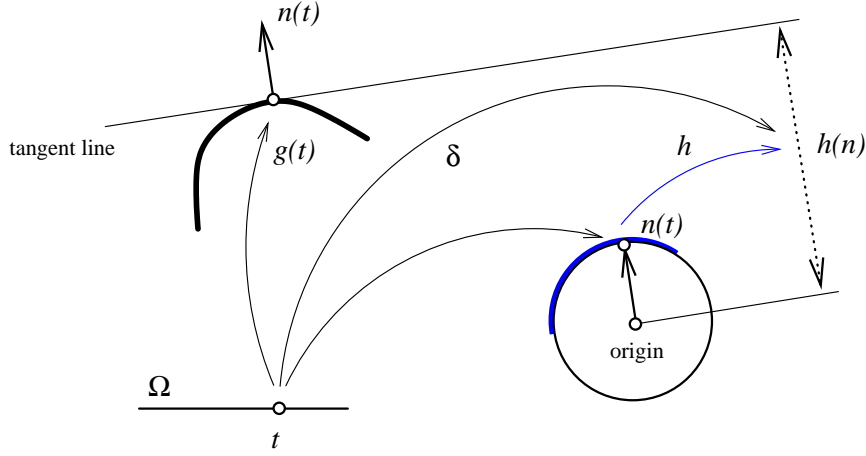


Figure 1: Support function for a convex curve

is a bijection. (For example,  $g$  must not contain inflection points if being a curve, or parabolic points if being a surface.) We define the *support function* of  $g$  as

$$h : n(\Omega) \rightarrow \mathbb{R}, \quad h = \delta \circ n^{-1} .$$

Figure 1 gives an illustration. For our purposes, it is of advantage to specify a support function as the restriction to  $\mathbb{S}^{d-1}$  of some suitable function defined on the entire space  $\mathbb{R}^d$ . For example, let the surface  $g$  in question be a sphere in  $\mathbb{R}^d$  with center  $c_i$  and radius  $r_i$ . Then, by simple calculations, the linear function

$$h_i : \mathbb{R}^d \rightarrow \mathbb{R}, \quad h_i(x) = c_i \cdot x + r_i \tag{1}$$

just yields the support function of  $g$  when being restricted to  $\mathbb{S}^{d-1}$ . In geometrical terms, the support function of a sphere in  $\mathbb{R}^d$  can be expressed by a hyperplane in  $\mathbb{R}^{d+1}$ . Note that this hyperplane passes through the origin if the sphere degenerates to a point in  $\mathbb{R}^d$ , i.e., has radius zero.

The following observation draws the connection to convex hulls. Its proof is immediate from the definition of a support function. Let  $\{s_1, \dots, s_n\}$  be a set of spheres in  $\mathbb{R}^d$ , each sphere  $s_i$  being associated with its linear function  $h_i$  as in (1).

**Property 1** *For a given vector  $x \in \mathbb{S}^{d-1}$ , let  $h_i(x) = \max_{1 \leq j \leq n} h_j(x)$ . Then there is a hyperplane  $H$  in  $\mathbb{R}^d$  normal to  $x$  and tangent to sphere  $s_i$  such that  $\{s_1, \dots, s_n\}$  lies on a fixed side of  $H$ .*

By Property 1, the convex hull of a set of  $n$  spheres in  $\mathbb{R}^d$  can be constructed by computing the upper envelope of  $n$  hyperplanes in  $\mathbb{R}^{d+1}$  and restricting it to  $\mathbb{S}^{d-1}$ . Interestingly, this already leads to worst-case optimal convex hull algorithms for circles in  $\mathbb{R}^2$ , and spheres in  $\mathbb{R}^3$ , respectively. Let us briefly explain this fact. The upper envelope of  $n$  hyperplanes in  $\mathbb{R}^{d+1}$  projects vertically to a power diagram [4] in  $\mathbb{R}^d$ , whose regions are then intersected with the unit sphere  $\mathbb{S}^{d-1}$ . The individual components of this intersection correspond, in a bijective way, to the individual components of the convex hull to be constructed. More precisely, convex hull patches defined by  $k$  spheres correspond to intersected faces of dimension  $d - k + 1$  of the power diagram, for  $1 \leq k \leq d$ . Note that the convex hull of  $n$  spheres with *equal* radius thus has the

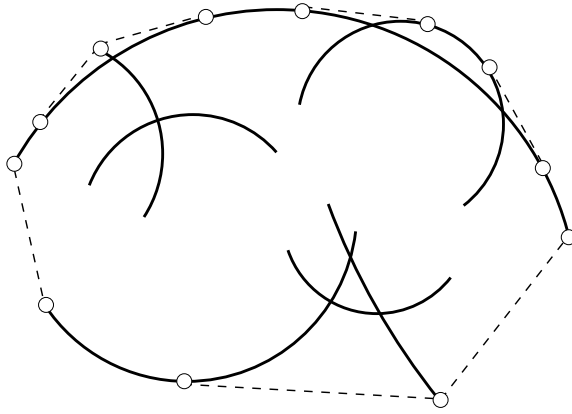


Figure 2: Convex hull of circular arcs

same combinatorial structure as the convex hull of the  $n$  sphere centers: From formula (1) it is obvious that the power diagram remains unchanged when all sphere radii are increased by the same constant.

The running time of this approach is dominated by the time needed to compute power diagrams, which is  $O(n \log n)$  in  $\mathbb{R}^2$  and  $O(n^2)$  in  $\mathbb{R}^3$ . This matches the performance of previously known worst-case optimal algorithms for convex hulls of circles [18], and of spheres [6], respectively. In fact, the method in [6] also transforms the problem to one dimension higher, and is similar to though less direct than ours. The convex hull of  $n$  spheres in  $\mathbb{R}^3$  has a combinatorial complexity of  $\Theta(n^2)$  in the worst case; see [6]. In contrast, by the considerations above, the size of the convex hull drops to  $O(n)$  if all spheres have the same radius.

Note finally that the very power diagram of the spheres (whose convex hull is to be constructed) cannot be used directly for this task. In contrast to the (euclidean) Voronoi diagram for spheres, unboundedness of the power region of a sphere does not indicate appearance of the sphere on the convex hull.

### 3 Circular arcs

In this section, we show that our approach based on support functions is particularly suited for computing the convex hull of  $n$  circular arcs in  $\mathbb{R}^2$ . Let  $\{a_1, \dots, a_n\}$  be a set of circular arcs, being positioned arbitrarily in the plane. We are going to describe an algorithm that computes the convex hull of  $\{a_1, \dots, a_n\}$  in worst-case optimal time, and that beats existing optimal methods in simplicity.

The convex hull's boundary consists of a cyclic sequence of *tangents* (line segments touching two input arcs) and *edges* (connected pieces of input arcs). The *vertices* of the convex hull are the endpoints of its tangents and edges; compare Figure 2. Note that a single arc may give rise to many edges. Our algorithm runs in  $O(h_m \log n)$  time and  $O(n)$  space, where  $h_m$  denotes the worst case number of vertices.

The first difference to the case of circles (see Section 2) arises because a circular arc is not a convex set. Its support function is defined only on an interval of  $\mathbb{S}^1$ . We might 'convexify' each arc by taking its convex hull beforehand, but this bears the problem that the support function of the resulting circular cap cannot be expressed by a linear object in  $\mathbb{R}^3$ . However, another simple trick can be applied. The endpoints of each arc are interpreted as circles of radii zero, and their resulting  $2n$  support functions are taken into account, too. This ensures that the upper envelope of all these support functions is defined on entire  $\mathbb{S}^1$  and, on the other hand, does not change the convex hull.

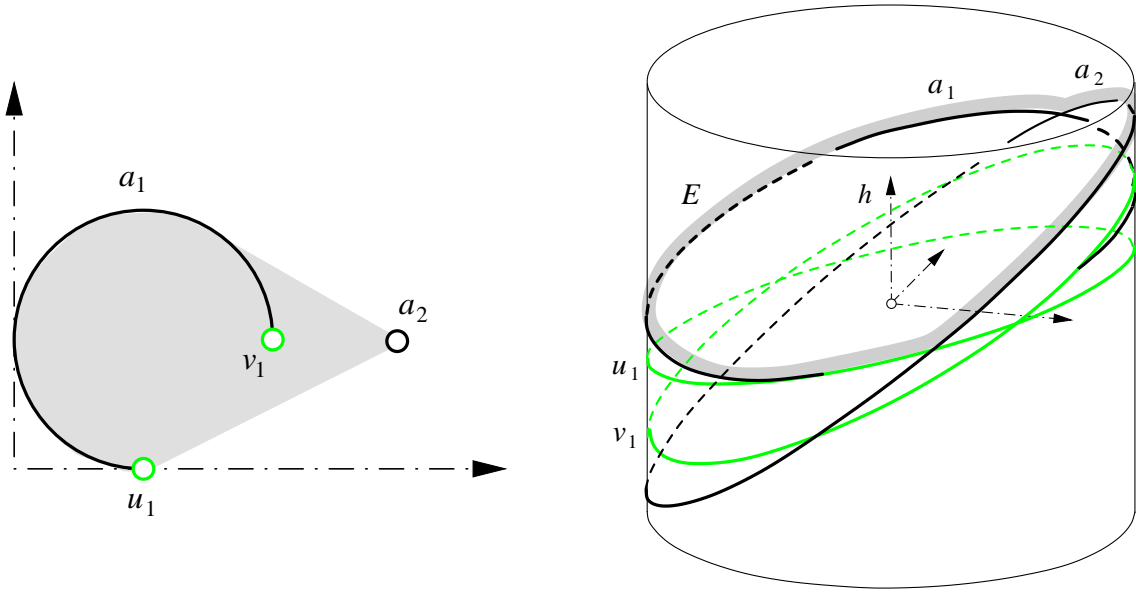


Figure 3: Circular objects and their support functions

In geometrical terms, each arc  $a_i$  now gives rise to three linear objects in  $\mathbb{R}^3$  as follows. Let  $u_i$  and  $v_i$  be the endpoints of arc  $a_i$ , and let the circle containing  $a_i$  be defined by its center  $c_i$  and radius  $r_i$ . Then we consider the part of the plane  $z = c_i \cdot x + r_i$  that projects vertically to the wedge with apex at the origin and aperture angle given by  $a_i$ . In addition, we consider the two planes  $z = u_i \cdot x$  and  $z = v_i \cdot x$ . By Property 1, the upper envelope of these  $3n$  linear objects, when restricted to  $\mathbb{S}^1$ , corresponds bijectively to the convex hull of the set  $\{a_1, \dots, a_n\}$  of circular arcs.

Let us have a closer look at this situation. Each of the linear object above intersects the vertical cylinder  $x^2 = 1$  either in a closed curve (if the object was a plane) or in an open curve (if the object was a planar wedge). See Figure 3, where closed curves (ellipses) are obtained for the radius-zero circles  $u_1$ ,  $v_1$ , and  $a_2$ , and an open elliptic curve is obtained for the circular arc  $a_1$ . We are interested in the upper envelope,  $E$ , of these curves (shown shaded in the figure). The curves pairwise cross at most twice, because each such crossing comes from intersecting a line in space with a cylinder. When applying the theory of Davenport-Schinzel sequences [21] to this setting, we get a superlinear upper bound  $O(\lambda_4(n))$  for the complexity of  $E$ , and thus on the quantity  $h_m$ . Recalling the bijection between curve crossings and convex hull tangents—either object witnesses equality of the support function—leads to a slightly weaker lower bound. To this end, we simulate with circular arcs the  $\Theta(\lambda_3(n))$ -size envelope construction in [21] for line segments, in a way such that each envelope crossing between two segments corresponds to a convex hull tangent for two arcs. Roughly speaking, each segment is bent to become an arc (so as to enable a tangent above each envelope crossing), and then is shortened such that the crossing gets sufficiently close to one of its endpoints (in order to prevent the endpoint from influencing the convex hull more than in an  $\varepsilon$ -range).

Note that the relationship to Voronoi diagrams mentioned in the introduction does not help us in analyzing the size of the convex hull, because we do not have easy control over the diagram size as soon as circular arcs start intersecting; the diagram's worst case complexity is  $\Theta(n^2)$ .

It remains to describe how to compute the upper envelope  $E$  in a simple and efficient way. Standard techniques, namely, divide-and-conquer paired with a radial sweep method for merging the upper envelopes of two subsets of curves, can be applied. This task basically mimics Mergesort for the angles of the envelope

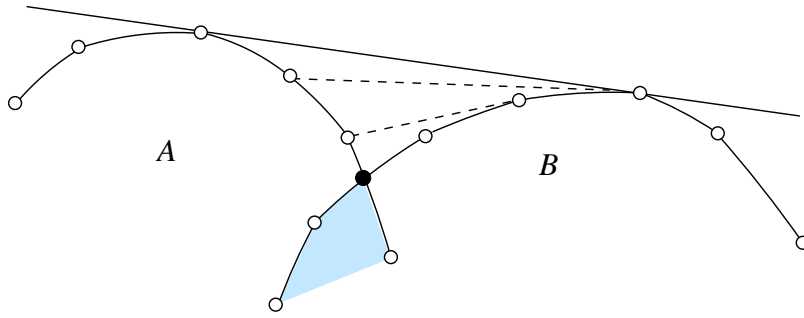


Figure 4: Constructing an intersection vertex

vertices. Clearly, in the merge phase we may have to consider new vertices (namely, crossings between curves), and disregard old ones on account of their  $z$ -values checked during the sweep. The good news is that such crossings can be obtained without intersecting the corresponding two spatial curves on  $x^2 = 1$ . In fact, these curves are never needed explicitly, neither during the algorithm nor as part of the output. For each crossing in question, it suffices to determine the projection  $\ell$  onto  $\mathbb{R}^2$  of the line of intersection of two planes in  $\mathbb{R}^3$ , intersect  $\ell$  with  $\mathbb{S}^1$ , and determine the  $z$ -value of the appropriate one among the two intersection points by plugging into the equation of a plane. By the maximal size of the envelopes constructed (see above), the overall running time of this algorithm is  $O(h_m \log n)$ . Note that only trivial data structures are used.

Let us work out the differences of this algorithm to a similar approach that come into mind: Graham's scan [9], which is based on radially sorting the objects (originally, points) whose convex hull is to be constructed. As, in general, no radial order exists for a set of circular arcs  $\{a_1, \dots, a_n\}$ , this method cannot be applied directly. An adequate generalization would have to find the envelope of the set  $\{a_1, \dots, a_n\}$  with respect to some point,  $c$ , interior to its convex hull, and then apply one of the  $O(n)$  time convex hull algorithms in [20, 5, 2] to the resulting closed curve. An envelope algorithm similar to the one described above will work, though not prior to splitting each arc  $a_i$  at internal points of tangency as seen from point  $c$ . The main advantage of our approach, however, is that after having the envelope of the support functions at hands, the combinatorial structure of the convex hull is completely determined, whereas additional effort (computing the convex hull of a noncrossing circular arc spline) is needed, otherwise.

We remark that neither for the *envelope* of circular arcs, nor for the *convex hull* of circular arcs, the upper complexity bound  $O(\lambda_4(n))$  is known to be tight.

Using our convex hull algorithm, the intersection of two *convex* shapes  $A$  and  $B$ , bounded by a total of  $n$  circular arcs, can be computed in a simple way. Each vertex of the intersection  $A \cap B$  can be uniquely assigned to a tangent of the convex hull of  $A \cup B$ , because this tangent signals equality of the two support functions. Starting at each hull tangent, we can identify the corresponding vertex (if it exists) by simple tests for tangency to one of  $A$  or  $B$ ; see Figure 4. The size of the convex hull is  $O(n)$ , because  $A \cap B$  has only  $O(n)$  vertices, and each vertex corresponds to a hull tangent if  $A \cap B \neq \emptyset$ . An  $O(n \log n)$  algorithm results that is less involved than the classical plane sweep approach.

## References

- [1] O. Aichholzer, H. Alt, G. Rote. *Matching shapes with a reference point*. Int'l J. Computational Geometry and Applications 7 (1997), 349-363.
- [2] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Jüttler, M. Oberneder, Z. Sir. *Computational and structural advantages of circular boundary representation*. Proc. 10th Workshop on Algorithms and Data Structures, WADS'07, Springer LNCS 4619, 2007, 374-385.
- [3] H. Alt, O. Cheong, A. Vigneron. *The Voronoi diagram of curved objects*. Discrete & Computational Geometry 34 (2005), 439-453.
- [4] F. Aurenhammer. *Power diagrams: properties, algorithms, and applications*. SIAM J. Computing 16 (1987), 78-96.
- [5] C.L. Bajaj, M.-S. Kim. *Convex hulls of objects bounded by algebraic curves*. Algorithmica 6 (1991), 533-553.
- [6] J.-D. Boissonnat, A. Cerezo, O. Devillers, J. Duquesne, M. Yvinec. *An algorithm for constructing the convex hull of a set of spheres in dimension  $d$* . Computational Geometry: Theory and Applications 6 (1996), 123-130.
- [7] D.P. Dobkin, D.L. Souvaine. *Computational geometry in a curved world*. Algorithmica 5 (1990), 421-457.
- [8] P.K. Ghosh, K.V. Kumar. *Support function representation of convex bodies, its application in geometric computing, and some related representations*. Computer Vision and Image Understanding 72 (1998), 397-403.
- [9] R. Graham. *An efficient algorithm for determining the convex hull of a finite point set*. Information Processing Letters 1 (1972), 132-133.
- [10] P.M. Gruber, J.M. Wills. *Handbook of Convex Geometry*. Elsevier, North-Holland, Amsterdam 1993.
- [11] R.A. Jarvis. *On the identification of the convex hull of a finite set of points in the plane*. Information Processing Letters 2 (1973), 18-21.
- [12] D.G. Kirkpatrick, R. Seidel. *The ultimate planar convex hull algorithm?* SIAM J. Computing 15 (1986), 287-299.
- [13] Z. Li, D.S. Meek. *Smoothing an arc spline*. Computers & Graphics 29 (2005), 576-587.
- [14] A. Melkman. *On-line construction of the convex hull of a simple polygon*. Information Processing Letters 25 (1987), 11-12.
- [15] F. Nielsen, M. Yvinec. *An output-sensitive convex hull algorithm for planar objects*. Int'l. J. Computational Geometry & Applications 8 (1998), 39-65.
- [16] L.A. Piegl, W. Tiller. *Biarc approximation of NURBS curves*. Computer-Aided Design 34 (2002), 807-814.
- [17] J.R. Prince, A.S. Willsky. *Reconstructing convex sets from support line measurements*. IEEE Trans. Pattern Analysis and Machine Intelligence 12 (1990), 377-389.
- [18] D. Rappaport. *A convex hull algorithm for discs, and applications*. Computational Geometry: Theory and Applications 1 (1992), 171-187.
- [19] T.J. Richardson. *Approximation of planar convex sets from hyperplanes probes*. Discrete & Computational Geometry 18 (1997), 151-177.

- [20] A.A. Schäffer, C.J. Van Wyk. *Convex hulls of piecewise-smooth Jordan curves*. J. Algorithms 8 (1987), 66-94.
- [21] M. Sharir, P.K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.
- [22] Y. Yue, J.L. Murray, J.R. Corney, D.E.R. Clark. *Convex hull of a planar set of straight and circular line segments*. Engineering Computations 16 (1999), 858-875.
- [23] C.K. Yap. *An  $O(n \log n)$  algorithm for the Voronoi diagram of a set of simple curve segments*. Discrete & Computational Geometry 2 (1987), 365-393.