

International Journal of Computational Geometry & Applications
 © World Scientific Publishing Company

TWO APPROXIMATE MINKOWSKI SUM ALGORITHMS

Victor Milenkovic

*Department of Computer Science, University of Miami
 Coral Gables, FL 33124-4245, USA
 vjm@cs.miami.edu*

Elisha Sacks

*Computer Science Department, Purdue University
 West Lafayette, IN 47907-2066, USA
 eps@cs.purdue.edu*

We present two approximate Minkowski sum algorithms for planar regions bounded by line and circle segments. Both algorithms form a convolution curve, construct its arrangement, and use winding numbers to identify sum cells. The first uses the kinetic convolution and the second uses our monotonic convolution. The asymptotic running times of the exact algorithms are increased by $km \log m$ with m the number of segments in the convolution and with k the number of segment triples that are in cyclic vertical order due to approximate segment intersection. The approximate Minkowski sum is close to the exact sum of perturbation regions that are close to the input regions. We validate both algorithms on part packing tasks with industrial part shapes. The accuracy is near the floating point accuracy even after multiple iterated sums. The programs are 10% faster than direct floating point implementations of the exact algorithms. The monotonic algorithm is 42% faster than the kinetic algorithm.

Keywords: Minkowski sum; kinetic framework; robust computational geometry.

1. Introduction

We present two approximate Minkowski sum algorithms for planar regions bounded by line and circle segments. Minkowski sums are an important computational geometry concept whose applications include robot path planning, part layout, mechanism design, and computer graphics. Prior algorithms apply to polygonal regions. The extension to circle segments is of theoretical and practical interest because line and circle segments are closed under Minkowski sums, so other algorithms can iterate these primitives. Moreover, curved shapes are approximated to a given accuracy with quadratically fewer circle segments than line segments. Applications typically model curves with 4–6 decimal digits accuracy, so employing circles reduces the model size by a factor of 100–1000. Although spline models are even more compact, they are not closed under Minkowski sums.

The standard Minkowski sum algorithm¹ forms the kinetic convolution curve of the input regions, constructs its arrangement, and selects the cells with positive

winding numbers. The kinetic convolution is suboptimal in that the portion in the Minkowski sum interior is formed, arranged, and then discarded. We have developed a monotonic convolution² whose size is nearly optimal. In this paper, we describe floating point implementations of both algorithms.

The naive approach is to replace real arithmetic with floating point arithmetic. This approach is fast and accurate for most inputs, but is prone to failure when rounding errors alter the combinatorial structure of the output.

The mainstream implementation strategy is exact arithmetic using algebraic geometry (<http://cs.nyu.edu/exact>). Wein³ presents an exact kinetic algorithm for polygons. Exact Minkowski sum computation with circle segments has not been reported. We expect it would be slow, since exact arrangement computation, which is the dominant cost, is slow. For example, the latest exact algorithm⁴ takes 220 seconds to arrange 100 degree 6 curves, versus 22 second for our approximate algorithm⁵ (both using Linux and GNU C++ on similar processors). The running time of the exact algorithm is much larger for degenerate input, whereas ours is unchanged.

A second problem with exact Minkowski sums is that the algebraic degree and bit complexity of the output are higher than in the input. Output simplification is required to prevent exponential growth in iterated operations. Iteration is essential to applications algorithms for packing,^{6,7} path planning,⁸ and mechanical design.⁹ The state of the art addresses bit complexity growth in 2D polygons,^{10,11,12,13,14} 3D line segments,¹⁵ polyhedral subdivisions,¹⁶ and polyhedra defined by plane equations.¹⁷ Output simplification is an open problem for circle segments.

These problems motivate our approach of computing approximate Minkowski sums in floating point. The asymptotic running times of the exact algorithms are increased by a low-degree polynomial that is negligible in practice. We prove that the approximate sum is close to the exact sum of perturbation regions that are close to the input regions. The topology of the approximate sum can differ from that of the exact sum of the input regions or of the perturbed regions. Output simplification is automatic, since floating point has constant bit complexity.

We validate the algorithms on part packing tasks with industrial part shapes. The accuracy is near the floating point accuracy even after multiple iterated sums. The programs are 10% faster than naive floating point implementations. The monotonic algorithm is 42% faster than the kinetic algorithm. The C++ source code is available at <http://www.cs.miami.edu/~vjm/robust> for teaching or research.

The paper is organized as follows. Section 2 contains definitions. Section 3 describes our formulation of the kinetic Minkowski sum algorithm and Section 4 analyzes the approximate algorithm. Section 5 describes our monotonic convolution and Section 6 analyzes the approximate algorithm. Section 7 extends the approximate monotonic algorithm to compute accurately the free placements of a moving part with respect to a fixed part. Section 8 presents the validation results. The final section contains conclusions and plans for future work.

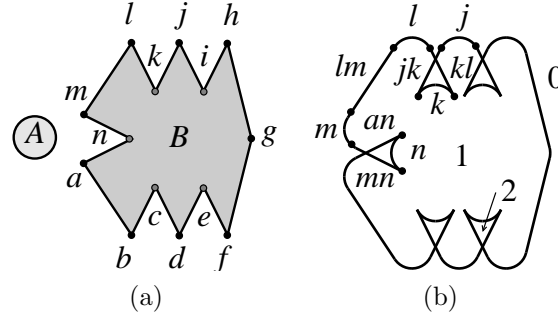


Fig. 1. Planar regions (a) and their kinetic convolution (b).

2. Definitions

A *planar region* is a closed, connected region whose boundary consists of simple loops of line and circle segments. We discuss the core case of a region whose boundary consists of one loop (Fig. 1a). The extension to multiple loops is straightforward.² The endpoints of segment a are designated $\text{tail}(a)$ and $\text{head}(a)$, so that the region interior is to the left when a is traversed from tail to head. A circle segment also has a center, $\text{center}(a)$, and a signed radius, $\text{radius}(a)$, that is positive/negative when a is *convex/concave*, meaning that the center is to the left/right when the segment is traversed from tail to head.

The rightward (outward) normal vector of a line segment, a , is $(y, -x)$ with $(x, y) = \text{head}(a) - \text{tail}(a)$. The rightward normal vector of a circle segment at p is $(p - \text{center}(a))/\text{radius}(a)$. The rightward normal angles at $\text{tail}(a)$ and $\text{head}(a)$ are called $\alpha(a)$ and $\beta(a)$. The *angle interval* of a is $[\alpha(a), \beta(a)]$ when a is convex and is $[\beta(a), \alpha(a)]$ otherwise. The point on a with angle θ is called $\text{point}(a, \theta)$.

The inputs to the kinetic convolution¹ are polygonal tracings. Imagine a wheelchair driving along the boundary of a polygon. When it reaches a vertex, it executes a turn to orient itself along the next edge. The turns eliminate boundary orientation discontinuities. We represent the turn from a to b with a *turn segment*: a circle segment, e , with $\text{center}(e) = \text{tail}(e) = \text{head}(e) = \text{head}(a)$, $\text{radius}(e) = 0$, $\alpha(e) = \beta(a)$, and $\beta(e) = \alpha(b)$. A polygonal tracing is represented by a smoothed region in which turn segments are inserted between consecutive boundary segments. Smoothed regions are abbreviated to regions from here on.

In the monotonic algorithm, we split circle segments at vertical turning points to obtain x -monotonic segments. A monotonic segment is *upper/lower* when the region interior is below/above it. A vertical segment that joins two upper/lower segments is labeled upper/lower. Otherwise, it is labeled upper/lower when its normal points right/left. An *upper/lower chain* is a maximal sequence of upper/lower segments. The region boundary decomposes into chains that meet at turning points and that are partially ordered in y : they are ordered in y iff they overlap in x . In Fig. 1a, the B upper chains are $ghijklm$ and na , and the lower chains are $abcdefg$ and mn .

3. Kinetic algorithm

The kinetic convolution, $A \otimes_k B$, of regions A and B is the sum of all pairs of boundary points with equal rightward normal angles. (The subscript in \otimes_k distinguishes this convolution from other convolutions, defined later.) The sum consists of line and circle segments called *sum segments*. Boundary segments $a \in A$ and $b \in B$ generate a sum segment, $e = a + b$, when their angle intervals overlap. In Fig. 1b, there is one A segment and selected sum segments are labeled with their B segments. For line segments a and b with $\alpha(a) = \alpha(b)$, e is the line segment from $\text{tail}(a) + \text{tail}(b)$ to $\text{head}(a) + \text{head}(b)$. For a convex circle segment a and a line segment b with $\alpha(b) \in [\alpha(a), \beta(a)]$, e is the line segment from $\text{point}(a, \alpha(b)) + \text{tail}(b)$ to $\text{point}(a, \alpha(b)) + \text{head}(b)$. When a is concave, the endpoints are interchanged. For a and b convex/concave circle segments with shared angle interval $[\alpha, \beta]$, e is the circle segment from $\text{point}(a, \alpha) + \text{point}(b, \alpha)$ to $\text{point}(a, \beta) + \text{point}(b, \beta)$ with $\text{radius}(e) = \text{radius}(a) + \text{radius}(b)$ and $\text{center}(e) = \text{center}(a) + \text{center}(b)$. When one segment is convex and the other is concave, the endpoints are interchanged.

The Minkowski sum, $A \oplus B$, is obtained by constructing the arrangement of $A \otimes_k B$ and computing cell winding numbers (called crossing numbers in our monotonic convolution paper²). The *winding number* of a cell is the oriented sum of the edge crossings along any path from an interior point to the unbounded cell. An edge contributes 1 when it is crossed in the direction of its rightward normal and contributes -1 otherwise. Winding numbers are computed by assigning the unbounded cell 0 and traversing the other cells. When cell b is visited from cell a with winding number c , its winding number is $c - 1$ when ab is crossed in the rightward normal direction and is $c + 1$ otherwise. In Fig. 1b, the large inner cell has winding number 1 and the five small cells have winding number 2. If point u lies in a cell with winding number K , $A \cap (-B + u)$ has K connected components. The boundary of $A \oplus B$ consists of the segments that separate cells of zero and positive winding number.

Our formulation yields the same sum segments as does the original kinetic convolution.¹ It remains to show that the rightward normal vectors have the correct orientation. Figure 2 illustrates the four cases that arise for upper circle segments; a similar analysis applies to lower circle segments and to line segments. If $e = e_1 + e_2$ for convex e_1 and e_2 (a), moving $-R_2$ down from point u on e adds a component to $R_1 \cap (-R_2 + u)$, so the e normal should point up. Since $h_1 < t_1$ and $h_2 < t_2$, $\text{head}(e) = h_1 + h_2 < t_1 + t_2 = \text{tail}(e)$, which implies an upward normal. If a small convex circle segment meets a large concave circle segment (b), moving $-R_2$ down adds a component. Since $t_{2x} - h_{2x} < t_{1x} - h_{1x}$, $\text{head}(e) = h_1 + t_2 < h_1 + t_2 = \text{tail}(e)$, so the normal is upward. If a large convex circle segment meets a small concave circle segment (c), two regions merge (arrows) as $-R_2$ moves down, so the e normal should point down. Since $t_{1x} - h_{1x} < t_{2x} - h_{2x}$, $\text{tail}(e) = t_1 + h_2 < t_2 + h_1 = \text{head}(e)$. If two concave circle segments meet (d), two regions merge (arrows) and $\text{tail}(e) = h_1 + h_2 < t_1 + t_2 = \text{head}(e)$.

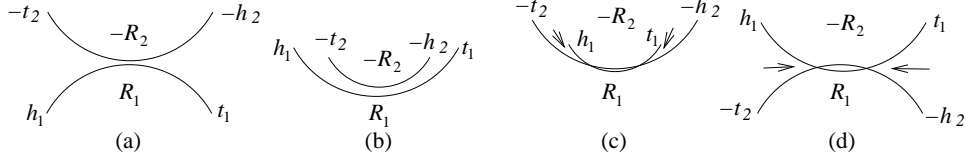


Fig. 2. The four types of circle segment sums.

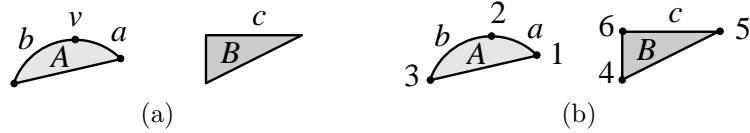


Fig. 3. Degenerate input (a) and symbolic perturbation (b).

The kinetic algorithm assumes generic input. A degeneracy occurs when incident segments, a and b , meet tangentially at a vertex, v , whose normal angle equals that of a line segment, c , on the other region boundary (Fig. 3a). The sum $v + c$ is generated as $a + c$ and as $b + c$ because the a and b angle intervals intersect the c angle interval. The extra sum invalidates the winding numbers.

We handle degeneracy by symbolic perturbation. Assign the vertices of A integer labels in counterclockwise order starting from 1. Assign the vertices of B labels starting one after the last A label. Order vertices with equal angles by label. The vertex order is realized by perturbing angle α with label l to $\alpha + l\eta$ with η sufficiently small. After perturbation, the v angle is strictly ordered with respect to the two c angles, so the a and b intervals cannot both overlap the c interval. In our example, the b interval, $[90^\circ, 180^\circ]$, contains the c interval, $[90^\circ, 90^\circ]$, because $2 < 5$, whereas the a interval, $[45^\circ, 90^\circ]$, is disjoint from the c interval because $2 < 5 < 6$ (Fig. 3b). Moreover, perturbed line segments have disjoint angle intervals, so we avoid sums of line segments, which simplifies the algorithms.

4. Approximate kinetic algorithm

The inputs to the approximate kinetic algorithm are the segment endpoints and the signed circle segment radii in floating point. In rare cases, the input is modified to ensure an accurate convolution. The convolution is formed in floating point. The arrangement is constructed with our approximate algorithm.⁵ Winding numbers are defined because the approximate convolution consists of loops of sum segments and the arrangement algorithm preserves segment incidence. They can be negative, which is impossible in the exact case. We assign the cells with nonzero winding number to the Minkowski sum. The extra cells do not increase the error (Sec. 4.3).

The asymptotic running time of the approximate algorithm matches that of the exact algorithm, plus $km \log m$ time to arrange m sum segments with k the number

of segment triples that are in cyclic vertical order. The parameter k quantifies the combinatorial error in the arrangement due to numerical error in segment intersection. The approximate arrangement algorithm is much faster than exact algorithms in practice, as discussed above.

We bound the error in the boundary segment normal angles and in the sum segments in terms of the floating point accuracy (Sec. 4.1). The Minkowski sum error is mainly due to the angles, since a tiny angle error can cause a large change in the convolution structure by altering the set of sum segments. We address this sensitivity by defining a perturbed input, called a realization, that is near the actual input and that has the same angles (Sec. 4.2). We prove that the approximate convolution and Minkowski sum are accurate (although not exact) for the realization (Sec. 4.3).

4.1. *Sum segments*

We employ floating point arithmetic operators, square roots, and trigonometric functions. The relative error in arithmetic operations is bounded by 2^{-b} with b the number of bits in the mantissa, which is about 10^{-16} in double float. The other operations can be slightly less accurate. We assume a bound of ϵ for all operations.

Suppose $f(x) = y$, but the computed value is $y + e$ with error e . When $g(f(x))$ is computed, the error consists of the g rounding error plus the propagated error, $g(y) - g(y + e)$, due to the error in the g input. The ratio of propagated error to input error is called the condition number. It is well approximated by $yg(y)/g'(y)$ by Taylor's theorem, since the error term is $O(e^2)$ and e is tiny. When the condition number is bounded, the g error is $O(\epsilon)$. We say that g is well-conditioned. In a sequence of operations, the error in each step is propagated to subsequent steps. When the sequence length is bounded and the operations are well-conditioned, the final error is $O(\epsilon)$.

We compute sum segments via short sequences of operations. Calculus shows that the operations are well-conditioned when certain bad values are excluded. In this section, we show that our computations avoid these values. Hence, the computation error is $O(\epsilon)$. This is a relative error that is proportional to the sum segment length. We remove the lengths from our analysis by scaling the input to the unit box. Section 8 shows that the actual error is a small multiple of ϵ .

The boundary segment normal angles are computed as follows. For a line segment, s , with tail t and head h , $\alpha(s) = \beta(s) = \gamma$ with $\gamma = \arctan(\frac{h_y - t_y}{t_x - h_x})$ (Fig. 4a). For a circle segment, γ is the normal angle of the secant (Fig. 4b). Compute $\delta = \arcsin(d/r)$ with $d = \|h - t\|/2$ and with $r = \text{radius}(s)$. The normal angles are $\alpha(s) = \gamma - \delta$ and $\beta(s) = \gamma + \delta$. The only possible bad values are $h_y - t_y$ and $t_x - h_x$, since subtraction is ill-conditioned when the arguments are nearly equal. But the arguments to our subtractions are inputs, so there is no error to propagate.

A circle segment point, $p = \text{point}(s, \theta)$, is computed as follows. We have $p_x = o_x + r \cos \theta$, $t_x = o_x + r \cos \alpha$, and $h_x = o_x + r \cos \beta$ with $o = \text{center}(s)$ and $r =$

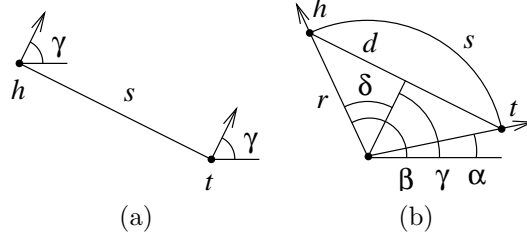


Fig. 4. Normal angle computation for line (a) and circle (b) segments.

radius(s). Subtracting the second equation from the first yields $p_x - t_x = r(\cos \theta - \cos \alpha)$ and subtracting the second from the third yields $h_x - t_x = r(\cos \beta - \cos \alpha)$. Eliminating r yields $p_x = t_x + d_x$ with $d_x = (h_x - t_x)(\cos \theta - \cos \alpha)/(\cos \beta - \cos \alpha)$. This formula is ill-conditioned when a cosine has a small argument or when two nearly equal cosines are subtracted. We obtain the well-conditioned formula

$$d_x = \frac{\sin \frac{\theta+\alpha}{2} \sin \frac{\theta-\alpha}{2}}{\sin \gamma \sin \delta} (h_x - t_x) = \frac{\sin \frac{\theta+\alpha}{2}}{2 \sin \gamma \cos \frac{\delta}{2}} (h_x - t_x)$$

using $\cos u - \cos v = -2 \sin \frac{u+v}{2} \sin \frac{u-v}{2}$, $\theta - \alpha = \delta$, and $\sin u = 2 \sin \frac{u}{2} \cos \frac{u}{2}$. A similar derivation yields $p_y = t_y - d_x \cot \frac{\theta+\alpha}{2}$, which is well-conditioned.

Every sum segment endpoint has the form $p + q$ where $p = \text{point}(s, \theta)$ and q is a boundary segment endpoint. This formula is ill-conditioned when $p_x \approx -q_x$ or $p_y \approx -q_y$. We compute $p_x = t_x + d_x + q_x$ by two applications of Dekker's method¹⁸ for computing the roundoff error of a floating-point operation. The computed sum is the round of the exact sum, hence has no propagated error. We compute p_y likewise. Circle segment centers are not computed or used.

Input modification There are two cases where input segments are modified to ensure Minkowski sum accuracy.

A circle segment, s , is replaced by its secant line segment when δ rounds to zero, so $\alpha(s) = \beta(s)$. The distance between the segment and the secant is bounded by ϵ . We calculate the maximum distance, which occurs at $p = \text{point}(\gamma, s)$, as follows:

$$\begin{aligned} (p - t) \cdot \begin{bmatrix} \cos \gamma \\ \sin \gamma \end{bmatrix} &= d_x \cos \gamma - d_x \cot \frac{\gamma + \alpha}{2} \sin \gamma = d_x \frac{\cos \gamma \sin \frac{\gamma+\alpha}{2} - \sin \gamma \cos \frac{\gamma+\alpha}{2}}{\sin \frac{\gamma+\alpha}{2}} \\ &= d_x \frac{-\sin \frac{\delta}{2}}{\sin \frac{\gamma+\alpha}{2}} = (h_x - t_x) \frac{\sin \frac{\gamma+\alpha}{2} - \sin \frac{\delta}{2}}{2 \sin \gamma \cos \frac{\delta}{2} \sin \frac{\gamma+\alpha}{2}} = (t_x - h_x) \frac{\tan \frac{\delta}{2}}{2 \sin \gamma} \end{aligned}$$

where the third step uses $\sin(u - v) = \sin u \cos v - \sin v \cos u$ and $\gamma - \alpha = \delta$. The maximum distance is $0.25\delta(t_x - h_x)/\sin \gamma + O(\delta^2)$ by Taylor's theorem. We drop the second term because $\delta < \epsilon \approx 10^{-16}$. The maximum is bounded by 0.5ϵ because $|t_x - h_x| \leq 1$ in the unit box and $|\sin \gamma| \geq 0.5$.

A line segment, s , is modified when $\text{tail}(s)_x \neq \text{head}(s)_x$ and $\alpha(s)$ equals 0° or 180° . The normal is horizontal because $|\text{tail}(s)_x - \text{head}(s)_x|$ is of order ϵ . Segment s

is split into a horizontal from $\text{tail}(s)$ to $p = (\text{head}(s)_x, \text{tail}(s)_y)$ and a vertical from p to $\text{head}(s)$. The maximum error occurs at p and is bounded by ϵ .

4.2. Realization

We define realization regions A^* and B^* for input regions A and B . We discuss A^* only, since B^* is defined identically. As explained above, the goal is for A^* to have the same normal angles as A at endpoints and at interior circle segment points that contribute to sum segment endpoints. Moreover, we want the same x coordinates at these points in preparation for the monotonic algorithm. The realization consists of four constructive steps that transform A to A^* .

Step 1 transforms the interior circle segment points to endpoints by splitting each boundary segment, s , at every $p = \text{point}(s, \theta)$ such that $p + q$ is a sum segment endpoint. The split segments are realized in steps 2–4. Figure 5 illustrates splitting. The four boundary segments of A become seven segments after splitting the upper/lower segment at the normal angles of the upper/lower triangle sides. The three interior points with these angles become segment endpoints.

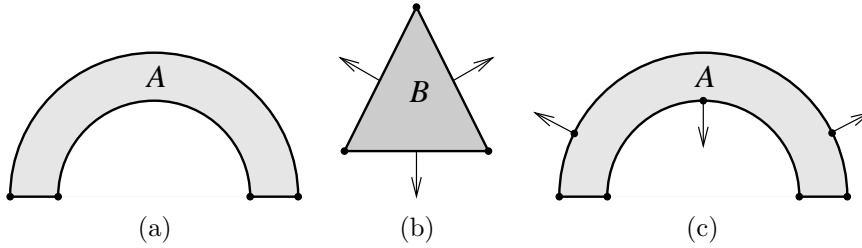


Fig. 5. Circle segment splitting: (a) region A ; (b) region B ; (c) output.

Step 2 realizes the boundary segment chains. Each segment, s , in a chain is adjusted to make its endpoint coordinates and angles consistent. Let $\gamma = \arctan\left(\frac{h_y - t_y}{t_x - h_x}\right)$ as above and let $\phi = (\alpha(s) + \beta(s))/2$. We have $\gamma = \phi$ for any actual line or circle segment, but the approximate values for s can be unequal. We set $\text{head}(s)_y$ to $\text{tail}(s)_y - (\text{head}(s)_x - \text{tail}(s)_x) \cot \phi$. Figure 6 illustrates line and circle segment adjustment. For a vertical segment, adjustment is unnecessary (and undefined) because $\gamma = \phi$ by construction. A chain s_1, \dots, s_k is realized by adjusting the s_i in order and for $i > 1$ translating s_i vertically so that $\text{tail}(s_i)_y = \text{head}(s_{i-1})_y$.

Step 3 vertically shifts the realization chains to eliminate intersections. A chain cannot self-intersect because its segments are x -monotonic with pairwise disjoint domain interiors. But two chains can intersect after step 2. Visit the chains in an order consistent with their partial y order and shift each chain upward until it no longer intersects any chain below it. Step 4 converts the shifted chains to a loop by inserting a vertical segment between realization chain endpoints that correspond to the same input chain endpoint.

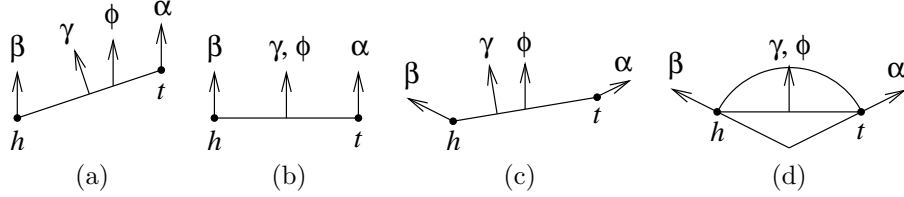


Fig. 6. Segment realization: (a) line segment; (b) realization; (c) circle segment; (d) realization.

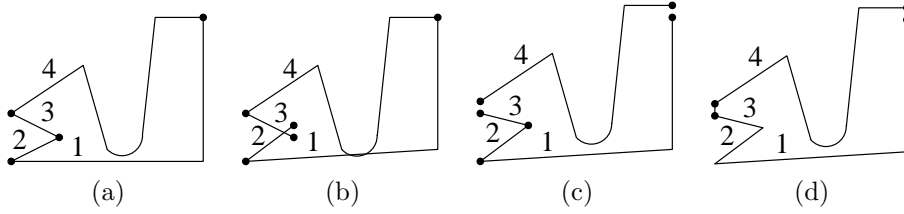


Fig. 7. Region (a), realization chains (b), shifting (c), and verticals (d).

Figure 7 illustrates steps 3 and 4. The region has four chains that are labeled in increasing y order. Realization chains 2 and 3 intersect near an endpoint, as is typical. Chains 1 and 4 intersect in their interiors. The region boundary is close to self-intersection, since the vertical distance between the chains is tiny by Thm. 1.

Theorem 1. *The vertical distance between A and A^* is $O(n\epsilon)$.*

Proof. Here n is the number of A boundary segments. Step 1 introduces no error. In step 2, the $\text{head}(s_i)_y$ adjustment is linear in the segment x extent because $\cot \phi$ is well-conditioned. The overall step 2 error is linear in the x extent of the chain because adjustments are propagated to subsequent segments. In step 3, the vertical shift in each chain is bounded by the maximum step 2 shift over the chains below, since the input chains are disjoint. The maximum is bounded by the maximum chain length times ϵ , which is $O(n\epsilon)$ because the segments are in the unit box. The two input modifications add $O(\epsilon)$ error per segment. \square

4.3. Error analysis

We use realizations to prove that our approximate Minkowski sums are accurate. The approximate sum of regions A and B is written as $A \hat{\oplus} B$; likewise $A \hat{\otimes}_k B$ is the approximate kinetic convolution and $\hat{e} = a \hat{+} b$ is an approximate sum segment. When we compute the arrangement of $A \hat{\otimes}_k B$ with our approximate algorithm,⁵ the output is correct for a perturbation of the input, $A \tilde{\otimes}_k B$, that preserves segment incidences. The perturbation size is $O(\epsilon + km\epsilon)$ for m sum segments with k inconsistencies; in practice it is $O(\epsilon)$. Let μ bound the sum of the perturbation

magnitude and the realization error of A^* and B^* . We prove that $A\hat{\oplus}B$ is μ -close to $A^* \oplus B^*$ in the Hausdorff metric.

Lemma 1. $A^* \otimes_k B^*$ is $O(n\epsilon)$ close to $A\hat{\otimes}_k B$.

Proof. Define a map, ψ_A , from the boundary of A^* onto the boundary of A : map point (a^*, θ) to point (a, θ) for a circle segment, map a line segment linearly, and contract a step 4 vertical to the common map of its endpoints. Define ψ_B likewise. Map $s^* = a^* + b^*$ to $\psi_A(a^*) + \psi_B(b^*)$. The map is a surjection from $A^* \otimes_k B^*$ onto $A\hat{\otimes}_k B$ and the distance from a point to its image is $O(n\epsilon)$ by Thm. 1. \square

Theorem 2. $A\hat{\oplus}B$ is μ -close to $A^* \oplus B^*$.

Proof. If $t \in A^* \oplus B^*$ and $t \notin A\hat{\oplus}B$, the winding number of t is positive in $A^* \otimes_k B^*$ and is zero in the approximate arrangement of $A\hat{\otimes}_k B$, hence is zero in the perturbed convolution $A\hat{\otimes}_k B$. Thus, t lies on opposite sides of e^* and \tilde{e} for some sum segment e . Since e^* and \tilde{e} are μ -close by Lemma 1, t is within μ of \tilde{e} . The $A\hat{\otimes}_k B$ region on the other side of \tilde{e} has winding number ± 1 , hence is in $A\hat{\oplus}B$. Thus, t is within μ of $A\hat{\oplus}B$.

If $t \in A\hat{\oplus}B$ and $t \notin A^* \oplus B^*$, the winding number of t is nonzero in $A\hat{\otimes}_k B$ and is zero in $A^* \otimes_k B^*$, so t is within μ of $A^* \otimes_k B^*$ as above. Since every point in the convolution is a limit point of the Minkowski sum, t is within μ of $A^* \oplus B^*$. \square

5. Monotonic convolution

A large portion of the kinetic convolution can lie in the Minkowski sum interior. Figure. 8 shows an example involving a convex polygon, A , and an unbounded polygonal region, B , whose boundary is a convex polygon. The Minkowski sum (b) is an unbounded polygonal region whose boundary is a convex polygon. Most of the kinetic convolution (c) lies in the interior of this region. A smaller *convex convolution* (d) is obtained by excluding convexly incompatible segments.¹⁹ Boundary segments $a \in A$ and $b \in B$ are convexly incompatible when the sum of their curvatures is negative. The curvature of s is 0 for a line segment and is $1/\text{radius}(s)$ otherwise. If points $p \in a$ and $q \in b$ have equal normals, A and $-B+p+q$ have a point of tangency and intersect in the neighborhood of this point, so $p+q$ is in the Minkowski sum interior (Fig. 9). The convex convolution, $A\otimes_c B$, is the set of convexly compatible sums. It is a superset of the $A \oplus B$ boundary.

The kinetic convolution has two advantages over the convex convolution for Minkowski sums. The convex convolution does not define winding numbers, so each arrangement cell must be classified by selecting a point t and testing whether A intersects $-B+t$. This takes $O(n \log n)$ time with n the input size, versus constant time with the kinetic convolution. The convex convolution topology is sensitive to changes in the sum segment endpoints and crossings, so an approximate algorithm

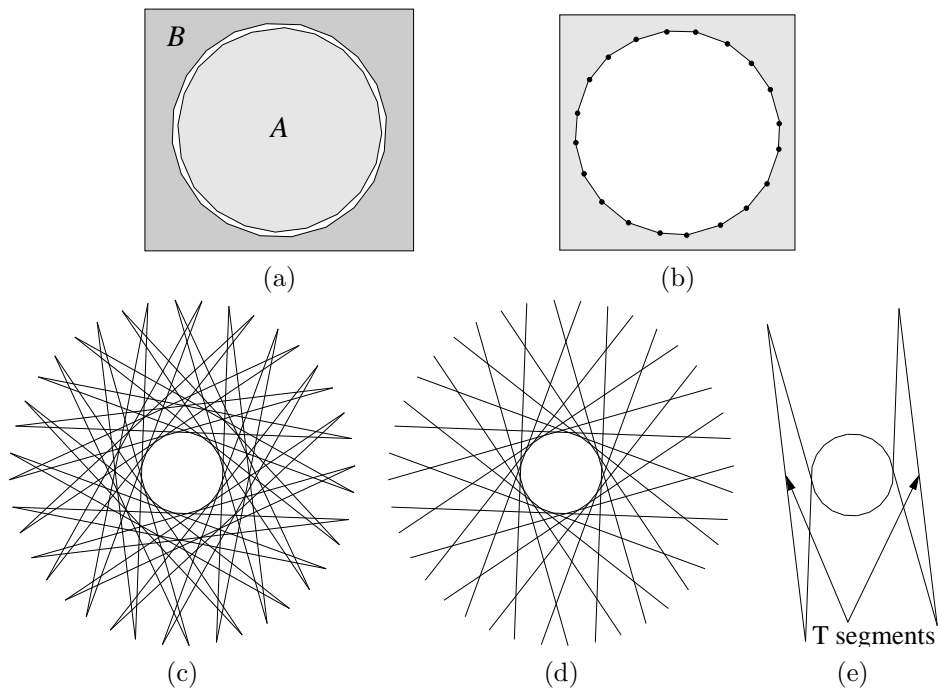


Fig. 8. Convolution comparison: (a) parts (actual size); (b) Minkowski sum ($\times 25$); (c) kinetic convolution ($\times 10$); (d) convex convolution ($\times 10$); (e) monotonic convolution ($\times 10$).

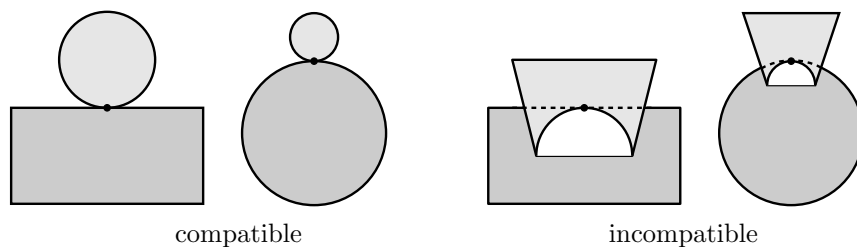


Fig. 9. Curvature test.

is prone to unbounded error. We have shown in Sec. 4.3 that the kinetic convolution does not have this problem.

Our monotonic convolution² combines the advantages of the kinetic and convex convolutions. It defines winding numbers that determine Minkowski sum membership. Yet it is only slightly larger than the convex convolution in the worst case and is often much smaller (Fig. 8e). We briefly describe the monotonic convolution (leaving the proofs to our prior paper), present an approximate version, and prove a stronger error bound than that of the approximate kinetic algorithm (Sec. 7).

5.1. Definition

The Π shape of an upper/lower chain is the region below/above it, which is bounded by the chain and by the downward/upward vertical rays at its endpoints. In Fig. 10a, B has upper $\Pi(ghijklm)$ with downward rays at g and m , upper $\Pi(na)$, lower $\Pi(abcdefg)$ with upward rays at a and g , and lower $\Pi(mn)$. For upper chains U and V , the Minkowski sum $\Pi(U) \oplus \Pi(V)$ is an upper Π shape whose boundary is the upper envelope of $\Pi(U) \otimes_c \Pi(V)$. The U, V crust is the upper envelope oriented right to left. For lower chains L and M , $\Pi(L) \oplus \Pi(M)$ is a lower Π shape whose crust is the lower envelope of $\Pi(L) \otimes_c \Pi(M)$ oriented left to right.

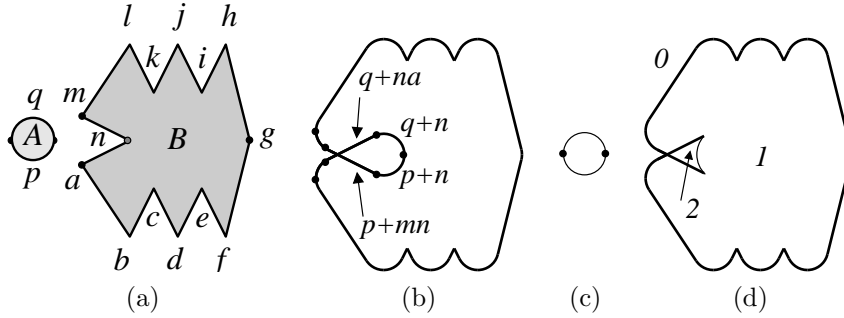


Fig. 10. (a) Planar regions; (b) crust segments; (c) T segments; (d) monotonic convolution.

The monotonic convolution, $A \otimes_m B$, is the multi-set union of the A, B crusts and T segments. The T segments are the boundaries of the regions $p_1 + B$ and $A + p_2$ for every *concave* turning point $p_1 \in A$ and $p_2 \in B$. A turning point is concave when its lower chain is above its upper chain, like n in Fig. 10a. In the multi-set union, a crust segment cancels a T segment when they lie on the same line or circle and are related. Related means that the crust segment is generated by chains C and D , and the T segment is generated by C and a concave endpoint of D . The identical portions of the two segments are deleted. Figure 10 illustrates these concepts. The union of the crust segments $q + n$ and $p + n$ with the T circle segment is the concave side of the region with winding number 2.

5.2. Monotonic algorithm

The monotonic convolution is computed as follows. Split the region boundaries into chains. Smooth an upper/lower chain to $0/180$ at its tail and to $180/0$ at its head. Compute the crusts. Form the sum segments for the convexly compatible pairs using the Sec. 3 definitions. Compute envelopes for the non-vertical segments. Insert vertical segments in the envelopes to close vertical gaps and to connect the left/right boundary points to the sum of the left/right chain endpoints. Compute the T segments from the definition. Form the multiset union. Arrange the resulting

monotonic convolution and assign winding numbers, as in the kinetic algorithm.

6. Approximate monotonic algorithm

The approximate monotonic algorithm is a floating point implementation of the exact algorithm with one modification. Two upper/lower segments are deemed convexly compatible when the head of their sum segment is to the left/right of the tail. The endpoint rule is equivalent to the curvature rule in exact arithmetic. Equivalence follows directly from the sum segment definitions in Sec. 3. In floating point, the endpoint rule prevents gaps in the convolution that can cause unbounded errors in the Minkowski sum.

As in the kinetic case, the asymptotic running time of the approximate algorithm matches that of the exact algorithm plus $km \log m$ time to arrange m sum segments with k the number of segment triples that are in cyclic vertical order.² The alternate convex compatibility test takes constant time. The envelopes are computed by the standard divide and conquer algorithm: split the input in half, compute envelopes recursively, and merge. We implement the merge with our approximate sweep algorithm. The running time is linear in the input size because the sweep contains at most two segments at all times.

6.1. Error analysis

We prove that the vertical error in the crusts is $O(n\epsilon)$ for n boundary segments. The T segment vertical error is trivially $O(\epsilon)$. The Sec. 4.3 error bounds transfer to the monotonic algorithm with Thm. 3 replacing Thm. 1 and with the monotonic convolution replacing the kinetic convolution.

We treat upper chains U and V ; lower chains are analogous. The U, V crust, C , is the approximate upper envelope of the approximate convex convolution $\Pi(U) \hat{\otimes}_c \Pi(V)$. Call a segment forward/backward when its head is left/right of its tail. The sum segment $s = a + b$ is included in the envelope when it is forward. Step 1 of the realization (Sec. 4.2) generates segments a_i^* and b_i^* from the portions of a and b in the shared angle interval, $[\alpha, \beta]$, as shown in Fig. 11. These segments generate a chain, d^* , of sum segments, $c_i^* = a_i^* + b_i^*$, and the convexly compatible c_i^* form $C^* = \Pi(U^*) \otimes_c \Pi(V^*)$. These are also the forward c_i^* , since the endpoint rule and the convexity rule are equivalent in exact arithmetic. Even when s is backward, hence is not in $\Pi(U) \hat{\otimes}_c \Pi(V)$, d^* can contain forward segments, such as c_2^* in Fig. 11c, which are in $\Pi(U^*) \otimes_c \Pi(V^*)$. Despite these missing segments, the approximate upper envelope is close to the exact one.

Theorem 3. *The vertical distance between C and C^* is $O(n\epsilon)$.*

Proof. The first step is to show that C is at most $O(n\epsilon)$ above C^* . C is never above the approximate upper envelope of $\Pi(U) \hat{\otimes}_k \Pi(V)$ because $\Pi(U) \hat{\otimes}_c \Pi(V)$ is a subset of $\Pi(U) \hat{\otimes}_k \Pi(V)$. The vertical distance between the approximate envelope

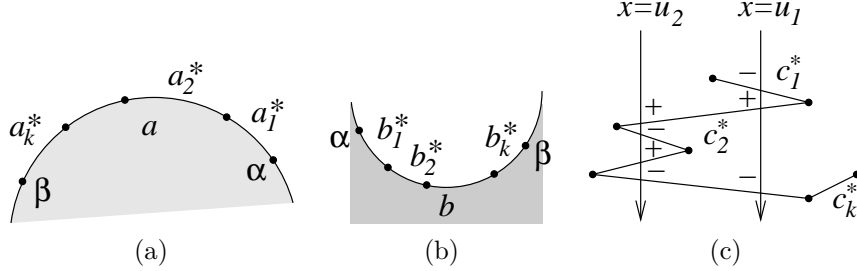


Fig. 11. Upper segments (a-b) and prototypical sum segment chain (c).

of $\Pi(U) \hat{\otimes}_k \Pi(V)$ and the envelope of $\Pi(U^*) \otimes_k \Pi(V^*)$ is $O(n\epsilon)$ by Lemma 1, since the error in envelope computation is $O(\epsilon)$. The envelope of $\Pi(U^*) \otimes_k \Pi(V^*)$ equals the envelope of $\Pi(U^*) \otimes_c \Pi(V^*)$, which is C^* . Since C is never above a curve that is $O(n\epsilon)$ close to C^* , it is at most $O(n\epsilon)$ above C^* .

The second step is to show that C is at most $O(n\epsilon)$ below C^* . Suppose c_i^* is forward and is defined at $x = u$. If s is forward, u is in the exact endpoint x interval of s by definition. Since the computed endpoint x coordinates equal the rounded exact values (Sec. 4.1), u is in the computed endpoint x interval. The vertical distance between it and c_i^* is $O(n\epsilon)$ by Lemma 1. For s backward, we show that another forward segment of $\Pi(U) \hat{\otimes}_c \Pi(V)$ is close to c_i^* at $x = u$.

Consider a downward vertical ray at $x = u$. If u is inside the d^* endpoint x interval, the ray intersects one more backward than forward d^* segment (2 versus 1 at u_1 in our example) because d^* is backward. The net contribution to the winding number of $\Pi(U^*) \otimes_k \Pi(V^*)$ is -1 because backward/forward segments contribute $-1/1$. If u is outside the endpoint x interval, the ray intersects the same number of forward and backward d^* segments (2 at u_2 in our example) because both d^* endpoints are on the same side of the ray. The net contribution is zero to the winding number of the cell just below the lowest d^* crossing. In both cases, the winding number is positive below the first crossing: $\Pi(U^*) \oplus \Pi(V^*)$ is a Π shape, so every point below the upper envelope is inside. This means that the ray crosses a forward chain no later than it crosses the lowest d^* segment.

The forward chain has the form $e^* + f^*$ with $g = e + f$ a forward sum segment. As above, u is in the endpoint x interval of g . By Lemma 1, g is at most $O(n\epsilon)$ below the lowest d^* segment at $x = u$. The y values of d^* at $x = u$ are $O(n\epsilon)$ close to each other by Lemma 1 because all the segments realize portions of s . Hence, g is at most $O(n\epsilon)$ below d^* . \square

7. Falsely free points

The main application of Minkowski sums is for computing free (disjoint) part placements: if t is in the complement of $A \oplus B$, called the free space, A and $-B + t$ are free. A natural error bound on the approximate free space is that a μ -perturbation

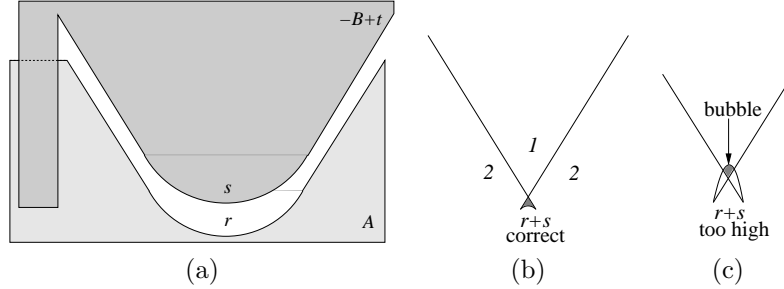


Fig. 12. Bubble: (a) part overlap; (b) non-free exact cell; (c) falsely free approximate cell.

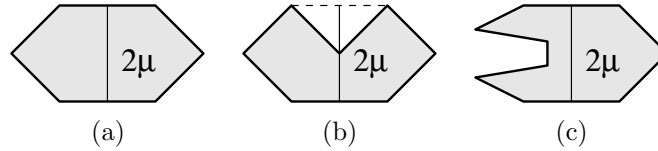


Fig. 13. Regions with (a) and without (b-c) 2μ separation.

of A and $-B + t$ makes them free. Here μ is the Minkowski sum accuracy defined in Sec. 4.3. The error bounds of Thm. 2 permit the approximate Minkowski sum to omit regions of diameter μ that are an unbounded distance from its boundary. Points in these regions, called falsely free points, violate the natural error bound. A cell comprised of falsely free points is called a bubble; the falsely free portion of a non-bubble is called a crack.

Figure 12 shows a bubble. Parts A and $-B + t$ have a large overlap, so a μ -perturbation cannot separate them. In the exact Minkowski sum, t is in a small cell of winding number 3 surrounded by cells of winding number 1 and 2. In the approximate sum, the circle segment $r + s$ is above the line segment intersection, so the t cell is assigned winding number 0 and is falsely free. Figure 19 shows a crack.

Falsely free points can be eliminated by removing the free space regions that are smaller than a threshold, as described in Sec. 8. There is no practical import because applications work at a much coarser resolution. Nevertheless, the approach is inelegant, empirical, and inaccurate. We prevent falsely free points in the approximate monotonic convolution without increasing the asymptotic running time. The method applies to regions whose non-incident boundary chains are 2μ separated (Fig. 13). We see no efficient way to prevent falsely free points in unseparated regions. But neither do these regions appear useful in applications.

A second application of Minkowski sums is for computing free placements of multiple parts, as described in Sec. 8. Let $U_{ij} = P_i \hat{\ominus} P_j$ denote the free space of part P_j with respect to part P_i . The core operation is forming the Minkowski sum $U_{ij} \hat{\oplus} U_{jk}$. Falsely free points here lead to falsely blocked regions of diameter μ that are omitted from the computed set of free placements, which is a negligible error.

Hence, falsely free points require no special treatment. Large errors would occur for points in the approximate sum that were far outside the exact sum, but these would violate the error bounds.

7.1. Algorithm

The sweep algorithm that forms the monotonic convolution arrangement imposes a partial y order on its edges. The input structure imposes a partial y order on the convolution segments at each x value. There are no falsely free points when the two orders agree: a is below b at x in the convolution order implies that the a edge at x is below the b edge at x in the sweep order. We modify the sweep algorithm to place the edges in convolution order.

The boundary chains of a region are partially ordered in y . The chain orders of two regions induce a partial order on their monotonic convolution that is the transitive closure of the following rules. For boundary chains or convolution segments, $V \leq W$ means that V is below W at every x in the intersection of their domains.

- (1) If V and W are chains and $V \leq W$, then $V + p \leq W + p$.
- (2) If L_1 and L_2 are lower chains and p is above L_2 , then $L_1 + L_2 \leq L_1 + p$.
- (3) If U_1 and U_2 are upper chains and p is below U_2 , then $U_1 + p \leq U_1 + U_2$.
- (4) If $L_1 \leq U_1$ and $L_2 \leq U_2$, then $L_1 + L_2 \leq U_1 + U_2$.

Rule 1 follows from the invariance of y order under translation. For rule 2, $L_1 + L_2$ is the lower envelope of $\Pi(L_1) \otimes_k \Pi(L_2)$, which is a superset of $L_1 + p$. Rule 3 follows likewise. For rule 4, pick p above L_2 and below U_2 , so $L_1 + L_2 \leq L_1 + p_2$ by rule 2, $L_1 + p_2 \leq U_1 + p_2$ by rule 1, and $U_1 + p_2 \leq U_1 + U_2$ by rule 3.

The sweep enforces rule 1 when V and W are not incident. The exact $V + p$ and $W + p$ are 2μ separated because they are p translations of V and W , which are 2μ separated by the separation assumption. The approximate segments are μ separated because they are μ close to the exact segments. The $V + p$ edges are placed below the $W + p$ edges because the arrangement algorithm is μ accurate.

The sweep enforces rule 4 when L_1 and U_1 are not incident. Pick p above L_2 and below U_2 . The exact $L_1 + p$ and $U_1 + p$ are 2μ separated because they are p translations of L_1 and U_1 . The exact $L_1 + L_2$ and $U_1 + U_2$ are 2μ separated because $L_1 + L_2 \leq L_1 + p$ by rule 2 and $U_1 + p \leq U_1 + U_2$ by rule 3. The approximate segments are μ separated because they are μ close to the exact segments.

Every other instance of the four rules involves segments from chains L_i and U_i that meet at p_i for $i = 1, 2$. There are three cases. When p_1 and p_2 are convex ($L_1 \leq U_1$ and $L_2 \leq U_2$), the segments are $\{L_1 + L_2, U_1 + U_2\}$ and their order is $L_1 + L_2 \leq U_1 + U_2$ by rule 4. When p_1 is convex and p_2 is concave, the segments are $\{L_1 + L_2, L_1 + p_2, U_1 + p_2, U_1 + U_2\}$ and their order is $L_1 + L_2 \leq L_1 + p_2$ by rule 2, $L_1 + p_2 \leq U_1 + p_2$ by rule 1, and $U_1 + p_2 \leq U_1 + U_2$ by rule 3. When p_1 and p_2 are concave, there are six segments whose order is $p_1 + U_2 \leq p_1 + L_2$ by rule 1, $L_1 + L_2 \leq p_1 + L_2$ by rule 2, $p_1 + U_2 \leq U_1 + U_2$ by rule 3, and three symmetric rules

with indices 1 and 2 switched. When two segments cancel, an inequality becomes an equality that generates a new inequality. At every x where $p_1 + U_2$ cancels $U_1 + U_2$, $U_1 + p_2 \leq p_1 + L_2$, since $U_1 + p_2 \leq U_1 + U_2$ by rule 3, $U_1 + U_2$ equals $p_1 + U_2$ at x , and $p_1 + U_2 \leq p_1 + L_2$ by rule 2. The other three cancellations are analogous.

We make two changes to the sweep algorithm that enforce the convolution order of edges from segments with shared endpoints. A swap event is canceled when the two segments are in convolution order at the swap x . A segment, e , is inserted in the sweep list, S , in convolution order. The chains in S that are related to e by the four rules are in convolution order by inductive hypothesis. Let a and b be the predecessor and successor of e among these chains in convolution order. Use the lowest or highest element of S when a or b is undefined. Segment e is inserted in the subtree of S bounded by a and b . The extra cost of insertion is constant because e is related to at most five segments.

7.2. Correctness

Let t lie in a free cell of the approximate arrangement of $A \otimes_m B$. We prove the existence of regions A' and B' , μ close to A and B , for which A' and $-B' + t$ are free (Thm. 4).

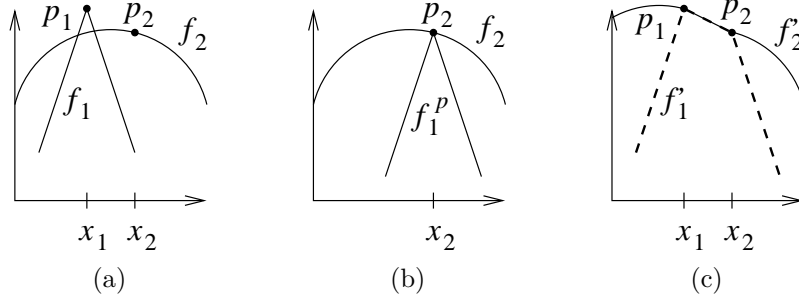
The first step is to define a partial order on the boundary chains of A and $-B + t$. The A chains and the B chains are partially ordered. The $-B + t$ chains have the reverse order of the B chains. We say that a chain is below t when its edge at t_x is below t in the sweep order. The crusts below t couple the A and $-B + t$ orders. When t is above upper crust $U_1 + U_2$, upper chain U_1 of A precedes lower chain $-U_2 + t$ of $-B + t$. When t is above lower crust $L_1 + l_2$, upper chain $-L_2 + t$ precedes lower chain L_1 . The chain relation is acyclic, hence defines a partial order.

Lemma 2. *The chain relation is acyclic.*

Proof. Suppose an increasing cycle exists. The cycle contains chains from both A and $-B + t$ because the individual orders are acyclic. Starting from an A chain, U_1 , follow the cycle until the first $-B + t$ chain, $-U_2 + t$, which must be an upper chain. Then follow the chain to the next A chain, L_1 , which must be a lower chain. If $U_1 \leq L_1$, shorten the cycle by erasing the chains $-U_2 + t$ through $-L_2 + t$. Repeat this process until $L_1 \leq U_1$. Since $-U_2 + t$ comes before $-L_2 + t$ in the sequence of $-B + t$ chains, $-U_2 + t \leq -L_2 + t$ and $L_2 \leq U_2$. By rule 4, $L_1 + L_2 \leq U_1 + U_2$. But $U_1 \leq -U_2 + t$ and $-L_2 + t \leq L_1$ imply that t is above $U_1 + U_2$ and is below $L_1 + L_2$, which contradicts $L_1 + L_2 \leq U_1 + U_2$. \square

The second step is a technical lemma. Let functions $f_i(x)$ be defined on $[l_i, r_i]$ for $i = 1, 2$. The translation of f_i by p is the function $f_i^p(x) = f_i(x - p_x) + p_y$ on $[l_i + p_x, r_i + p_x]$. Let there exist points $p_i = (x_i, f_i(x_i))$ on f_i such that $p = p_2 - p_1$ satisfies $f_1^p \leq f_2$ (Fig. 14).

Lemma 3. *There exist functions f'_i on $[l_i, r_i]$, $\|p\|$ close to f_i , such that $f'_1 \leq f'_2$.*


 Fig. 14. Technical lemma: (a) f_i ; (b) f_1^p ; (c) f_i' .

Proof. Suppose $x_1 \leq x_2$; the other case is similar. Define

$$\begin{aligned} f_1'(x) &= f_1(x) \quad \text{and} \quad f_2'(x) = f_2^{-p}(x) \quad \text{for } x < x_1 \\ f_1'(x) &= l(x) \quad \text{and} \quad f_2'(x) = l(x) \quad \text{for } x_1 \leq x \leq x_2 \\ f_1'(x) &= f_1^p(x) \quad \text{and} \quad f_2'(x) = f_2(x) \quad \text{for } x > x_2 \end{aligned}$$

with $l(x)$ the line from p_1 to p_2 . If $x_1 = x_2$, the middle line segment is vertical. The inequality holds trivially in the second case and holds by assumption in the third case. It holds in the first case because this is a $-p$ translation of the third case. The distance from a curve to its p translation is $\|p\|$, as is the length of l . \square

The third step employs our prior results.² Let O be the number of pairs of an upper chain of A and a lower chain of B that overlap in x . An upper and lower chain that overlap in x with the upper below the lower are called facing. Let F be the number of facing pairs of A/B and B/A chains. Let T be the number of concave turning points of one region interior to the other region. The monotonic intersection number of A and B is defined as $M = O - F - T$. It is zero when A and B are free and is positive otherwise. The winding number of the $A \otimes_m B$ cell that contains the point t equals the monotone intersection number of A and $-B + t$; the winding number with respect to the crusts equals $O - F$.

Lemma 4. *A t where $O - F = 0$ is not falsely free.*

Proof. If t is above the approximate upper crust $U_1 + U_2$ and below the exact crust, it is μ close to the exact crust, since the arrangement is μ accurate. The translation that maps t to the nearest point on the exact crust translates $-U_2 + t$ to face U_1 and be in contact. Let p_1 and p_2 be the points of contact on U_1 and on $-U_2 + t$ before translation. By Lemma 3, there exist μ perturbations such that $U_1' \leq -U_2' + t$, which implies that t is above $U_1' + U_2'$. Likewise for lower crusts.

We have defined μ perturbations that realize the inter-region inequalities in the chain order. The intra-region inequalities are correct. Since the chain order is the transitive closure of these inequalities, any $U \leq V$ follows from an increasing path from U to V that contains at most one inter-region inequality. The perturbation

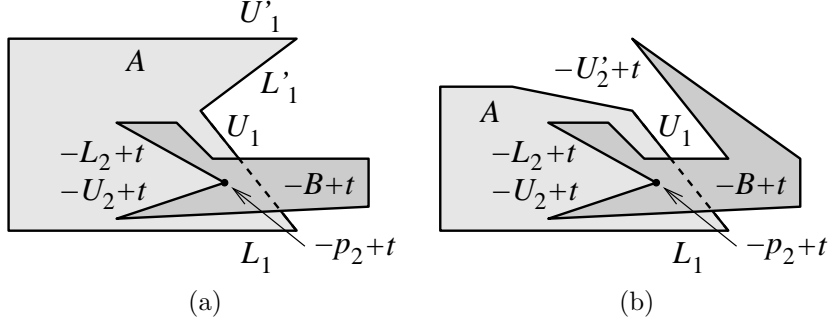


Fig. 15. Extended chain order: (a) $U_1 \leq L'_1$; (b) $U_1 \leq -U'_2 + t$.

that realizes this inequality realizes the intra-region inequalities by 2μ separation, hence realizes $U \leq V$. We conclude that there exists a μ perturbation, A' and B' , that realizes the chain order by Lemmas 9 and 10 in our prior paper.⁵

Using the downward vertical ray from t , $O - F$ equals the number of lower crusts below t minus the number of upper crusts below t . This number is zero in $A' \otimes_m B'$ because it is zero in the approximate arrangement and A' and B' realize the chain order. The winding number of t in $A' \otimes_m B'$, $O - F - T = -T$, is zero because winding numbers and T values are non-negative. \square

The final step is to prove the general result. We extend the chain order to the concave vertices of each region that are inside the other. Let $-p_2 + t$ be a concave vertex of $-B + t$ with lower chain $-U_2 + t$ and upper chain $-L_2 + t$. If $-p_2 + t$ is inside A , it is above a lower chain, L_1 , of A and is below an upper chain, U_1 , of A (Fig. 15). In the approximate arrangement, t is above the T segment $L_1 + p_2$ and is below $U_1 + p_2$. We add $L_1 \leq -L_2 + t$ and $-U_2 + t \leq U_1$ to the chain order at $x = -p_{2x} + t_x$. Likewise for a concave vertex of A that is inside $-B + t$.

Lemma 5. *The extended chain relation is acyclic.*

Proof. A cycle must include one of the new orders, since the base relation is acyclic. Consider $-U_2 + t \leq U_1$; the other cases are similar. The new order occurs when t is below $U_1 + p_2$. Since $U_1 + p_2 \leq U_1 + U_2$ by rule 3, t is below $U_1 + U_2$, so $U_1 \leq -U_2 + t$ is not in the chain order. Thus, the chain after U_1 in the cycle is not $-U_2 + t$.

If a lower chain, L'_1 , of A is next (Fig. 15a), the cycle reaches an upper chain, U'_1 , of A before returning to $-U_2 + t$. Since upper chains cannot be incident, U'_1 is 2μ distant from U_1 . Neither U'_1 nor its successors in the cycle can be below $-U_2 + t$ by μ accuracy. An upper chain of A cannot follow U_1 because some lower chain of A intervenes. The last case is a chain, $-U'_2 + t$, of $-B + t$ (Fig. 15b), which is 2μ distant from $-U_2 + t$ and cannot be followed by $-U_2 + t$ as before. Hence a cycle is impossible. \square

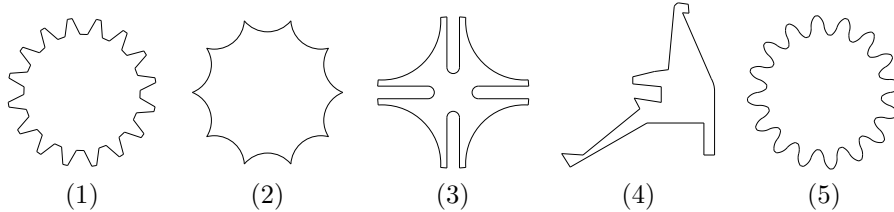
20 *Victor Milenkovic and Elisha Sacks*

Fig. 16. Test part shapes.

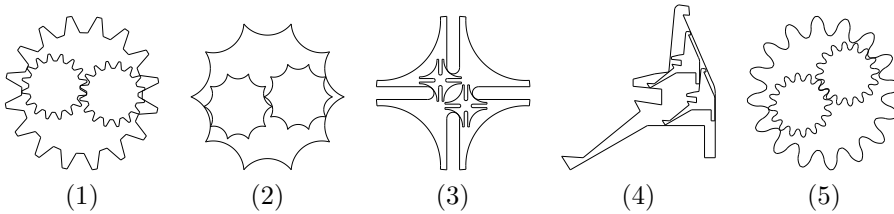


Fig. 17. Packing two copies into the minimum scale copy of a third.

Theorem 4. *The monotonic convolution has no falsely free points.*

Proof. Let the winding number of t be zero in the approximate arrangement. The extended chain order at x is acyclic by Lemma 5. Hence, it can be realized by the method of our prior paper. (The realization error is unbounded, but that is irrelevant.) Define regions A^+ and B^+ that realize the extended order. The winding number of t in $A^+ \otimes_m B^+$ is zero because the exact and approximate $O - F$ are equal and the exact T is no smaller than the approximate one, since the T segments are realized. Hence A^+ and $-B^+ + t$ are free, so neither can have a vertex inside the other, so the exact $T = 0$, so $O - F = 0$ and Lemma 2 applies. \square

8. Validation

We validated the Minkowski sum algorithms by using them to implement Avnaim and Boissonnat's algorithms⁶ for translating two parts into an arbitrary container and three parts into a rectangular container. We tested five industrial part shapes (Fig. 16). The number of boundary segments without turn segments is $s = 32, 10, 24, 22, 96$. First, we packed two instances of each part into a third instance (Fig. 17). We determined the minimum scale of the third part by binary search on the scale down to the accuracy of double precision arithmetic. Second, we determined the smallest square that contains the three parts by binary search on the size of the square (Fig. 18).

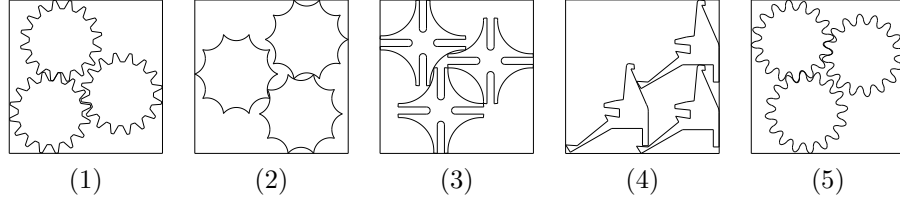


Fig. 18. Packing three copies into the minimum square.

8.1. Packing algorithms

The algorithm for two-part packing of P_1, P_2 into container C is as follows. Let $P_0 = \overline{C}$ be the complement of the container. For $0 \leq i, j \leq 2$, calculate $U_{ij} = \overline{P_i \oplus -P_j}$. Calculate $U'_{01} = U_{01} \cap (U_{02} \oplus -U_{12})$. If U'_{01} is empty, there is no solution. Otherwise, pick $t_1 \in U'_{01}$. Calculate $U'_{02} = U_{02} \cap (U_{12} + t_1)$. Pick $t_2 \in U'_{02}$. $P_1 + t_1$ and $P_2 + t_2$ lie inside C without overlap.

The algorithm for three-part packing of P_1, P_2, P_3 into a rectangle $R = \overline{P_0}$ is as follows. For $0 \leq i, j \leq 3$, calculate $U_{ij} = \overline{P_i \oplus -P_j}$. For $1 \leq i, j \leq 3$, calculate $U'_{ij} = U_{ij} \cap (-U_{0i} \oplus U_{0j})$. Calculate $U''_{13} = U'_{13} \cap (U'_{12} \oplus U'_{23})$. If U''_{13} is empty, there is no solution. Otherwise, pick $t_{13} \in U''_{13}$. Calculate $U'_{01} = U_{01} \cap (U_{03} - t_{13})$ and $U''_{12} = U'_{12} \cap (-U'_{23} + t_{13}) \cap (-U'_{01} \oplus U_{02})$. Pick $t_{12} \in U''_{12}$. Calculate $U''_{01} = U'_{01} \cap (U_{02} - t_{12})$. Pick $t_1 \in U''_{01}$. Set $t_2 = t_1 + t_{12}$ and $t_3 = t_1 + t_{13}$.

8.2. Implementation

We pick t in region U using the trapezoidal decomposition generated by the arrangement algorithm. The width of a trapezoid is its x extent. Define its height as the y extent at the midpoint of the x extent. Define its size as the minimum of its width and height. We choose the maximum size trapezoid and set t to the midpoint of the vertical at the x midpoint.

In the kinetic algorithm, cracks and bubbles are eliminated heuristically. Each vertex that is within a threshold of the edge immediately above or below is connected to it by a vertical line segment. These segments convert cracks into bubbles (Fig. 19). After insertion, a free cell is rejected when its maximum trapezoid size is less than the threshold. We set the threshold to 10^{-13} units based on a preliminary run of the algorithm. We selected a t in each region, as described above, and computed the maximum trapezoid size for which A overlaps $-B + t$.

Region U'_{02} in the first algorithm may be empty even though U'_{01} is not. Similarly, U''_{12} or U''_{01} may be empty in the second algorithm. These situations are artifacts of approximate computation. When they occur, we abort the binary search and return the smallest container size seen so far.

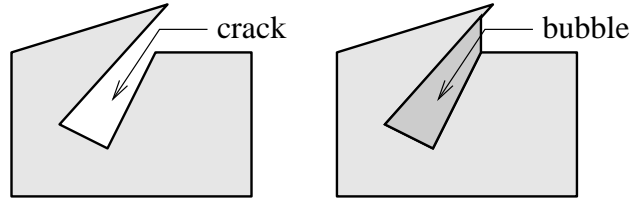


Fig. 19. Crack is converted to bubble by vertical segments.

8.3. Results

Table 1 shows the time and error for the the kinetic and monotonic algorithms. There are nine test cases. The first three are variants of part 1 (Fig. 16) with 8, 16, and 32 teeth. The next three are variants of part 2 with 10, 20, and 40 arcs. The final three are parts 3–5.

The total time, T , is 2%–70% less for the monotonic algorithm with a 42% average reduction. The convolution computation time, T_c , is greater for the monotonic convolution than for the kinetic convolution, but the arrangement time, T_a , is much smaller. The time, T_s , for completing the Minkowski sum is larger for the kinetic algorithm because of the crack and bubble heuristics. The time T_o , which should be identical, is comparable. The error tests were conducted in a separate run of the validation, so they do not affect the running times.

The robustness time, T_f , is the cost of our algorithms over direct floating point implementations of the exact algorithms. The main costs are Dekker’s method for sum segment endpoint computation and falsely free point elimination. The average/maximum cost is 2%/11% of the total running time. Although the monotonic cost is smaller than the kinetic cost, it is a larger percentage of the running time, since the algorithm is much faster.

Theorem 4 implies that a μ -erosion makes A and $-B + t$ free at every t in the free space computed by the monotonic algorithm. We estimate μ as the maximum size overlap region as t ranges over all vertices and midpoints of edges of $A \oplus B$. The size of an overlap region is defined as the size of its largest trapezoid. For the kinetic algorithm, a second error metric is the maximum size region that is smaller than the threshold, yet contains a truly free t . The estimated μ is the maximum of the two errors. In two instances, two-part packing for parts 3 and 4, this second error metric makes the kinetic algorithm significantly less accurate than the monotonic algorithm. In all cases, the accuracy, $\alpha = -\log_2 \mu$, is 43–50 bits. This accuracy far exceeds manufacturing accuracy, which is at most 24 bits. Each algorithm aborted in five cases due to an incorrectly empty U' or U'' set, but always after at least 50 iterations of the binary search. Hence, the packing algorithm was at least as accurate as the underlying Minkowski sum algorithm.

The quantity n_c is the total number of cells in all the arrangements of all the convolutions involved in the packing algorithm. This number is averaged over the

Table 1. Packing results: k_2 and m_2 are for two-part packing with the kinetic and monotonic algorithms; k_3 and m_3 are for three-part packing; T average solution time in milliseconds; breakdown into time T_c for convolution, T_a for arrangement of convolution, T_s for rest of Minkowski sum, T_o for other than Minkowski sum, and T_f for robustness; accuracy α in bits; average n_c cells in thousands and n_b bubbles per problem.

	part 1a				part 1b				part 1c			
	k_2	m_2	k_3	m_3	k_2	m_2	k_3	m_3	k_2	m_2	k_3	m_3
T	67	40	131	129	662	341	616	479	9185	5037	5686	3726
T_c	3	14	7	23	26	87	26	95	290	867	162	468
T_a	55	18	51	37	609	229	425	210	8634	4008	4681	2431
T_s	3	2	21	18	9	9	47	52	210	113	215	202
T_o	6	5	53	52	17	16	118	121	50	49	628	625
T_f	1	2	0	2	10	6	6	9	164	74	116	53
α	48	48	48	48	47	47	47	47	48	48	46	47
n_c	1.5	0.2	1.3	0.4	15.7	3	17.2	3.1	177	55.1	193.3	45.4
n_b	21	0	4	0	12	0	42	0	46	0	110	0
	part 2a				part 2b				part 2c			
	k_2	m_2	k_3	m_3	k_2	m_2	k_3	m_3	k_2	m_2	k_3	m_3
T	67	39	98	78	140	53	163	112	315	93	219	102
T_c	3	10	5	12	7	19	7	21	13	40	14	33
T_a	56	21	59	28	115	20	84	24	269	23	141	19
T_s	2	2	10	10	6	4	21	20	9	10	18	15
T_o	6	6	24	27	13	10	50	47	24	20	45	36
T_f	1	0	1	1	2	1	2	2	4	2	2	1
α	44	44	46	46	43	43	45	45	44	44	48	48
n_c	1.7	0.3	2.1	0.4	3.5	0.3	3.1	0.3	6.8	0.3	4.6	0.1
n_b	173	0	286	0	525	0	556	0	956	0	776	0
	part 3				part 4				part 5			
	k_2	m_2	k_3	m_3	k_2	m_2	k_3	m_3	k_2	m_2	k_3	m_3
T	82	54	126	93	32	25	42	35	2024	872	2059	1299
T_c	4	11	9	23	2	6	1	5	84	255	94	273
T_a	66	32	89	39	26	16	25	15	1873	569	1554	610
T_s	5	5	8	9	1	2	5	5	35	18	116	124
T_o	8	6	20	21	3	1	11	10	32	28	295	293
T_f	1	1	1	1	0	1	0	1	32	18	31	21
α	43	50	48	48	43	51	50	50	46	45	46	45
n_c	2.3	0.6	4.2	0.6	0.7	0.3	0.6	0.2	33.4	3.9	49.4	4.9
n_b	42	0	86	0	11	0	6	0	744	0	1215	0

iterations of the binary search. This quantity is larger for the kinetic algorithm, thus accounting for the larger time T_a to construct the arrangement.

We limited the validation to shapes with 2μ separation. Manufacturing processes

cannot generate unseparated shapes because process accuracy is much lower than μ . One could model unseparated shapes via set operations on separated shapes. The Minkowski sum algorithm would be applied to the separated shapes and the final sum would be obtained via set operations. Alternately, one could sum an unseparated shape with a small disk to obtain a separated shape.

9. Conclusion

Our prior work² shows that the monotonic convolution is less complex than the kinetic convolution in theory and in practice. This paper demonstrates that the reduced complexity translates into lower running time and higher accuracy. We present approximate convolution algorithms that permit iterated application of set operations and of Minkowski sums with high accuracy. The monotonic algorithm avoids falsely free points by enforcing consistency rules, whereas the kinetic algorithm uses a threshold based on an *a priori* accuracy estimate. The approximate Minkowski sum algorithms easily handle Avnaim and Boissonnat's algorithms for two and three part containment applied to profiles with circle segments. An exact algorithm appears impractical because the output algebraic degree is 16 for two parts and is 256 for three parts, and the bit complexity grows analogously.

The next step is to handle planar regions that rotate and translate. The kinetic algorithm generalizes to this case, but not the monotonic algorithm. The generalization, called a configuration space partition, is useful for robot path planning, part layout, mechanical design, and more. We are working on an algorithm that constructs the approximate configuration space and that employs it for these tasks.

Acknowledgments

Research supported by NSF grants IIS-0082339, CCF-0306214, and CCF-0304955.

References

1. L. Guibas, L. Ramshaw, and J. Stolfi. A Kinetic Framework for Computational Geometry. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 100–111, 1983.
2. Victor Milenkovic and Elisha Sacks. A monotonic convolution for Minkowski sums. *International Journal of Computational Geometry and Applications*, 17(4):383–396, 2007.
3. Ron Wein. Exact and efficient construction of planar Minkowski sums using the convolution method. In *Proceedings of the 14th Annual European Symposium on Algorithms*, 2006.
4. Arno Eigenwillig and Michael Kerber. Exact and efficient 2d-arrangements of arbitrary algebraic curves. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA08)*, pages 122–131, 2008.
5. Victor Milenkovic and Elisha Sacks. An approximate arrangement algorithm for semi-algebraic curves. *International Journal of Computational Geometry and Applications*, 17(2), 2007.

6. Francis Avnaim and Jean-Daniel Boissonnat. Simultaneous containment of several polygons. In *Symposium on Computational Geometry*, pages 242–247, 1987.
7. Victor Milenkovic and Karen Daniels. Translational polygon containment and minimal enclosure using mathematical programming. *International Transactions in Operational Research*, 6:525–554, 1999.
8. Elisha Sacks. Path planning for planar articulated robots using configuration spaces and compliant motion. *IEEE Transactions on Robotics and Automation*, 19(3), 2003.
9. Leo Joskowicz and Elisha Sacks. Computer-aided mechanical design using configuration spaces. *Computing in Science and Engineering*, 1(6):14–21, 1999.
10. Dan Halperin and Eli Packer. Iterated snap rounding. *Computational Geometry: Theory and Applications*, 23(2):209–222, 2002.
11. John Hershberger. Improved output-sensitive snap rounding. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 357–366, New York, NY, USA, 2006. ACM Press.
12. Victor Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2d constructions. *Computational Geometry: Theory and Applications*, 13:3–19, 1999.
13. Victor Milenkovic. Shortest path geometric rounding. *Algorithmica*, 27(1):57–86, 2000.
14. Eli Packer. Iterated snap rounding with bounded drift. In *Proceedings of the Symposium on Computational Geometry*, pages 367–376, New York, NY, USA, 2006. ACM Press.
15. Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Symposium on Computational Geometry*, pages 284–293, 1997.
16. S. Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. *Discrete and Computational Geometry*, 22:593–618, 1999.
17. S. Fortune. Polyhedral modelling with multiprecision integer arithmetic. *Computer-Aided Design*, 29(2):123–133, 1997.
18. T. J. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.
19. Kaul A., M. A. O’Connor, and V. Srinivasan. Computing Minkowski sums of regular polygons. In *Proceedings of the Third Canadian Conference on Computational Geometry*, pages 74–77, 1991.