Available online at www.sciencedirect.com

ScienceDirect

Computer-Aided Design 39 (2007) 518–527

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Exact and approximate construction of offset polygons[☆]

Ron Wein

School of Computer Science, Tel-Aviv University, Israel

Received 21 August 2006; accepted 22 January 2007

Abstract

The Minkowski sum of two sets $A, B \in \mathbb{R}^2$, denoted $A \oplus B$, is defined as $\{a + b \mid a \in A, b \in B\}$. We describe an efficient and robust implementation of the construction of the Minkowski sum of a polygon in $\mathbb{R}^2$ with a disc, an operation known as *offsetting* the polygon. Our software package includes a procedure for computing the exact offset of a straight-edge polygon, based on the arrangement of conic arcs computed using exact algebraic number-types. We also present a conservative approximation algorithm for offset computation that uses only rational arithmetic and decreases the running times by an order of magnitude in some cases, while having a guarantee on the quality of the result. The package will be included in the next public release of the Computational Geometry Algorithms Library, CGAL Version 3.3. It also integrates well with other CGAL packages; in particular, it is possible to perform regularized Boolean set-operations on the polygons the offset procedures generate.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Minkowski sums; Offset polygons; Exact geometric computation; Robustness

1. Introduction

Given two sets $A, B \in \mathbb{R}^2$, their *Minkowski sum*, denoted by $A \oplus B$, is the set $\{a + b \mid a \in A, b \in B\}$. Planar Minkowski sums are used in many applications, such as motion planning and computer-aided design and manufacturing. In this paper we focus on one important variant of the planar Minkowski-sum computation problem: computing the Minkowski sum of a simple polygon with a disc. This operation is also known as *offsetting* a polygon.

While the process of computing an offset polygon is quite easy and straightforward to describe, giving a robust software implementation for this process is a non-trivial task. The offsetting process can be divided into three steps:

1. Computing the convolution of the polygon and the disc, which is a closed curve comprising straight-line segments and circular arcs.
2. Constructing the arrangement of the convolution components, namely the planar subdivision they induce into 2-dimensional, 1-dimensional and 0-dimensional cells (*faces*, *edges* and *vertices*, respectively).
3. Deducing the shape of the offset polygon from the arrangement computed in the previous step. This is done by determining which arrangement faces are contained in the offset polygon.

The main difficulty lies in the second step. In the presence of degeneracies (e.g., three or more arcs that intersect at a common point, an intersection point that coincides with an arc endpoint, etc.) – or even in nearly degenerate situations – the arrangement construction is prone to numeric instabilities and cannot be accomplished using machine-precision arithmetic. On the other hand, computing the arrangement using exact arithmetic is a computationally demanding operation, as it requires handling algebraic numbers of degree up to 4 in a precise manner.

In this paper we present a software package that includes the implementations of two algorithms for computing offset polygons. The first uses an exact construction, and relies on the exact number-types provided by external libraries, such as CORE or LEDA (see below). The second algorithm provides a conservative approximation scheme for the offset polygon. Its main advantage is that it uses only exact rational arithmetic, thus it is typically much faster than the exact construction. Moreover, as it is often required to apply set-operations (e.g.

[☆] Partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No. IST-006413 (ACS — Algorithms for Complex Shapes), and by the Hermann Minkowski-Minerva Center for Geometry at Tel Aviv University.

E-mail address: wein@post.tau.ac.il.

union or intersection) on the resulting offset polygons, using the approximated offsets can lead to yet a larger benefit. The algorithms presented in this paper are part of the Minkowski-sum package, which will be included in the next public release of CGAL (Version 3.3), and integrate well with other CGAL packages.

1.1. Related work: exact computation of Minkowski sums

The simplest variant of the Minkowski-sum computation problem involves two planar polygons. If P and Q are simple planar polygons having m and n vertices respectively, then $P \oplus Q$ is a subset of the *arrangement* (see, e.g., [8]) of $O(mn)$ line segments, where each segment is the Minkowski sum of an edge of P with a vertex of Q, or vice versa. The size of the sum is therefore bounded by $O(m^2n^2)$, and this bound is tight [14]. However, if both P and Q are convex, then $P \oplus Q$ is a convex polygon with $m + n$ vertices at most, and can be computed in $O(m + n)$ time (see, e.g., [3, Chapter 13]). If only P is convex, the Minkowski sum of P and Q is bounded by $O(mn)$ [15], and this bound is tight as well.

In a survey paper on Minkowski-sum computation, which dates back less than a decade ago, Lee et al. [16] mention that "*in spite of the paramount importance of the Minkowski sum operation in practice, ... there has been no known implemented algorithm that can determine the curve arrangement in a robust way*". Indeed, robustness in geometric computation is a fundamental, difficult and well-known problem arising from the special nature of geometric algorithms; see, e.g., [10,11,25]. In the classic computational-geometry literature two assumptions are usually made to simplify the design and analysis of geometric algorithms. First, inputs are in "general position" — that is, degenerate input (e.g., three curves intersecting at a common point) is precluded. Secondly, operations on real numbers yield accurate results (the "real RAM" model [20], which also assumes that each basic operation takes constant time). Unfortunately, these assumptions do not hold in practice, making the implementation of robust geometric algorithms such a difficult task.

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 of CGAL [23] adopts, as does CGAL in general, the *exact computation* paradigm [26]. Namely, it requires that all geometric operations are carried out in a precise manner, so it can correctly detect degenerate situations and handle them properly. Furthermore, it makes no general-position assumptions on its input. The arrangement package of CGAL thus provides the required foundations for the implementation of the basic Minkowski-sum operations. We also mention the 2D Boolean set-operation package [5], another important CGAL package whose implementation is based on the arrangement infrastructure, which supports the application of regularized Boolean set-operations on general polygons, namely polygons whose edges are given as general planar curves.

As mentioned above, computing the Minkowski sum of two convex polygons can be performed in linear time in the total number of their edges using a simple procedure that is easily implemented in software. The prevailing method for computing the sum of two non-convex polygons P and Q, is therefore based on *convex decomposition*: we decompose P into convex sub-polygons $P_1, \ldots, P_k$, and Q into convex sub-polygons $Q_1, \ldots, Q_\ell$, obtain the Minkowski sum of each pair of sub-polygons, and compute the union of the $k\ell$ pairwise sub-sums. Namely, we calculate $P \oplus Q = \bigcup_{i,j}(P_i \oplus Q_j)$. We note that if both polygons are given by sequences of vertices with rational coordinates,[2] then all calculations and geometric operations can be carried out using rational arithmetic.

Flato [4] (see also [1]) developed an exact and robust implementation of the decomposition method for computing the Minkowski sum of two simple polygons. He implemented about a dozen different polygon-decomposition strategies and several methods for the union computation, and conducted thorough experiments to determine the optimal decomposition and union strategies. Flato's code is based on CGAL Version 2.0, and employs exact rational arithmetic to guarantee robustness and produce exact results even on degenerate inputs. This was the first implementation capable of handling degenerate inputs, and the only one that correctly identifies low-dimensional elements of the Minkowski sum, such as antennas or isolated vertices.

The LEDA library [18] also contains functions for robust Minkowski-sum computation of two polygons based on convex polygon decomposition that use exact rational arithmetic.[3] However, these functions are limited to performing *regularized* Minkowski-sum computations, which eliminate low-dimensional features of the output.

Another approach to computing the Minkowski sum of two polygons is to calculate the *convolution* of the boundaries of P and Q [6,7]. Our software package contains functions for computing Minkowski sums of two polygons, employing either the decomposition method or the convolution method, using exact rational arithmetic. To the best of our knowledge, this is the first implementation of software for robust and exact computation of Minkowski sums that is based on the convolution method. As our experiments show [22], using the convolution method we construct intermediate geometric entities that are more compact than the ones constructed using the decomposition method. Subsequently, we obtain better runtime performance.

While the Minkowski sum of two polygons can be combinatorially complex, the complexity of the Minkowski sum of a polygon with n vertices with a disc is always

[1] See the CGAL project homepage, at http://www.cgal.org/.

[2] Recall that the machine-precision `float` or `double` types have bounded mantissas, so if the input coordinates are given as floating-point numbers, we can easily convert them into rational numbers.

[3] For more details and detailed online documentation, see: http://www.algorithmic-solutions.info/leda_guide/geo_algs/minkowski.html.

$O(n)$. However, the difficulty in offsetting a polygon in a robust manner is *numerical* and not *combinatorial*. Let us assume that all polygons we handle are *rational* — namely, all the polygon vertices have rational coordinates. In this paper we assume that a rational number is represented using two unbounded integers (its numerator and denominator), so the basic arithmetic operations on rational numbers ($+$, $-$, $\times$ and $\div$) can be carried out without any loss of precision. As mentioned above, the Minkowski sums of two such polygons is also a rational polygon, and we need only exact rational arithmetic to compute it in a precise manner. In contrast, rational arithmetic is insufficient for offsetting a rational polygon by a rational radius. The resulting offset polygon comprises line segments and circular arcs, and the coordinates of its vertices are typically algebraic numbers of degree up to four.[4] While implementing an exact rational number-type is not difficult for an experienced programmer, and there are also several implementations that are publicly available (e.g., GMP, the Gnu Multi-Precision library[5]), handling algebraic numbers in a precise manner is an extremely difficult, and a highly time-consuming task. The CORE library[6] [12] and the numerical facilities of LEDA[7] [18, Chapter 4]) provide certified computations with algebraic numbers, but using the algebraic number-types they provide may be too time consuming for industrial applications.

Alongside an implementation of a robust algorithm that computes the offset of a rational polygon in an exact manner, we present a simple yet powerful approximation algorithm for offsetting a simple polygon, which overcomes the algebraic difficulties by using only exact rational arithmetic. Our algorithm is conservative, namely if $P_r = P \oplus B_r$ is the exact Minkowski sum of a polygon P with a disc B_r of radius r, we compute a generalized polygon $\tilde{P}_r$, bounded by line segments and circular arcs with rational coefficients, such that $P_r \subseteq \tilde{P}_r$. We can also control the approximation error and make it arbitrarily small.

1.2. Paper outline

The rest of this paper is organized as follows. In Section 2 we explain the algebraic difficulties that the exact construction of the offset polygon poses, and present our approximation scheme that offsets a polygon using only rational arithmetic in Section 3. We also prove an upper bound on the approximation error. The experimental results we bring in Section 4 show the considerable speedup our approximation algorithm achieves over the exact construction. We finally give some concluding remarks in Section 5.

2. The exact offsetting procedure

Offsetting is a fundamental task in CAD/CAM. The main body of the CAD literature on this subject concentrates on computing offsets of curves and surfaces (see, e.g., [13,16,17] and the references therein). In the general case, the offset curves of rational planar curves are not rational, so it is possible to compute them only using approximate techniques. As we focus in this paper on the special case of offsetting a polygon, we are able to provide exact construction of the offset, or alternatively an approximate construction with guaranteed quality.

At first glance offsetting may seem an easier task compared to computing the Minkowski sum of two polygons. However, when we aim for a robust implementation, offsetting efficiently is much more computationally demanding. Let us assume that we are given a polygon P with n vertices $(p_0, \dots, p_{n-1})$ that are ordered counterclockwise around P's interior. All vertices have rational coordinates. We wish to compute the offset polygon P_r, namely the Minkowski sum of P with a disc of radius r, where r is rational. We show that even this relatively simple task involves exact computation with algebraic numbers, if we wish our computations to be exact.

We use the arrangement package of CGAL [23] for computing the offset of a polygon. The arrangement package provides one central class-template named `Arrangement_2<Traits>` that is parameterized by a *geometric traits-class*. This traits-class defines the type of curves it can handle and provides a set of geometric predicates and constructions involving these curves. Using these operations, the `Arrangement_2` class can construct and maintain an arrangement of curves of the type defined by the traits class. For example, when computing the sum of two polygons we need to construct arrangements of line segments, which is quite straightforward to implement using exact rational arithmetic, based on the basic geometric operations provided by the geometric kernels of CGAL [9]. When we offset a polygon we need to handle more complex geometric curves. In this section we explain how to construct the exact offset of a polygon using arcs of conic curves, and in the next section we introduce our approximation scheme that enables working with simpler curves.

If P is a convex polygon, the offset is easily computed by shifting each polygon edge by r away from the polygon, namely to the right side of the edge (we regard each edge of the polygon as directed from p_i to p_{i+1}). As a result we obtain a collection of n disconnected *offset edges*. Each pair of adjacent offset edges, induced by $p_{i-1}p_i$ and $p_i p_{i+1}$,[8] are connected by a circular arc of radius r, whose supporting circle is centered at p_i. The angle that defines such a circular arc equals $\pi - \measuredangle(p_{i-1}, p_i, p_{i+1})$; see Fig. 1(a) for an illustration. Naturally, the running time of this simple process is linear in the size of the polygon.

If P is not convex, its offset can be obtained by decomposing it into convex sub-polygons $P_1, \dots, P_m$ such that $\bigcup_{i=1}^{m} P_i = P$, computing the offset of each sub-polygon and finally

[4] A real number $\alpha \in \mathbb{R}$ is called algebraic of degree d if there exists a polynomial of degree d with integer coefficients such that α is a root of this polynomial, and α is not a root of any integer polynomial with smaller degree.

[5] http://www.swox.com/gmp/.

[6] http://www.cs.nyu.edu/exact/core/.

[7] http://www.algorithmic-solutions.com/enleda.htm.

[8] In this paper, when we increment or decrement an index of a polygon vertex, we always do it modulo the size of the polygon n.

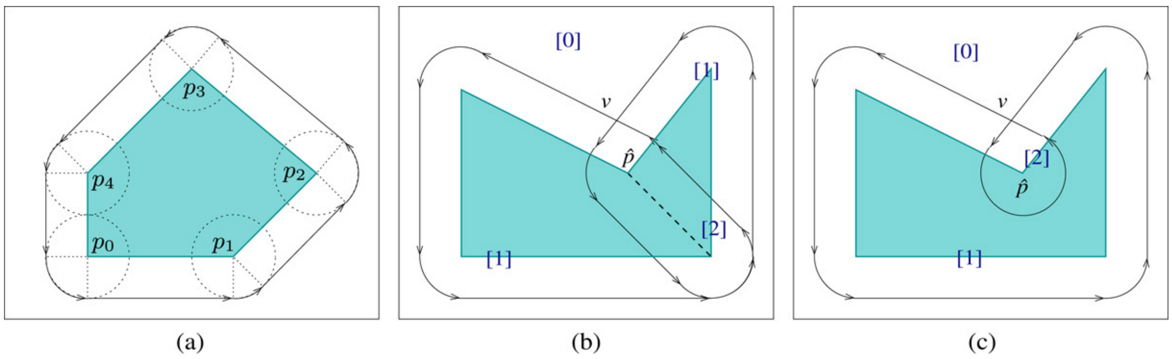


Fig. 1. (a) Offsetting a convex polygon. (b) Computing the offset of a non-convex polygon by decomposing it into convex sub-polygons by adding the dashed diagonal; $\hat{p}$ is a reflex vertex. The boundary curves of the two sub-offsets induce an arrangement with four faces, whose *cover numbers* are shown in brackets. (c) Offsetting a non-convex polygon by computing its convolution with a disc. The convolution cycle induces an arrangement with three faces, whose *winding numbers* are shown in brackets.

calculating the union of these sub-offsets (see Fig. 1(b)). Flato [4, Appendix A] describes a simple algorithm for computing the union of a set of polygons from the arrangement of the curves that constitute their boundaries. This is done by computing the *cover number* $N(f)$ for each arrangement face f, namely the number of polygons that cover this face. This procedure is linear in the size of the arrangement and involves only combinatorial operations (and no geometric operations, which are usually more computationally expensive). Naturally, the Minkowski sum is the union of all faces with a positive cover number.

However, as the experiments in [22] show, it is more efficient to compute Minkowski sums using the convolution method. In our case, the convolution of the polygon with a disc is a single cycle comprising line segments and circular arcs. The sub-segments of the convolution cycle can be constructed by applying the process described in the previous paragraphs. The only difference is that a circular arc induced by a reflex vertex p_i is defined by an angle $3\pi - \measuredangle(p_{i-1}, p_i, p_{i+1})$; see Fig. 1(c) for an illustration. Once we obtain the convolution cycle, we construct the arrangement of the line segments and circular arcs that constitute this cycle. It is now possible to associate a non-negative number with each face f in the arrangement, which counts the number of times the convolution curve winds in a counterclockwise direction around f, minus the number of times it winds in a clockwise direction around this face. This number is called the *winding number* of f and we denote it $W(f)$ (see, e.g., [19, Chapter 7] for the topological definition of the winding number and some examples). As explained in [22], computing the winding number in the arrangement of the convolution segments is done in the same way the cover numbers are computed in an arrangement of boundary curves (see above). We finally report on the union of all faces with $W(f) > 0$.

Note that the number of arcs in the convolution cycle of a non-convex polygon P and a disc is always smaller than the number of arcs that constitute the boundary of the sub-offsets (namely, the offset polygon of the convex sub-polygons) when using the convex decomposition method. As a result, the arrangement we compute as an intermediate structure is less complex and can be constructed faster. From now on we therefore concentrate on the implementation of the convolution method alone.

2.1. Exact representation of the offset edges

We now focus on the algebraic representation of the line segments and circular arcs that form the convolution cycle. We first note the all circular arcs are clearly supported by rational circles, as their center points (the polygon vertices) always have rational coordinates and their radii equal $r \in \mathbb{Q}$. Nevertheless, as we show next, the coordinates of the vertices of the offset of a rational polygon by a rational radius r are in general *irrational*.

Let us examine how the offset edges look like. We consider the polygon edge $\vec{p_1 p_2}$ to be directed from $p_1 = (x_1, y_1)$ to $p_2 = (x_2, y_2)$, and denote by θ the angle it forms with the x axis. Let $\ell = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ be the edge length, so we have $\cos\theta = \frac{1}{\ell}(x_2 - x_1)$ and $\sin\theta = \frac{1}{\ell}(y_2 - y_1)$. As we traverse the polygon edges in a counterclockwise orientation, we construct the offset edge $v_1 v_2$ that corresponds to $p_1 p_2$ by shifting either polygon vertex by a vector whose length is r and which forms an angle of $\phi = \theta - \frac{\pi}{2}$ with the x-axis. It is easy to see that:

$$\sin\phi = \sin\theta \cdot \cos\frac{\pi}{2} - \cos\theta \cdot \sin\frac{\pi}{2} = -\cos\theta = \frac{1}{\ell}(x_1 - x_2), \quad (1)$$

$$\cos\phi = \cos\theta \cdot \cos\frac{\pi}{2} + \sin\theta \cdot \sin\frac{\pi}{2} = \sin\theta = \frac{1}{\ell}(y_2 - y_1). \quad (2)$$

Thus, $v_j = (x_j + \frac{r}{\ell}(y_2 - y_1), y_j + \frac{r}{\ell}(x_1 - x_2))$ for $j = 1, 2$. Indeed, the coordinates of these points are solutions of quadratic equations with rational coefficients (algebraic numbers of degree two), but the segment $v_1 v_2$ is supported by a line with *irrational* coefficients: it is easy to show that if the supporting line of $p_1 p_2$ is $ax + by + c = 0$ (where $a, b, c \in \mathbb{Q}$), then the line supporting $p_1 p_2$ is $ax + by + (c + \ell r) = 0$, where ℓ is usually an irrational number. The intersection points between two segments representing offset edges (see for example the point v in Fig. 1(b) and (c)), or between an offset edge and a circular arc that represents an offset vertex, are algebraic numbers of degree four, namely roots of polynomials with integer coefficients of degree 4.

A simpler representation of the offset edges is based on the fact that the locus of all points lying at distance r from the line $ax + by + c = 0$ is given by:

$$\frac{(ax + by + c)^2}{a^2 + b^2} = r^2. \tag{3}$$

Eq. (3) above specifies a degenerate conic curve (a pair of parallel lines) with rational coefficients, hence offset edges can be represented as bounded arcs of rational conic curves. Note that the circular arcs of the offset are also special cases of conic arcs. Indeed, the arrangement package of CGAL contains a traits-class that handles conic arcs with rational coefficients [21]. This class uses the exact number-types implemented in the CORE library (see Section 1) to carry out the necessary geometric operations in a precise manner. Offset polygons can therefore be constructed in an exact manner using the conic-traits class.

3. Approximating the offset polygon

Although our software package is capable of constructing the exact representation of offset polygons, as explained in the previous section, the computational overhead incurred by the exact computation with algebraic numbers makes this process relatively slow (see more in Section 4), while for many practical applications it is sufficient to obtain a conservative approximation of the offset polygon with some bounded error, rather than computing the exact representation.

In this section we introduce *one-root numbers* and explain how they can be used to robustly construct arrangements of segments of rational lines and arcs of rational circles using only exact rational arithmetic. Then we introduce our approximation algorithm that uses these algebraic foundations and constructs a tight superset of the offset polygon using only rational arithmetic.

3.1. One-root numbers and the circle/segment traits-class

Given $\alpha, \beta, \gamma \in \mathbb{Q}$ with $\gamma > 0$, the real number $\alpha + \beta\sqrt{\gamma}$ is called a *one-root number*. The term "one-root number" was given by Berberich et al. [2]. In their work, they used the fact that one-root numbers can be handled by LEDA rather efficiently; this gives them the ability to compare two such numbers in an exact manner. In our context, one-root numbers play an important role, since the solution of any quadratic equation with rational coefficients (namely $ax^2 + bx + c = 0$, where $a, b, c \in \mathbb{Q}$) is a one-root number, as it equals $\frac{-b}{2a} \pm \frac{1}{2a}\sqrt{b^2 - 4ac}$.

Observe that if $x = \alpha + \beta\sqrt{\gamma}$ is a one-root number and $q \in \mathbb{Q}$, then $x \pm q$, $x \cdot q$ and $\frac{x}{q}$ are obviously one-root numbers. In addition, $\frac{q}{x} = \frac{q}{\alpha+\beta\sqrt{\gamma}} = \frac{q\cdot(\alpha-\beta\sqrt{\gamma})}{\alpha^2-\beta^2\gamma}$ and $x^2 = (\alpha + \beta\sqrt{\gamma})^2 = (\alpha^2 + \beta^2\gamma) + 2\alpha\beta\sqrt{\gamma}$ are also one-root numbers. The important property of one-root numbers is that the operations of evaluating the sign of a one-root number and comparing two one-root numbers can be carried out *precisely* using only exact rational arithmetic; see Appendix A.1 for the proofs.

The arrangement package of CGAL contains the *circle/segment traits-class* that handles curves that are either:

- Arcs of *rational circles*, namely circles of the form $(x - x_0)^2 + (y - y_0)^2 = R$, where the circle center (x_0, y_0) has rational coordinates and the *squared* radius R is also rational. Note that the radius itself may not be rational. A general circular arc is given by its supporting circle, two endpoints s and t that satisfy the equation of the circle (these endpoints may be rational, or have one-root coordinates), and the orientation of the arc between the endpoints (clockwise or counterclockwise).
- Segments of *rational lines*, namely lines whose equation is $ax + by + c = 0$, where a, b and c are rational. The segment is given by its supporting line and its two endpoints s and t, whose coordinates can either be rational or one-root numbers.

Note that the coordinates of the intersection points of two rational circles, or of a rational circle and a rational line, are one-root numbers, as they are the roots of quadratic equations with rational coefficients. Therefore, when we split a circular arc at its intersection point with another arc or with a line segment, the two resulting arcs are representable by the curve type defined by the traits class.

Using the properties of one-root numbers it is possible to robustly implement all the geometric predicates and constructions needed for the arrangement construction and maintenance using only exact rational arithmetic; see Appendix A.2 for the fine technical details. This fact makes the circle/segment traits-class about an order of magnitude faster than the conic-traits class (recall that circular arcs and line segments are special cases of rational conic arcs); see [24] for experimental results. Unfortunately, as we showed in the previous section, offset edges cannot be realized as segments of lines with rational coefficients, hence they are not representable by the circle/segment traits-class. We next describe how we overcome this difficulty.

3.2. The approximation scheme

We next describe our approximation algorithm that avoids using expensive computations with algebraic numbers. First, we note that in case of a horizontal edge (where $y_1 = y_2$) or a vertical edge (where $x_1 = x_2$) its length ℓ is a rational number. In these cases we can construct the offset edge v_1v_2 in an exact manner, so in the following we assume that $x_1 \neq x_2$ and $y_1 \neq y_2$.

We approximate the offset edge by two line segments with rational coefficients, as shown in Fig. 2: in a manner we describe next, we find two points v'_1 and v'_2 with *rational* coefficients, such that v'_j lies on the circle $(x - x_j)^2 + (y - y_j)^2 = r^2$ (for $j = 1, 2$). Moreover, v'_1 and v'_2 are selected such that the angle ϕ'_1 that $\overrightarrow{p_1v'_1}$ forms with the x-axis is slightly smaller than ϕ, and we let $\Delta\phi_1 = \phi - \phi'_1$, and the angle ϕ'_2 that $\overrightarrow{p_2v'_2}$ forms with the x-axis is slightly larger than ϕ, and we denote $\Delta\phi_2 = \phi'_2 - \phi$ (recall that ϕ is the angle that the normal to the polygon edge $\overrightarrow{p_1p_2}$ forms with the x-axis).

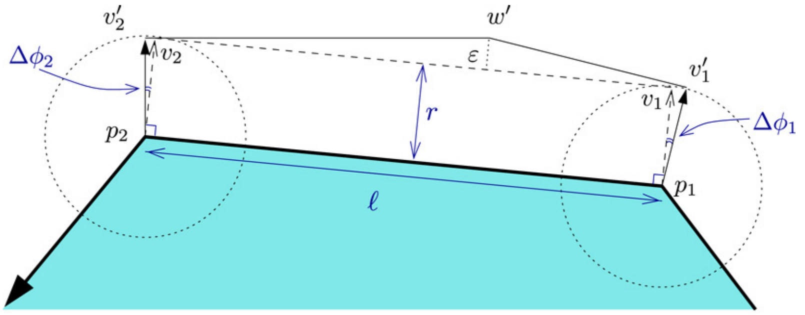


Fig. 2. Approximating the offset edge induced by the polygon edge $p_1 p_2$.

Observe that the lines tangent to the two circles at v_1' and v_2' have rational coefficients, so their intersection point w' also has rational coordinates. We use the two line segments $v_1' w'$ and $w' v_2'$ to approximate the offset edge $v_1 v_2$.

We now explain how to compute the rational points v_1' and v_2' with the properties mentioned above. Note that if $\tau = \tan\frac{\phi}{2}$ were a rational number, then $\sin\phi = \frac{2\tau}{1+\tau^2}$ and $\cos\phi = \frac{1-\tau^2}{1+\tau^2}$ would be rational as well, and v_1 and v_2 would both have rational coordinates. We therefore aim for a rational approximation of τ. Using the half-angle formulae[9] we can write:

$$\tau = \tan\frac{\phi}{2} = \frac{1-\cos\phi}{\sin\phi} = \frac{1-\frac{1}{\ell}(y_2-y_1)}{\frac{1}{\ell}(x_1-x_2)} = \frac{\ell+(y_1-y_2)}{x_1-x_2}, \tag{4}$$

$$\tau = \tan\frac{\phi}{2} = \frac{\sin\phi}{1+\cos\phi} = \frac{\frac{1}{\ell}(x_1-x_2)}{1+\frac{1}{\ell}(y_2-y_1)} = \frac{x_1-x_2}{\ell+(y_2-y_1)}. \tag{5}$$

As $\ell > |y_2 - y_1|$, the sign of τ is determined by $\mathrm{sign}(x_1 - x_2)$. If $x_1 > x_2$ (as is the case in the example depicted in Fig. 2), we have $\frac{\pi}{2} < \theta < \frac{3\pi}{2}$, hence $0 < \phi < \pi$ and $\tau > 0$. Let $\underline{\ell} \in \mathbb{Q}$ be a rational approximation of ℓ from below (that is, $0 < \ell - \underline{\ell} < \eta$ for some small $\eta > 0$). We now define the angles ϕ_1' and ϕ_2', based on Eqs. (4) and (5), respectively:

$$\tau_1' = \tan\frac{\phi_1'}{2} = \frac{\underline{\ell}+(y_1-y_2)}{x_1-x_2} < \frac{\ell+(y_1-y_2)}{x_1-x_2} = \tau, \tag{6}$$

$$\tau_2' = \tan\frac{\phi_2'}{2} = \frac{x_1-x_2}{\underline{\ell}+(y_2-y_1)} > \frac{x_1-x_2}{\ell+(y_2-y_1)} = \tau. \tag{7}$$

It is clear that $\phi_1' < \phi < \phi_2'$. The two tangency points of the approximating segments are therefore given by:

$$v_j' = \left(x_j + \frac{1-\tau_j'^2}{1+\tau_j'^2}\cdot r,\ y_j + \frac{2\tau_j'}{1+\tau_j'^2}\cdot r \right) \quad (j = 1, 2).$$

In case $x_1 < x_2$ we have $\tau < 0$. In this case we compute a rational approximation of ℓ from *above*, denoted $\bar{\ell}$ (thus $0 < \bar{\ell} - \ell < \eta$ for some small $\eta > 0$), and define $\tau_1' < \tau$ and $\tau_2' > \tau$ in an analogous manner to the definitions in Eqs. (6) and (7).

[9] See, e.g., http://mathworld.wolfram.com/Half-AngleFormulas.html.

Obtaining a rational approximation for ℓ is easy. Recall that $\ell^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$ is a rational number; for any rational $l_0 > 0$, the recursively defined series $l_{i+1} = \frac{1}{2}(l_i + \frac{\ell^2}{l_i})$ converges to ℓ. If we need to approximate ℓ from below, we simply look for the minimal index k such that $0 < \ell^2 - l_k^2 < \delta$, or such that $0 < \ell^2 - (\frac{\ell^2}{l_k})^2 < \delta$; we take $\underline{\ell} \longleftarrow l_k$ in the former case and $\underline{\ell} \longleftarrow \frac{\ell^2}{l_k}$ in the latter case. Computing an approximation from above is symmetric. Observe that if we fix a rational $\delta > 0$ value, all calculations are carried out using rational arithmetic.

3.3. The approximation bound

We next show how tight should the approximation of the edge length ℓ be, in order to guarantee that the polyline $v_1' w' v_2'$ we use for approximating the offset edge does not lie too far from the exact offset $v_1 v_2$.

Theorem 1. *For any polygon edge connecting $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, where $x_1 \neq x_2$ and $y_1 \neq y_2$, and any given $\varepsilon > 0$, let $\hat{\ell}$ be a rational approximation of the edge length $\ell = \|v_2 - v_1\|$ such that $|\ell^2 - \hat{\ell}^2| < \ell|\frac{\ell+(y_1-y_2)}{2(x_1-x_2)}|\cdot\varepsilon$. If we compute a polyline approximation $v_1' w' v_2'$ of the offset edge as described above, then the distance of the point w' from the supporting line of the true offset edge $v_1 v_2$ is upper bounded by ε.*

Proof. Let us assume that $x_1 > x_2$ and that $\hat{\ell}$ is an approximation of the edge length from below, so that we have $\ell^2 - \hat{\ell}^2 < \delta$ for some $\delta > 0$ (the proof for $x_1 < x_2$ is symmetric). Note that:

$$\ell^2 - \hat{\ell}^2 = (\ell + \hat{\ell})(\ell - \hat{\ell}) < 2\hat{\ell}(\ell - \hat{\ell}),$$

so $\ell - \hat{\ell} < \frac{\delta}{2\hat{\ell}}$. We now use the fact that $\tan(\alpha - \beta) = \frac{\tan\alpha - \tan\beta}{1+\tan\alpha\tan\beta}$ and using Eqs. (4) and (6) we obtain:

$$\tan\left(\frac{\phi - \phi_1'}{2}\right) = \frac{\tau - \tau_1'}{1 + \tau\tau_1'} = \frac{\frac{\ell+(y_1-y_2)}{x_1-x_2} - \frac{\hat{\ell}+(y_1-y_2)}{x_1-x_2}}{1 + \frac{\ell+(y_1-y_2)}{x_1-x_2}\cdot\frac{\hat{\ell}+(y_1-y_2)}{x_1-x_2}}$$
$$= \frac{(x_1-x_2)(\ell-\hat{\ell})}{(x_1-x_2)^2 + \ell\hat{\ell} + (\ell+\hat{\ell})(y_1-y_2) + (y_1-y_2)^2}$$
$$< \frac{(x_1-x_2)\frac{\delta}{2\hat{\ell}}}{\ell^2 + \ell\hat{\ell} + (\ell+\hat{\ell})(y_1-y_2)} < \frac{x_1-x_2}{4\hat{\ell}^2(\hat{\ell}+(y_1-y_2))}\cdot\delta.$$

Table 1
The running times (measured in milliseconds) of exact and approximate offset computations

Input polygon	Size	Bounding box	Offset radius (r)	Running times		
				Exact offset	Approx. offset	
					($\varepsilon = 10^{-7}r$)	($\varepsilon = 10^{-10}r$)
`Wheel`	40 (14)	$10^8 \times 10^8$	$5 \cdot 10^6$	88	35	54
`Spiked`	64 (40)	600×510	5	1378	60	71
`Random`	40 (19)	775×788	15	95	56	68
`Comb`	53 (24)	1250×200	25	138	45	50
`Chain`	82 (37)	$1.2 \cdot 10^8 \times 4 \cdot 10^7$	$2 \cdot 10^6$	1210	109	134
`Country`	50 (24)	$1.7 \cdot 10^6 \times 4 \cdot 10^6$	10^5	451	66	82

The *Size* column lists the number of polygon vertices and the number of reflex vertices (in parentheses).

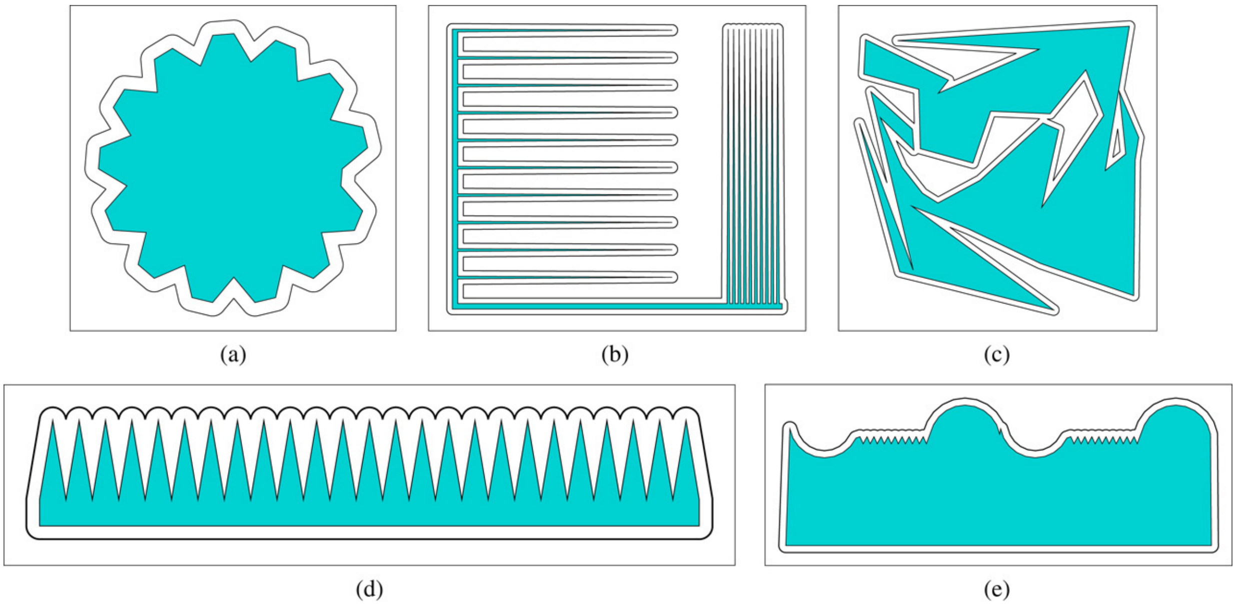


Fig. 3. Selected polygons used in the polygon-offset benchmarks: (a) `wheel`, (b) `spiked`, (c) `random`, (d) `comb`, (e) `chain`. The offset boundary is drawn in a thick black line.

The approximated angle ϕ'_1 is always very close to ϕ, namely $\Delta\phi_1 = \phi - \phi'_1$ is small, and we can safely bound $\tan(\Delta\phi_1)$ by $4 \cdot \tan(\frac{\phi - \phi'_1}{2})$. Note that $\Delta\phi_1 = \measuredangle(v_1, p_1, v'_1)$ is equal to the angle between the supporting lines of $v_1 v_2$ and $v'_1 w'$ (see Fig. 2 for an illustration), so as $1 < \frac{\ell}{\hat{\ell}} \ll 2$, the distance of w' from $v_1 v_2$ is upper bounded by:

$$\ell \tan(\Delta\phi_1) < \frac{\cdot(x_1 - x_2)}{\hat{\ell}^2(\hat{\ell} + (y_1 - y_2))} \cdot \delta\ell < \frac{2(x_1 - x_2)}{\hat{\ell}(\hat{\ell} + (y_1 - y_2))} \cdot \delta.$$

We conclude that if $\delta < \ell |\frac{\ell + (y_1 - y_2)}{2(x_1 - x_2)}| \cdot \varepsilon$, then this distance is smaller than ε. $\square$

An important property of our approximation algorithm is that it is *conservative*. That is, given a polygon P and an offset radius r it always computes a super-set $\tilde{P}_r$ of the exact offset polygon P_r. This property is crucial for many applications. For example, if we use our algorithm to approximate the forbidden configuration space of a round tool-tip moving amidst polygonal obstacles, we will never have "false positives" — namely, we will never declare a location of the tool center as collision-free when in fact it collides with an obstacles. We can have "false negatives", namely regarding a collision-free location as forbidden, but these errors are usually not crucial to the successful performance of the algorithm. Moreover, the probability of having "false negatives" can be made arbitrarily small by selecting a small enough approximation error ε, as proved above in Theorem 1.

In other applications, where one wishes the approximate offset polygon $\tilde{P}_r$ to be *contained* in the exact offset polygon P_r, we proceed as follows. Given a rational $\varepsilon > 0$, we let $\underline{r} = r - \varepsilon$ and apply the approximation algorithm with the offset radius $\underline{r}$. As the polylines that approximate the offset edges can lie at most $\underline{r} + \varepsilon = r$ from the original polygon edges, the result is guaranteed to be a subset of the exact offset polygon.

4. Experimental results

We compared the performance of the exact construction algorithm using the conic-traits class with the approximate construction scheme that based on the circle/segment traits-class. Table 1 summarizes the running times of the two approaches; the various input polygons and their offset boundaries are shown in Fig. 3 (the `country` polygon is a map of Israel).

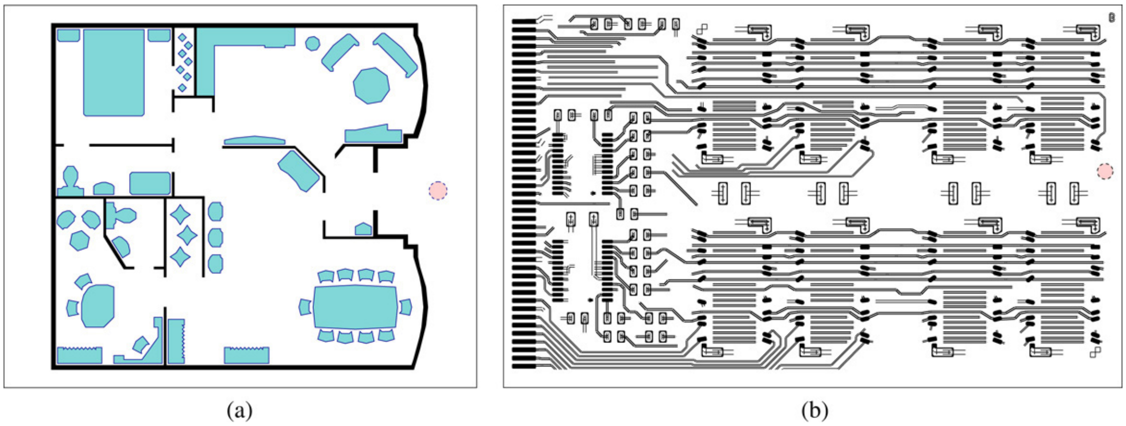
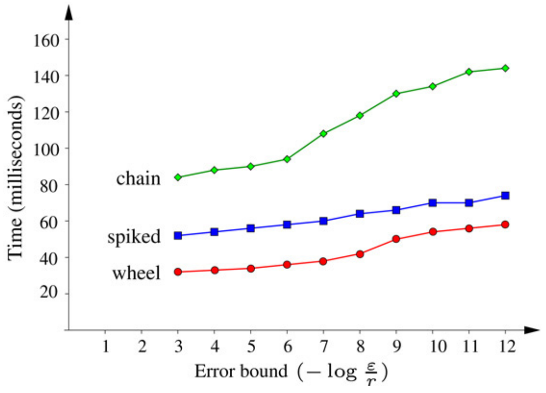


Fig. 4. Inputs of polygon sets: (a) `house`, (b) `VLSI`. The dashed disc that appear in each figure illustrate the offset radius we use in each case.

The offset radius chosen for each polygon is typically two orders of magnitude smaller than the size of the bounding box of the polygon. For the approximation scheme we selected the error bound accordingly. Table 1 includes the running time for error bounds of 10^{-7} and 10^{-10} times the offset radius. Our approximation scheme yields a significant speedup in the offset computations over the exact computation, especially for polygons that contain many small-size features (e.g., `chain`), spikes (e.g., `spiked`), or cavities (e.g., `random`). In such cases, the intermediate arrangement induced by the convolution cycle contains relatively many intersections; as computing and manipulating the intersection points of two arcs in the circle/segment traits-class is more efficient than in the conic-traits class, we can gain considerable speed-ups in these cases.

Note that as we decrease the error bound, we have to use rational numbers with longer bit-lengths (namely the sizes of the numerators and denominators increase), which incurs some running-time penalty. The graph below shows the running time of the approximate offset-computation process for three selected polygons as a function of the error bound (for $\varepsilon = 10^{-k}r$, where $k = 3, 4, \ldots, 12$). It shows that the running time is linear, or moderately super-linear, in $\log \frac{\varepsilon}{r}$.



As we have already mentioned, the Minkowski-sum package integrates well with other CGAL packages. In particular, it is possible to perform Boolean operations on offset polygons using the Boolean set-operations package [5]. The last set of experiments demonstrates the application of the union operation on a set of offset polygons, which has many important applications in many fields, such as computer-aided design and robotic motion planning.

Given a set of straight-edge polygons we compute the Minkowski sum of each polygon with a disc of radius r, either in an exact manner *or* using the approximation algorithm, and finally compute the union of all offset polygons. In the former case we use the traits-class for conic arcs to carry out the union computation in an exact manner, while in the latter case it is possible to use the circle/segment traits-class, as we operate on segments of rational lines and arcs of rational circles. Note that the results of the exact and the approximate computations are *not* equal in the geometric sense. Yet, we choose an approximation error-bound such that the two results are topologically equivalent (namely they contain the same number of polygons and the same number of holes in each polygon).

The `house` data set (Fig. 4(a)) consists of 55 polygons. We use an offset radius $r = 200$. It takes 0.376 s to compute the union of the approximate offset polygons (with an error bound of $\varepsilon = 10^{-6}r$), while the exact construction takes 3.176 s. The `VLSI` data set (Fig. 4(b)) is much larger and contains 22,400 polygons and straight line segments; here we use an offset radius of $r = 1$. The approximate computation (with $\varepsilon = 10^{-6}r$) takes 51.81 s, while constructing the union of the exact offset polygons takes 30.87 min.

5. Conclusions

We present the implementation of two algorithms that compute the offset of a simple polygon. Our software package, as does CGAL in general, employs the exact computation paradigm and provides robust implementations that can handle all inputs, including highly degenerate ones, yielding topologically correct results. The main contributions of our package is an algorithm that yields a conservative (and tight) approximation of the Minkowski sum of a polygon with rational vertices with a disc with a rational radius. This approximation scheme allows the robust handling of offset polygons in an efficient manner, using only exact rational arithmetic. It significantly reduces the processing time compare to handling exact offset polygons, which our software can also compute.

We also mention that our package include a robust implementation of the Minkowski-sum computation for two polygons, using either convex polygon decomposition or the convolution method. This package will be included in the next public release of CGAL (the forthcoming Version 3.3).

Acknowledgement

The work presented in this paper is part of the author's Ph.D. thesis, under the supervision of Prof. Dan Halperin.

Appendix A. Computing with rational line segments and circular arcs

A.1. One-root numbers

Given $\alpha, \beta, \gamma \in \mathbb{Q}$ with $\gamma > 0$, the real number $\alpha + \beta\sqrt{\gamma}$ is called a *one-root number*. We next study the properties of one-root numbers more closely and show how they can be handled using only rational arithmetic.

Lemma 2. *It is possible to determine the sign of a one-root number $x = \alpha + \beta\sqrt{\gamma}$ using only rational arithmetic.*

Proof. In case that $\text{sign}(\alpha) = \text{sign}(\beta)$, then this is also the sign of the entire expression x and we are done. Otherwise, we have to compare $|\alpha|$ and $|\beta|\sqrt{\gamma}$, such that the sign of x is the sign of the term whose absolute value is larger. But this is easily done by comparing α^2 and $\beta^2\gamma$, both are rational numbers. □

Lemma 3. *It is possible to compare two one-root numbers $x_1 = \alpha_1 + \beta_1\sqrt{\gamma_1}$ and $x_2 = \alpha_2 + \beta_2\sqrt{\gamma_2}$ using only rational arithmetic.*

Proof. We first note that if $\beta_2 = 0$, then $x_2 \in \mathbb{Q}$, and the comparison can be performed by evaluating the sign of the one-root number $x_1 - \alpha_2$. (Similarly, if $\beta_1 = 0$ we evaluate the sign of $x_2 - \alpha_1$.)

If both $x_1 = \alpha_1 + \beta_1\sqrt{\gamma_1}$ and $x_2 = \alpha_2 + \beta_2\sqrt{\gamma_2}$ are non-trivial one-root numbers, then comparing them is equivalent to comparing $\alpha_1 - \alpha_2$ and $\beta_2\sqrt{\gamma_2} - \beta_1\sqrt{\gamma_1}$. We therefore compute the sign of $\beta_2\sqrt{\gamma_2} - \beta_1\sqrt{\gamma_1}$ (this is easily done be comparing $\beta_1^2\gamma_1$ and $\beta_2^2\gamma_2$, if $\text{sign}(\beta_1) \neq \text{sign}(\beta_2)$) and check whether it is equal to the sign of $\alpha_1 - \alpha_2$. If the two terms have different signs, then we can deduce the comparison result at this stage. Otherwise, we continue by squaring both terms, such that the comparison result is equivalent to evaluating the sign of the one-root number $((\alpha_1 - \alpha_2)^2 - (\beta_1^2\gamma_1 + \beta_2^2\gamma_2)) + 2\beta_1\beta_2\sqrt{\gamma_1\gamma_2}$. (If both terms are negative, we have to negate the sign.) As Lemma 2 suggests, this can be done using only rational arithmetic. □

We finally note that in the general case, given two non-trivial one-root numbers $x_1 = \alpha_1+\beta_1\sqrt{\gamma_1}$ and $x_2 = \alpha_2+\beta_2\sqrt{\gamma_2}$ (with $\beta_1, \beta_2 \neq 0$), the numbers $x_1 \pm x_2$, $x_1 \cdot x_2$ and $\frac{x_1}{x_2}$ are *not* one-root numbers, unless of course $\gamma_1 = q^2\gamma_2$, where $q \in \mathbb{Q}$. In the next section we demonstrate how special care is taken so that no such operations are invoked when constructing an arrangement of circular arcs and line segments.

A.2. The circle/segment traits-class

The circle/segment traits-class included in the arrangement package of CGAL handles curves that are either arcs of rational circles, or segments of lines with rational coefficients. We next describe how the traits-class implements the various predicates and constructions needed for the aggregated construction of arrangements of circular arcs and line segments, based on the properties of one-root numbers.

To simplify the arrangement construction, it is easier to work with x-monotone circular arcs. A continuous planar curve is *x-monotone* if every vertical line intersects it at most once, so a line segment is always x-monotone (we consider vertical segments to be *weakly* x-monotone). A circular arc supported by the rational circle $(x - x_0)^2 + (y - y_0)^2 = R$ can be subdivided into three x-monotone arcs at most, depending on whether it contains the two points $(x_0 \pm \sqrt{R}, y_0)$ (a whole circle is subdivided into two x-monotone arcs). Moreover, we can label each x-monotone as a *"lower"* arc, if it lies below the horizontal line $y = y_0$, or as an *"upper"* arc, if it lies above this line.

The rest of the traits-class operations involve only points and x-monotone curves. We give the details of the operations that handle circular arcs and points with one-root coordinates. The treatment of line segments is simpler, and we omit its details here:

Compare xy: *Compare two points lexicographically, by their x and then by their y-coordinates*. As the point coordinates are one-root numbers, these operations can be easily performed using rational arithmetic, as stated by Lemma 3.

Point position: *Given an x-monotone arc C of the rational circle $(x - x_0)^2 + (y - y_0)^2 = R$ and a point $p = (\hat{x}, \hat{y})$, such that $\hat{x}$ is in the x-range of C (namely $\hat{x}$ lies between the x-coordinates of C's endpoints), determine whether p is above, below, or lies on C*. If C is a lower arc, it lies under the horizontal line $y = y_0$, so if $\hat{y} > y_0$, p obviously lies above C. Otherwise, we have to substitute p into the equation of the supporting circle, but as $\hat{x}$ and $\hat{y}$ are one-root numbers (and not necessarily rational numbers), we have to be a bit careful: we compare the one-root numbers $z_1 = (\hat{y} - y_0)^2$ and $z_2 = r^2 - (\hat{x} - x_0)^2$. p lies above C if $z_1 < z_2$, below it if $z_1 > z_2$, and on C in case of equality. Evaluating the predicate for an upper arc is symmetric.

Compare to right: *Given two x-monotone arcs C_1 and C_2 that share a common left endpoint $p = (\hat{x}, \hat{y})$, determine the relative position of the two curves immediately to the right of p*. First note that we can easily detect whether one of the arcs (say C_1) has a vertical tangent at p; in this case it is above C_2 if it is an upper arc and below it if it is a lower arc. Otherwise, the two arcs have a well-defined slope at p and we simply have to compare these slopes. The slope of C_j at p is given by $\frac{\hat{x}-x_j}{y_j-\hat{y}}$, where (x_j, y_j) is C_j's center. It is easy to show that comparing these two slopes is equivalent to comparing the two one-root numbers $y_2(\hat{x} - x_1) - y_1(\hat{x} - x_2)$ and $\hat{y}(x_2 - x_1)$. In case of equality, the two supporting circles are tangent at p, and we can simply compare their radii in order to determine their relative order near the tangency point.

Intersect: *Compute the intersection points of two x-monotone arcs C_1 and C_2*. We first compute the intersection points between the two supporting circles by solving two quadratic equations, whose solutions are the x and y-coordinates of the intersection point. For each intersection point $p = (\hat{x}, \hat{y})$, we have to check whether it really lies on both C_1 and C_2: for this, we simply check whether $\hat{y}$ is above or below the y-coordinate of the C_j's supporting circle (depending on whether C_j is an upper or a lower arc), and if so we just need to check whether it is in the x-range of the arc.

References

[1] Agarwal PK, Flato E, Halperin D. Polygon decomposition for efficient construction of Minkowski sums. Computational Geometry: Theory and Applications 2002;21:39–61.

[2] Berberich E, Eigenwillig A, Hemmer M, Hert S, Mehlhorn K, Schömer E. A computational basis for conic arcs and Boolean operations on conic polygons. In: Proc. 10th Europ. sympos. alg.. LNCS, vol. 2461. 2002. p. 174–86.

[3] de Berg M, van Kreveld M, Overmars M, Schwarzkopf O. Computational geometry: Algorithms and applications. 2nd ed. Berlin (Germany): Springer-Verlag; 2000.

[4] Flato E. Robust and efficient construction of planar Minkowski sums. Master's thesis. School of Computer Science, Tel-Aviv University; 2000. http://www.cs.tau.ac.il/CGAL/Theses/flato/thesis/.

[5] Fogel E, Wein R, Zukerman B, Halperin D. 2D regularized Boolean set-operations. In: CGAL Editorial Board, editor. CGAL-3.2 user and reference manual. http://www.cgal.org/Manual/3.2/doc_html/cgal_manual/Boolean_set_operations_2/Chapter_main.html, 2006.

[6] Guibas LJ, Ramshaw L, Stolfi J. A kinetic framework for computational geometry. In: Proc. 24th sympos. found. comput. sci. 1983. p. 100–11.

[7] Guibas LJ, Seidel R. Computing convolutions by reciprocal search. Discrete and Computational Geometry 1987;2:175–93.

[8] Halperin D. Arrangements. In: Goodman JE, O'Rourke J, editors. Handbook of discrete and computational geometry. 2nd ed. Chapman & Hall/CRC; 2004. p. 529–62 [chapter 24].

[9] Hert S, Hoffmann M, Kettner L, Pion S, Seel M. An adaptable and extensible geometry kernel. In: Proc. 5th internat. wrkshp. alg. eng. LNCS, vol. 2141. 2001. p. 79–90.

[10] Hoffmann CM. The problems of accuracy and robustness in geometric computation. IEEE Computer 1989;22(3):31–41.

[11] Hoffmann CM. Robustness in geometric computations. Journal of Computing and Information Science in Engineering 2001;1(2):143–55.

[12] Karamcheti V, Li C, Pechtchanski I, Yap C. A core library for robust numeric and geometric computation. In: Proc. 15th annu. ACM sympos. comput. geom. 1999. p. 351–9.

[13] Kaul A, Farouki RT. Computing Minkowski sums of plane curves. International Journal of Computational Geometry and Applications 1995; 5(4):413–32.

[14] Kaul A, O'Connor MA, Srinivasan V. Computing Minkowski sums of regular polygons. In: Proc. 3rd Canad. conf. comput. geom. 1991. p. 74–7.

[15] Kedem K, Livne R, Pach J, Sharir M. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. Discrete and Computational Geometry 1986;1:59–71.

[16] Lee I-K, Kim M-S, Elber G. Polynomial/rational approximation of Minkowski sum boundary curves. Graphical Models and Image Processing 1998;60(2):136–65.

[17] Maekawa T. An overview of offset curves and surfaces. Computer-Aided Design 1999;31(3):165–73.

[18] Mehlhorn K, Näher S. LEDA: A platform for combinatorial and geometric computing. Cambridge (UK): Cambridge University Press; 2000.

[19] Needham T. Visual complex analysis. Oxford (UK): Oxford University Press; 1997.

[20] Preparata FP, Shamos MI. Computational geometry: An introduction. Springer; 1985.

[21] Wein R. High-level filtering for arrangements of conic arcs. In: Proc. 10th Europ. sympos. alg. LNCS, vol. 2461. 2002. p. 884–95.

[22] Wein R. Exact and efficient construction of planar Minkowski sums using the convolution method. In: Proc. 14th Europ. sympos. alg. LNCS, vol. 4186. 2006. p. 829–40.

[23] Wein R, Fogel E, Zukerman B, Halperin D. 2D arrangements. In: CGAL Editorial Board, editor. CGAL-3.2 user and reference manual. http://www.cgal.org/Manual/3.2/doc_html/cgal_manual/Arrangement_2/Chapter_main.html, 2006.

[24] Wein R, Zukerman B. Exact and efficient construction of planar arrangements of circular arcs and line segments with applications. Technical report ACS-TR-121200-01. Tel-Aviv (Israel): Tel-Aviv University; March 2006.

[25] Yap CK. Robust geometric computation. In: Goodman JE, O'Rourke J, editors. Handbook of discrete and computational geometry. 2nd ed. Chapman & Hall/CRC; 2004. p. 927–52 [chapter 41].

[26] Yap CK, Dubé T. The exact computation paradigm. In: Du DZ, Hwang FK, editors. Computing in Euclidean geometry. 2nd ed. Lecture notes series on computing, vol. 1. Singapore: World Scientific; 1995. p. 452–92.