14th ACM Symposium on Computational Geometry, June 1998

Rotational Polygon Containment and Minimum Enclosure

Victor J. Milenkovic
University of Miami
Department of Mathematics and Computer Science*

Abstract

An algorithm and implementation is given for *rotational polygon containment*: given polygons $P_1, P_2, P_3, \ldots, P_k$ and a container polygon C, find rotations and translations for the k polygons that place them into the container without overlapping. A version of the algorithm and implementation also solves *rotational minimum enclosure*: given a class $\mathcal{C}$ of container polygons, find a container $C \in \mathcal{C}$ of minimum area for which containment has a solution. Minimum enclosure algorithms are given for the following classes: 1) rectangles of fixed width, 2) scaled copies of a fixed convex polygon, 3) arbitrary rectangles. Containment and minimum enclosure are NP-hard (even in the purely translational case). The minimum enclosure is approximate: it bounds the the minimum area between $(1-\epsilon)A$ and A. Experiments are done to determine the largest practical value of k for both containment and minimum enclosure. Important applications for these algorithm to industrial problems are discussed. The paper also gives practical algorithms and numerical techniques for robustly calculating polygon set intersection, Minkowski sum, and *range intersection*: the intersection of a polygon with itself as it rotates through a range of angles.

1 Introduction

A number of industries generate new parts by cutting them from stock material: cloth, leather (hides), sheet metal, glass, *etc.* These industries need to generate dense non-overlapping layouts of polygonal shapes. Because fabric has a grain, apparel layouts usually permit only a finite set of orientations. Stripes, plaids, or other patterns on the fabric can further limit the allowed orientations and translations. Nevertheless, apparel manufacturers often allow small rotations (called "tilting"), typically not more than 3 degrees, but still large enough to make a significant difference in cloth utilization. Glass and sheet metal (and sometimes leather) have no grain (or stripes or plaids), and therefore layout on these materials allows arbitrary orientations for the parts.

The problem these manufacturers need to solve is either *containment* or *minimum enclosure*.

Containment Given polygons P_1, P_2, ..., P_k and a fixed container C, place the polygons into C without overlapping. As in the case of overlap minimization, P_0 denotes $\overline{C}$, in which case the goal becomes to find a non-overlapping layout of P_0, P_1, ..., P_k.

Minimum Enclosure Find a non-overlapping layout of P_1, P_2, ..., P_k that minimizes some measure of the container. The most common version in the textile industry is *strip packing*: minimize the length of a rectangular container R of fixed width. It is also useful to minimize the area of R.

The values of k one sees is often in the dozens or even hundreds. Unfortunately, even the translational versions of containment and minimum enclosure are NP-hard, and therefore one has to expect the running time of a containment or minimum enclosure algorithm to be exponential in k. In the apparel industry, no layout software has yet replaced a human. All hope is not lost: one has to use a heuristic or meta-heuristic of which there are many for the layout problem.

In the translational case, we have demonstrated [8] that a containment *algorithm* for modest k ($k = 5$ or even perhaps $k = 2$) is an excellent *tool* for the creation of containment *heuristics* for larger values of k, even exceeding expert human performance in some case. This work is based on packing one column at a time, but it is only one among many ways a "large k" heuristic can use a "small k" algorithm. For instance, subsets of the polygons might be packed together tightly and then the union treated as a single polygon.

This paper examines the following question: for what values of k is it possible to practically solve rotational containment and minimum enclosure problems *algorithmically*? In other words, for what value of k is it necessary to switch to heuristics and give up on performance guarantees? The two main issues are running time and numerical robustness.

1.1 New Results.

This paper presents new algorithms and robust implementations for rotational polygon containment and minimum enclosure and implementations. As far as we know, there are

*http://www.cs.miami.edu/~vjm, vjm@cs.miami.edu. This research was funded by the Textile/Clothing Technology Corporation from funds awarded by the Alfred P. Sloan Foundation, by NSF grants CCR-91-157993 and NSF-CCR-97-12401, and by a subcontract of a National Textile Center grant to Auburn University, Department of Consumer Affairs.

no other *algorithms* for $k > 2$ or implementations for $k \geq 2$ for multiple non-convex polygons. Minimum enclosure algorithms are given for the following classes: 1) rectangles of fixed width, 2) scaled copies of a fixed convex polygon, 3) arbitrary rectangles. The minimum enclosure is approximate: it bounds the the minimum area between $(1-\epsilon)A$ and A. Experiments are done to determine the largest practical value of k for both containment and minimum enclosure.

The property that distinguishes an *algorithm* from a *heuristic* is the ability to say "no". The key to detecting infeasibility in our algorithms is the *range intersection* $P(\alpha, \beta)$, of a polygon P: the intersection of P with itself as it rotates through the ranges of angles from α to β about the origin. The new containment/enclosure algorithms integrate the range intersection into the framework we developed originally for solving translational containment/minimum enclosure [28, 33].

This paper also shows that the range intersection has linear complexity, although it may have circular arcs in its boundary. It gives practical algorithms for computing the range intersection and for approximating it by a polygon in a manner that is suitable for the containment/enclosure algorithms.

Like our earlier translational algorithms, our new containment/enclosure algorithms are very numerically demanding. They apply polygon set intersection and Minkowski sum over and over to the same collection of polygons. *Cascading*, the repeated use of the output as the next input, cannot be implemented using "pure" exact arithmetic because the number of bits required to represent coordinates can grow exponentially [31]. Cascading also quickly crashes any naive rounded arithmetic implementation. This paper presents a new hybrid technique *nearest pair rounding* for implementing polygon set operations in floating point arithmetic. Currently, nearest pair rounding must use some exact arithmetic to ensure correctness, but we have created an version that uses no exact arithmetic at all and which, so far, has been numerically robust, even under the cascading "torture test".

1.2 Related Work.

In the literature, there are essentially three approaches to layout of non-convex polygons: 1) heuristics and meta-heuristics (neural nets, simulated annealing, genetic algorithms), 2) "classical" computational/combinatorial geometry, 3) a combination of computational geometry and mathematical programming (which we use here).

Heuristics and meta-heuristics are limited in theory because they cannot say "no" if there is no solution. According to our sources in the apparel industry, available layout software is limited in practice to falling about 5% behind humans in cloth utilization. There are a number of surveys of packing/nesting heuristics [11, 7, 38, 10, 36, 12]. Recent work has used simulated annealing [21], boundary matching [22], grouping of polygons into sub-rectangles [13], genetic algorithms [5], database driven layout [23, 24], or a hybrid approach [18, 19, 37, 34]. Heuristic approaches have been tailored to sheet metal [35] and leather [20], both of which permit rotations.

Chazelle [6] introduced the single-polygon containment problem: place m-gon P into n-gon container Q. For convex Q, the running time bound is $O(mn^2)$, and for non-convex P and Q, $O(m^3n^3(m+n)\log(m+n))$. Avnaim *et al.* [4, 3] improve this to $O(m^3n^3\log(m+n))$. Most recent work deals with finding the largest copy of a convex P that can be placed, which is equivalent to finding the minimum (scaled) enclosure. For convex Q the best running time is $O(mn^2\log n)$ [1], and for non-convex Q, $O(m^2n^2)$ [2]. Grinde and Cavalier use an extensive case analysis and linear programming to place a single convex P [15] or convex P_1 and P_2 [16]. The running time of the first algorithm appears to be $O(m^2n^3)$, and it is not clear what the running time of the second is, but for one of its cases they are able to use parametric programming and find a solution by solving $O(m^4n^4)$ linear programs.

For multi-polygon *translational* layout, we have had considerable theoretical and practical success using a combination of computational geometry and mathematical programming (CG/MP) [9, 29, 28]. These algorithms can practically solve containment for up to ten polygons and minimum enclosure for at least five. We have also proved a number of theoretical running time bounds including $O((m^2+mn)^{2k}\log n)$ for placing k non-convex m-gons into a non-convex n-gon and $O(m^{4k-4}\log m))$ for placing them into a minimum area, fixed orientation rectangle [27] (this paper also surveys other results in multi-polygon translational layout).

1.3 Relation of New Algorithm to Previous Work

Avnaim and Boissonnat gave a formula for placing two polygons in a container. This formula works for both the translational and the rotational case. Our previous translational algorithm generalizes this work to $k > 2$ polygons through the use of a branch and bound paradigm. It is possible to generalize our translational algorithm to handle the rotational case in the same manner. One might call this the *true generalization* of our translational algorithm.

The algorithm presented here is *not* the true generalization. The true generalization would require robust set operations on subsets of $\mathbf{R}^2 \times S^1$. These subsets would either represent either a) valid configurations (translation plus rotation) of a single polygon to place it in the container or b) the set of valid (non-overlapping) relative positions of one polygon with respect to another. Unfortunately, the configuration sets are three dimensions with curved surface boundary. As yet, there are no numerically robust ways to implement the necessary set operations. As we have previously stated, the branch and bound paradigm cascades these set operations in a manner that is extremely stressful numerically.

The algorithm presented here only requires set operations on polygons in two dimensions, which we know how to do robustly. It is likely that the true generalization would run faster than the current algorithm–if and when it could be implemented robustly. The algorithm presented here represents a practical tradeoff of running time for numerical robustness.

1.4 Outline.

Section 2 gives the algorithms for rotational containment and minimum enclosure using the range intersection. Section 3 analyzes the range intersection and gives algorithms for constructing it and useful polygonal approximations to it. Section 4 gives the nearest pair rounding algorithm. Finally, Section 5 gives results.

2 Containment/Enclosure Algorithms

The rotational containment/enclosure algorithm uses a "branch and bound" approach which generalizes our previous practical translational containment algorithm [28, 33]. However, this is also essentially the approach used by many mathematical programming algorithms.

The input to the containment algorithm is a set of polygons $P_1, P_2, P_3, \ldots, P_k$ and a container polygon C. In the case of minimum enclosure, the container is replaced by a set $\mathcal{C}$ of containers. A *configuration* is an assignment of translations $t_1, t_2, \ldots, t_k$ and rotations $\theta_1, \theta_2, \ldots, \theta_k$ to the polygons. To simplify the notation, it is convenient to replace the container by an additional polygon $P_0 = \overline{C}$ which is the complement of the container and to fix its translation t_0 and rotation θ_0 at $(0, 0)$ and 0, respectively.

In the case of containment, the goal is to find all configurations (or at least one configuration) with zero overlap among the polygons. Since the complement of the container is P_0, this means that $P_1, P_2, \ldots, P_k$ lie inside the container. In what follows these configurations are called the *solutions*. In the case of minimum enclosure, the goal is to find a non-overlapping layout which fits in the minimum area container $C \in \mathcal{C}$.

2.1 Abstract Algorithm.

We use the same language as our translational work. A *hypothesis* is a set of constraints on the configuration. *Restriction* adds to this set of constraints in a way that is guaranteed not to eliminate any solutions. *Evaluation* attempts to find a solution within the hypothesis (that satisfies its constraints). *Subdivision* splits a hypothesis H into two sub-hypotheses H' and H'' such that all solutions in H reside in (satisfy all the constraints of) either H' or H''.

The containment algorithm first generates a root hypothesis: a set of constraints which all solutions must satisfy. It restricts this hypothesis and evaluates it. If it cannot either a) restrict it to the empty set or b) find a solution, it subdivides the hypothesis and recurses on the two subhypotheses. A hypotheses for which evaluation is successful is called a *solution hypothesis*. If only one solution is desired, the algorithm can stop when it finds the first one. Otherwise, it generates a set of solution hypotheses. Note that we only find one solution for each solution hypothesis, not all. In the translational case, it is possible to find all solutions [27], but we have not done this for the rotational case.

The minimum enclosure algorithm acts the same as the containment algorithm, except that every time it finds a solution hypothesis, it establishes a new upper bound on the area of the minimum container. It uses this upper bound as a constraint to restrict the previously discovered solution hypotheses, and it recurses on these. Its output is a solution hypothesis with minimum area (or possibly several solution hypothesis whose solutions have the same area). To establish that the area is indeed minimal, it is necessary to run the algorithm on its output hypotheses with a slightly smaller upper bound. We first diminish the upper bound by 1%. If no new solution is found, then we roll back and try diminishing the upper bound by 0.01%, and so forth. By these means, the upper bound can be established to any degree of numerical accuracy. In theory, it might be possible to apply symbolic perturbation to the solution to establish that there is no solution for any smaller area. In practice, this is not possible since much of our algorithm is numerical.

2.2 Root Hypothesis.

The hypothesis for the containment algorithm takes the following form,

$$\alpha_i \le \theta_i \le \beta_i, \qquad 0 \le i \le k, \tag{1}$$
$$t_j - t_i \in U_{ij}, \qquad 0 \le i < j \le k, \tag{2}$$

where α_i and β_i are bounds on the angle and U_{ij} is a (usually non-convex) planar region.

In the case of translational containment [28, 33] all solutions must satisfy,

$$t_j - t_i \in \overline{P_i \oplus P_j}, 0 \le i < j \le k, \tag{3}$$

where $\oplus$ is the Minkowski sum,

$$A \oplus B = \{a + b \,|\, a \in A \text{ and } b \in B\}.$$

Hence, we set $U_{ij} = \overline{P_i \oplus P_j}$ in the root hypothesis. However, this does not work for rotational containment.

Define the *range intersection*

$$P(\alpha, \beta) = \bigcap_{\alpha \le \theta \le \beta} P(\theta), \tag{4}$$

where $P(\theta)$ is P rotated by angle θ.[1] The range intersection is the intersection of all copies of P as it is rotated from α to β. The following lemma provides a way of constructing a root hypothesis for rotational containment.

Lemma 2.1 *There is a solution to rotational containment for $\alpha_i \le \theta_i \le \beta_i$, $0 \le i \le k$ only if (but not if, alas) there is a solution to translational containment for $P(\alpha_i, \beta_i)$, $0 \le i \le k$.*

Proof. If $\alpha_i \le \theta_i \le \beta_i$, then $P(\alpha_i, \beta_i) \subseteq P(\theta_i)$. Given the solution to rotational containment, just use the same translations. □

In the case of rotational containment, we set

$$\alpha_i = 0, \beta_i = 2\pi, 0 \le i \le k,$$

in the root hypothesis, and set

$$U_{ij} = \overline{P_i(0, 2\pi) \oplus P_j(0, 2\pi)}, 0 \le i < j \le k.$$

By Equation 2.1 and Equation 3, all solutions to containment will satisfy the constraint Equations 1 and 2 for this hypothesis.

In the case of minimum enclosure, if the set $\mathcal{C}$ of containers is either 1) a set of rectangles with fixed width, 2) a set of similar (scaled) copies of a convex polygon, then we set C equal to the element whose size (length or scale) equals a known upper bound. (For sufficiently large size, it is trivial to find a solution.) We postpone the discussion of more general sets, such as 3) *all rectangles*, until Section 2.6.

2.3 Restriction.

In our previous work on translational containment, we established two types of restriction: *geometric restriction* [7] and *linear programming restriction* [27]. These restrictions were both derived from Equation 2, and therefore they apply in the case of rotational containment as well.

[1] All rotations are about the origin. It is assumed that all polygons contain the origin.

Repeatedly, for all triples h, i, j, geometric restriction replaces U_{ij} by a subset,[2] $U_{ij} \cap (U_{ih} \oplus U_{hj})$. It stops when no polygon diminishes in area by more than a threshold fraction. This repetition (cascading) can lead to numerical difficulties which we deal with using nearest pair rounding (Section 4). Note: for $k = 2$, geometric restriction is an exact algorithm. In this special case, it becomes essentially the same as Avnaim and Boissonnat's exact formula for this case [4, 3]. If either geometric restriction or linear programming restriction generates the empty set, then there is no solution in the current hypothesis.

2.4 Evaluation.

We have previously developed a practical algorithm for *rotational overlap minimization* [32]: find translations and rotations that minimizes the sum of the overlaps among $P_0, P_1, P_2, \ldots, P_k$. This algorithm can also perform *rotational compaction*: given a non-overlapping layout of P_1, P_2, ..., P_k inside a rectangular container C, find a layout which minimizes (local minimum) of the container while keeping the width fixed. In the full paper we show how to generalize compaction to minimize the area of a scaled convex enclosure or even the area of an arbitrary rectangular container. (The latter problem is non-linear.)

Evaluation for the containment algorithm simply consists of running rotational overlap minimization with a set of additional constraints derived from the current hypothesis. Since compaction is based on linear programming, the additional constraints must be convex. The additional constraints are a) the angle constraints, Equation 1, and b) the "relaxed" translation constraints, Equation 2 with each U_{ij} replaced by its convex hull.

For the minimum enclosure algorithm, if the algorithm finds a solution hypothesis, it compacts the non-overlapping layout into a local minimum area container. This establishes an upper bound on the area of the container which is also a local minimum.

2.5 Subdivision.

Subdivision has two cases based on the output of evaluation. If the translations satisfy Equation 2, then the output is a non-overlapping layout of the range intersections $P_i(\alpha_i, \beta_i)$ (Lemma 2.1). It is necessary in this case to subdivide an angle range. If the output does not satisfy Equation 2, then the algorithms use essentially the same subdivision algorithm as the translational containment algorithm. We explain the difference in the full paper.

To subdivide an angle range, the algorithm selects the polygon P_i whose total overlap with the other polygons is maximal at its current translation and rotation (the output of evaluation). If θ_i is closer to the midpoint of $[\alpha_i, \beta_i]$ than the endpoints, then the algorithm divides the range at θ_i. Otherwise, it divides the range at its midpoint. Let the two new ranges be $[\alpha_i', \beta_i']$ and $[\alpha_i'', \beta_i'']$. To generate H', it generates the root hypothesis for $[\alpha_i, \beta_i]$ replaced by $[\alpha_i', \beta_i']$ and adds these constraints to H. Generating H'' is analogous.

2.6 Minimum Area Rectangle Enclosure.

The preceding sections have described the whole of the containment and minimum enclosure algorithms for the case of a rectangle with fixed width and variable length or the case of scaled copies of a fixed convex polygon. The case of minimum area rectangle with arbitrary length and width requires a more sophisticated root hypothesis. We summarize the idea here. It is necessary to add an additional "container" polygon P_{k+1} where $U_{0,k+1}$ is the region $xy \leq A$. This enforces that that container an arbitrary rectangle with area no greater than A. To keep everything polygonal, we approximate the hyperbola $xy = A$ by a polygon.

[2]Replacing U_{ij} by a subset is equivalent to adding more constraints since the previous constraints must still hold.

3 Range Intersection

Section 2.2 gave a definition for the *range intersection* $P(\alpha, \beta)$ of a polygon P: the intersection of all copies of P as it is rotated from angle α to angle β about the origin. The range intersection is an "arc-gon": a planar region bounded by line segments and circular arcs. In particular, these circular arcs are concentric with the origin.

The set intersection, convex hull, and even Minkowski sum of arc-gons is an arc-gon, and therefore it is possible in principle to implement all the operations used in Section 2 on arc-gons. However, generalizing the overlap minimization/compaction algorithm to arc-gons would require a switch from linear program to quadratic or convex programming. Also, it is difficult to handle the intersection of circular arcs in a numerically robust fashion, which would make it difficult to implement the intersection and Minkowski sum of arc-gons.

This section establishes that the complexity of the range intersection $P(\alpha, \beta)$ is linear in the complexity of P. It gives a way to approximate arc-gons by polygons in a way that is suitable for the containment/enclosure algorithms. In particular, the approximation is a) a subset b) whose error is proportional to $(\alpha - \beta)^2$ and c) whose complexity is also linear. Finally, it gives the "practical" implementation we use to construct approximate range intersections.

3.1 Theoretical Complexity.

Lemma 3.1 *The range intersection $P(\alpha, \beta)$ has complexity $O(|P|)$.*

Proof. Construct a *concentric trapezoidalization* of P which is analog to the standard trapezoidalization of P. From each vertex v of P extend an arc (centered at the origin) in each direction (clockwise or counter-clockwise) that goes into the interior of P. Terminate the arc when it hits the boundary of P. This procedure cuts P into a linear number of *concentric trapezoids*: regions bounded by two line segments and two concentric circular arcs.

The intersection $P(\alpha, \beta)$ is a rotated copy of $P(0, \gamma)$, where $\gamma = \beta - \alpha$. The set $P(0, \gamma)$ is equivalent to the set of counter-clockwise endpoints of arcs of angle γ which lie inside P. It is clear that each such arc lies in a single concentric trapezoids. Therefore, the range intersection of each concentric trapezoids can be computed independently in constant time. It is also an concentric trapezoids. The range intersections of the concentric trapezoids meet only at their boundaries and in a nice fashion. It is easy to show that the complexity of the boundary of their union is also linear: a linear number of arcs and line segments. □

3.2 Polygonal Approximations.

In order to compute a polygonal approximation to $P(\alpha, \beta)$, it is necessary to perform a *radial trapezoidalization*. From each vertex v of $P(\alpha, \beta)$, extend a line towards the origin

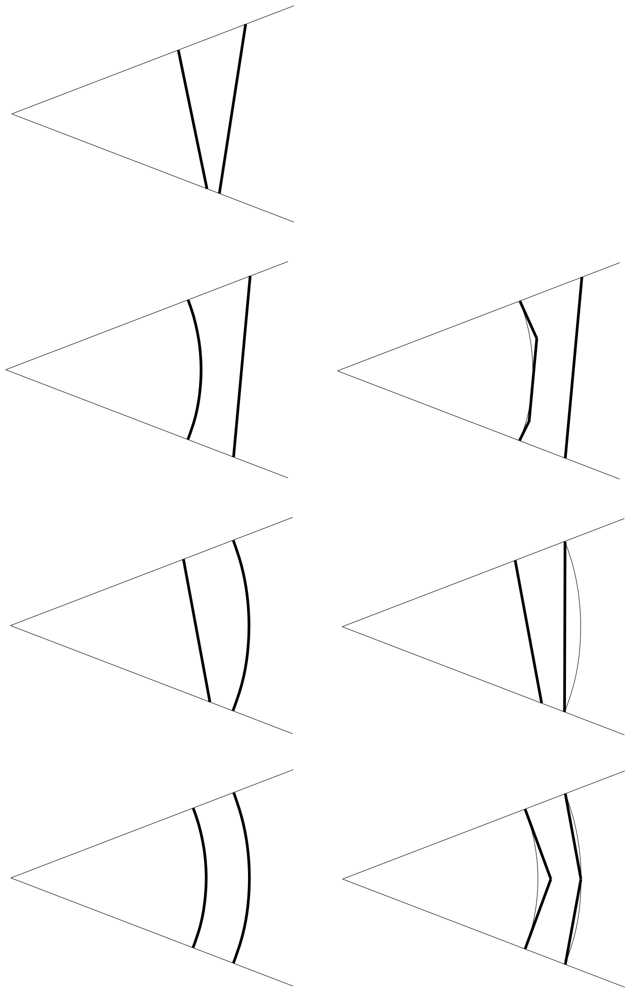

Figure 1: Radial trapezoids and their inner polygonal approximations.

and/or away from the origin, in each direction that goes into the interior of $P(\alpha, \beta)$. Terminate a line when it hits the boundary.

Figure 1 shows the four types of radial trapezoids and how to form inner polygonal approximations. The most complex case has an inner arc and outer line segment. We approximate the inner arc by three tangent line segments: two of them tangent at the endpoints and the middle one parallel to the outer bounding line segment. In the case of two arcs, the polygonal approximations can intersect. In this case, we lose the middle part of the arc, and the topology changes as illustrated in Figure 2.

Lemma 3.2 *Let $B(r)$ be the ball of radius r. Let $Q(\alpha, \beta)$ be the polygonal approximation to the range intersection $P(\alpha, \beta)$. For $|\gamma|$ sufficiently small ($\gamma = \beta - \alpha$),*

$$Q(\alpha, \beta) \subseteq P(\alpha, \beta) \subseteq Q(\alpha, \beta) \oplus B(\mathrm{O}(\gamma^2)).$$

Proof. Clearly the approximation is a subset. It can be shown that $P(\alpha, \beta)$ can have no arc in its boundary longer than $|\gamma|$. Basic trigonometry shows that a chordal approximation to an outer arc has error at most $1 - \cos(\gamma/2) \approx \gamma^2/4$ times the radius. A tangential approximation to an inner arc has error at most $1/\cos(\gamma/2) - 1 \approx \gamma^2/4$ times its radius.

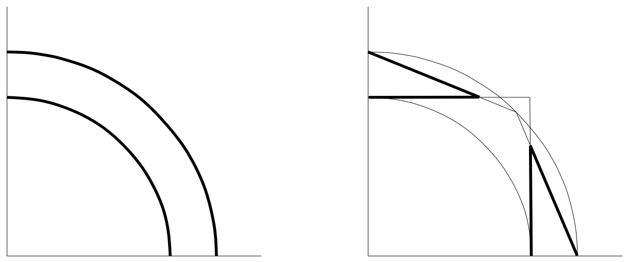

Figure 2: Polygonal approximation of two-arc trapezoid can have different topology.

The difference between the inner and the outer radius is at least the thickness of the thinnest concentric trapezoid of the original trapezoidalization of P. This value is independent of γ. Therefore, for sufficiently small γ, the topological change of Figure 2 does not occur. Each individual radial trapezoid approximation satisfies the condition of the lemma, and therefore their union does too. □

Lemma 3.2 is exactly the property we need to ensure that we can substitute the polygonal approximation to the range intersection for the actual range intersection in the containment/enclosure algorithms. As the containment algorithm subdivides the angle ranges, the accuracy of the approximation increases with the square of the angle range. Thus we expect to have to divide a given angle range no more than b times to obtain accuracy at least $\epsilon = 2^{-b}$.

3.3 Practical Algorithm.

In the full paper we describe the "practical algorithm" we use in lieu of constructing a full circular trapezoidalization. We decompose the complement $\overline{P}$ into *angularly monotonic* regions: each ray out of the origin intersects the region in a single line segment, a points, or the empty set. This implies that a monotone component M is bounded by two functions of angle:

$$M = \{(r, \theta) \mid f(\theta) \leq r \leq g(\theta)\},$$

in polar coordinates. We compute the minimum of $f()$ over a angle "window" of size α as the window rotates about the origin. Similarly, we compute the maximum of $g()$ over a sliding window of angle α. We compute an inner polygonal approximation to the minimum curve and an outer polygonal approximation to the maximum curve. The two curves bound an approximation to the range union $M(0, \gamma)$ of the component M. Finally, we take the complement of the union of the range unions.

4 Nearest Pair Rounding

In previous work, [26, 25, 30, 31], we give methods for *geometric rounding* of line segment arrangements: reducing the precision of the coordinates while maintaining a consistent topology. Geometric rounding is absolutely necessary in cases of *cascading*: the output of one operation repeatedly becomes the input to another. This section summarizes a new geometric rounding technique: *nearest pair rounding*. We use this technique in our current implementations of polygon intersection and Minkowski sum.

Like our previous method, *shortest path rounding*, nearest pair rounding can round to the non-uniform lattice of points with floating point coordinates. Other rounding methods [14, 17], can only round to the integer lattice. Nearest pair rounding is much simpler than shortest path rounding, but it may have larger numerical error. We also discuss an experimental method in which we do not use exact arithmetic at all. So far this has worked well, even under the containment algorithm "torture test".

4.1 Exact Rounding Algorithm.

Nearest pair rounding repeats a rounding step until no more rounding is necessary. Each step "rounds together" the nearest pair of non-incident features: vertex/vertex or vertex/edge. If after rounding, any pair of edges are intersecting, a new vertex is created as near as possible to their actual intersection, and rounding resumes. Rounding stops when every pair of non-incident features is farther apart than some threshold and when no two edges are intersecting.

In the vertex/vertex case, the rounding step creates new vertex at the midpoint of the two vertices and moves both vertices to that new vertex along with an edges that have them as an endpoint. In the vertex/edge case, a new vertex is created halfway from the vertex to the edge. The original vertex is moved to this new vertex, along with edges that have it as an endpoint. The original edge is broken into two edges each which have the new vertex as one of their endpoints. The rounding step does not consider a vertex/edge pair unless the vertex forms an angle less than 45 degrees with either endpoint.

Theorem 4.1 *Nearest pair rounding converges without introducing an new intersections.*

Proof. The key to nearest path rounding is that the perturbation it introduces is at most 3/4 of the minimum distance between non-incident features. Therefore, applying the rounding step cannot cause other non-incident features to "collide". Because we ignore vertex/edge pairs unless the angles with the endpoints are less than 45 degrees, the measured minimum feature separation may be $\sqrt{2}$ times the actual minimum. However, rounding move features to a midpoint, and therefore the actual perturbation is at most $\sqrt{2}/2 < 3/4$ times the actual minimum feature separation.

Consider a partial order on sets of line segments, ordered on: 1) number of edge/edge intersections, 2) number of vertices, $3, 4, 5, \ldots$) length of longest edge, length of second longest edge, and so forth. Adding an intersection point decreases (1). Rounding two vertices together decreases (2) without increasing (1). Because of the 45 degree rule, breaking an edge forms two shorter edges without increasing (1) or (2). □

4.2 Experimental Rounded Arithmetic Rounding Algorithm.

In practice, we use floating point arithmetic for all operations and apply nearest pair rounding without using exact arithmetic to measure distances. Because the rounding step has $1 - 3/4 = 25\%$ "leeway" for error, we expect it to be numerically stable. So far, it has been stable even after hundreds of cascaded operations.

5 Results

All algorithms were implemented in C++. Overlap minimization code: 9100 lines. Polygon set operations: 2700 lines. Containment/Enclosure algorithms: 1200 lines. Overlap minimization uses CPLEX 4.0 by CPLEX Optimization, Inc. Experiments were run on a DEC Alpha 3000/700 which has roughly the same speed as a 266MhZ Intel Pentium II.

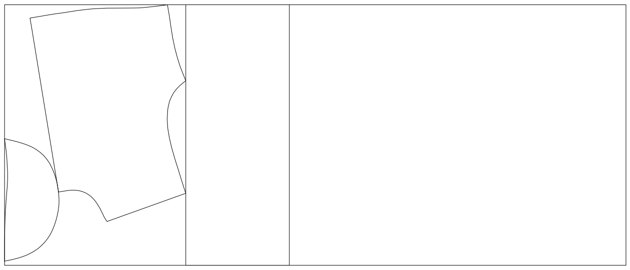

Figure 3: Target length: 2519. Output length after containment and leftward compaction: 1749.750.

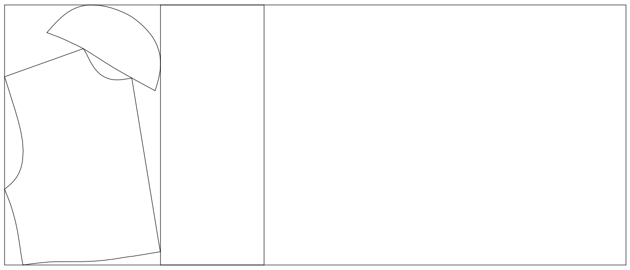

Figure 4: Target length: 1749. Output length after containment and leftward compaction: 1505.643.

5.1 Two Polygons

For the two-polygon example, we alternated containment with leftward compaction. Each positive example in Figures 3 through 5 ran in under a minute. There are still so many checks, prints, and dumps in the program, that running times under a minute are not very accurate.

Finally, in about 18 minutes, the containment algorithm evaluated 191 hypotheses and verified that there was no solution shorter than 1386.

5.2 Three Polygons

We ran the algorithm on industrial parts with 71, 65, and 64 vertices. The good news is that it appeared to find the minimum length enclosure quickly as shown in Figures 6 though 8.

The bad news is that we challenged the system with a third input length of 2173 (1% less than previous output) and it took more than 5 hours to evaluate 811 hypotheses and verify that there was no solution with this length. We did not even try running it with a target closer to the previous output, as we did in the two-polygon example.

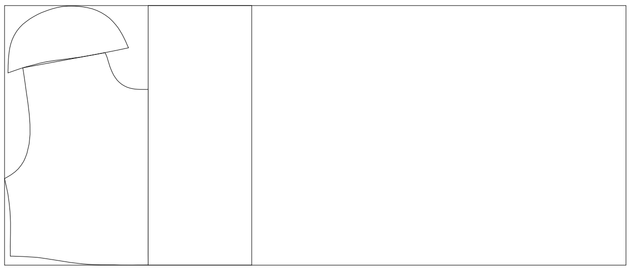

Figure 5: Target length: 1505. Output length after containment and leftward compaction: 1386.546.

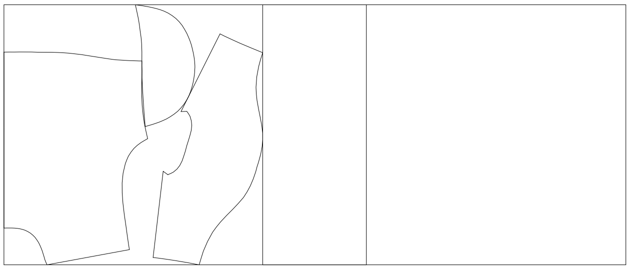

Figure 6: Hand placed and compacted leftwards. Length: 2496.542.

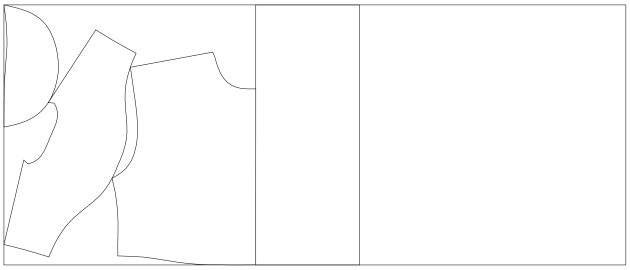

Figure 7: Target length: 2496. Output length after containment and leftward compaction: 2429.647. Time: 42 seconds.

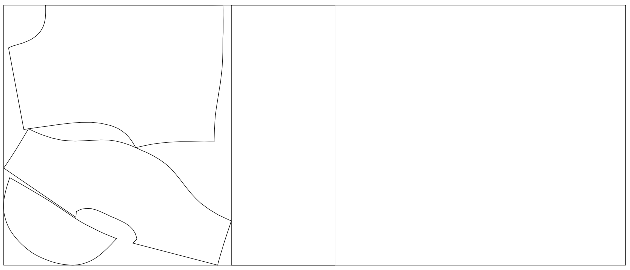

Figure 8: Target length: 2429. Output length after containment and leftward compaction: 2196.433. Time: 96 seconds.

6 Conclusion

At present, the range intersection method appears to be practical for two-polygon minimum enclosure. For three polygons, it appears to generate a minimum quickly but require a very long time to verify it.

Some modifications might speed up the algorithm. For instance, it can search the hypothesis space for each individual polygon and then each pair before considering three at time. That might speed up the three-polygon example somewhat because it appeared to sometimes make a bad choice for on polygon's angle and then spend much time investigating all hypotheses with this angle.

Another conclusion is that it is worth pursuing the "true generalization" of translational containment described in Section 1.3. This will involve performing complex set operations on curved three dimensional objects. It will probably be difficult to make these operations robust, but the current results demonstrate that these algorithms would be worth the effort. For instance, in the special case of two polygons, the true generalization would be essentially the same as Avnaim and Boissonnat's formula and thus solve this case exactly. Our experience with translational work would indicate that it could solve at least the three-polygon case and possibly the four-polygon case also in a reasonable amount of time.

Finally, we should not forget the first paragraph in which it was mentioned that apparel manufacturers only allow small tilts. All experiments in this paper were run with a full 360 degrees of rotation allowed. With only small rotations allowed, the range intersection algorithm should be practical for much larger values of k.

References

[1] P. Agarwal, N. Amenta, and M. Sharir. Largest placement of one convex polygon inside another. In *Proceedings of the Second Workshop on Algorithmic Foundations of Robotics*, July 1996.

[2] P.K. Agarwal, B. Aronov, and M. Sharir. Motion planning for a convex polygon in a polygonal environment. In preparation., 1992.

[3] F. Avnaim. *Placement et déplacement de formes rigides ou articulées.* PhD thesis, Université de Franche-Comté, France, 1989.

[4] F. Avnaim and J. Boissonnat. Polygon placement under translation and rotation. In *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science; Lecture Notes in Computer Science (294)*, pages 322–333. Springer-Verlag, 1988.

[5] C. Bounsaythip and S. Maouche. Irregular shape nesting and placing with evolutionary approach. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation (Cat. No.97CH36088-5*, page 5. IEEE; New York, NY, USA, 12-15 October 1997.

[6] B. Chazelle. The Polygon Containment Problem. In Preparata, editor, *Advances in Computing Research, Volume 1: Computational Geometry*, pages 1–33. JAI Press, Inc., Greenwich, Connecticut, 1983.

[7] K. Daniels. *Containment algorithms for nonconvex polygons with applications to layout.* Ph.D. thesis, Harvard University, Cambridge, MA, 1995.

[8] K. Daniels and V. Milenkovic. Column-based strip packing using ordered and compliant containment. In *Proc. 1st ACM Workshop on Appl. Comput. Geom.*, pages 33–38, 1996.

[9] K. Daniels and V. J. Milenkovic. Multiple Translational Containment, Part I: An Approximate Algorithm. *Algorithmica*, 19:148–182, 1997.

[10] K. A. Dowsland and W. B. Dowsland. Packing Problems. *European Journal of Operational Research*, 56:2 – 14, 1992.

[11] K.A. Dowsland and W.B. Dowsland. Solution approaches to irregular nesting problems. *European Journal of Operational Research*, 84(3):506–21, August 1995.

[12] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operations Research*, 44:145–159, 1990.

[13] R. M. S. Abd El-Aal. A new technique for nesting irregular shapes based on rectangular modules. *Current Advances in Mechanical Design and Production 1996*, 6:533–540, 1996.

[14] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[15] R. Grinde and T. Cavalier. Containment of a Single Polygon Using Mathematical Programming. Technical Report IMSE Working Paper 92-164, The Pennsylvania State University, Department of Industrial and Management Systems Engineering, 1993.

[16] R. B. Grinde and T. M. Cavalier. A new algorithm for the two-polygon containment problem. *Computers and Operations Research*, page to appear, 1996.

[17] Leonidas Guibas and David Marimont. Rounding arrangements dynamically. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 190–199, 1995.

[18] G. C. Han and S. J. Na. Two-stage approach for nesting in two-dimensional cutting problems using neural network and simulated annealing. *Journal of Engineering Manufacture*, 210(B6):509–19, 1996.

[19] R. Heckmann and T. Lengauer. Computing upper and lower bounds on textile nesting problems. *LECTURE NOTES IN COMPUTER SCIENCE*, 0302-9743(1136):392–405, 1996.

[20] Khoi Hoang. Aspects in automatic nesting of irregular shapes. In *Proc. SPIE - Int. Soc. Opt. Eng. (USA), Proceedings of the SPIE - The International Society for Optical Engineering, vol.2589*, pages 234–41. SPIE-Int. Soc. Opt. Eng, 23-26 October 1995.

[21] P. Jain, P. Fenyes, and R. Richter. Optimal blank nesting using simulated annealing. *Journal of mechanical design*, 114(1):160–165, March 1992.

[22] H. J. Lamousin, W. N. Waggenspack, and G. T. Dobson. Nesting of complex 2-d parts within irregular boundaries. *Journal of Mechanical Design*, 118(4):615, 1996.

[23] Z. Li. *Compaction algorithms for nonconvex polygons and their applications.* Ph.D. thesis, Harvard University, Cambridge, MA, 1994.

[24] Z. Li and V. Milenkovic. Compaction and separation algorithms for nonconvex polygons and their applications. *European Journal of Operations Research*, 84:539–561, 1995.

[25] V. Milenkovic. Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 500–505, 1989.

[26] V. Milenkovic. Rounding face lattices in the plane. In *Abstracts 1st Canad. Conf. Comput. Geom.*, page 12, 1989.

[27] V. Milenkovic. Translational polygon containment and minimal enclosure using linear programming based restriction. In *Proc. 28th Annu. ACM Sympos. Theory Comput. (STOC 96)*, pages 109–118, 1996.

[28] V. Milenkovic and K. Daniels. Translational polygon containment and minimal enclosure using geometric algorithms and mathematical programming. Technical Report 25-95, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, Cambridge, MA, 1995.

[29] V. J. Milenkovic. Multiple Translational Containment, Part II: Exact Algorithms. *Algorithmica*, 19:183–218, 1997.

[30] Victor J. Milenkovic. Practical methods for set operations on polygons using exact arithmetic. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 55–60, 1995.

[31] Victor J. Milenkovic. Shortest path rounding. *Algorithmica*, page (submitted), 1997.

[32] Victor J. Milenkovic. Rotational polygon overlap minimization. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*. ACM, 1997 (Journal version accepted with revisions by *Computational Geometry: Theory and Applications*).

[33] V.J. Milenkovic and K. Daniels. Translational Polygon Containment and Minimal Enclosure using Mathematical Programming. *International Transactions in Operational Research (accepted with revisions)*, 1997.

[34] Y.K.D.V. Prasad. A set of heuristic algorithms for optimal nesting of two- dimensional irregularly shaped sheet-metal blanks. *Computers in Industry*, 24(1):55–70, May 1994.

[35] Y.K.D.V. Prasad, S. Somasundaram, and K.P. Rao. A sliding algorithm for optimal nesting of arbitrarily shaped sheet metal blanks. *International Journal of Production Research*, 33(6):1505–20, June 1995.

[36] P. E. Sweeney and E. R. Paternoster. Cutting and Packing Problems: A Categorized, Application-Oriented Research Bibliography. *Journal of the Operational Research Society*, 43(7):691–706, 1992.

[37] J. Y. Wang, D. Y. Liu, E. W. Lee, and T. H. Koh. An algorithm for nesting patterns in apparel. In *COMPUTER INTEGRATED MANUFACTURING - INTERNATIONAL CONFERENCE- WORLD SCIENTIFIC 1995; VOL 1*, pages 377–384. World Scientific; 1995, 1995.

[38] P.F. Whelan and B.G. Batchelor. Automated packing systems: review of industrial implementations. In *Proceedings of the SPIE - The International Society for Optical Engineering, vol.2064*, pages 358–69. SPIE, September 1993.