DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Compaction and Separation Algorithms for Non-Convex Polygons and Their Applications

Citation

Li, Zhenyu and Victor Milenkovic. 1993. Compaction and Separation Algorithms for Non-Convex Polygons and Their Applications. Harvard Computer Science Group Technical Report TR-13-93.

Permanent link

http://nrs.harvard.edu/urn-3:HUL.InstRepos:25620473

Terms of Use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Compaction and Separation Algorithms for Non-Convex Polygons and Their Applications*

Harvard University
Center for Research in Computing Technology
Aiken Computational Laboratory
33 Oxford Street

Cambridge, MA 02138

Zhenyu Li † *Victor Milenkovic*‡

Abstract

Given a two dimensional, non-overlapping layout of convex and non-convex polygons, *compaction* can be thought of as simulating the motion of the polygons as a result of applied "forces." We apply compaction to improve the material utilization of an already tightly packed layout. Compaction can be modeled as a motion of the polygons that reduces the value of some functional on their positions. *Optimal compaction*, planning a motion that reaches a layout that has the global minimum functional value among all reachable layouts, is shown to be NP-complete under certain assumptions. We first present a compaction algorithm based on existing physical simulation approaches. This algorithm uses a new *velocity-based* optimization model. Our experimental results reveal the limitation of physical simulation: even though our new model improves the running time of our algorithm over previous simulation algorithms, the algorithm still can not compact typical layouts of one hundred or more polygons in a reasonable amount of time. The essential difficulty of physical based models is that they can only generate velocities for the polygons, and the final positions must be generated by numerical integration. We present a new *position-based* optimization model that allows us to calculate directly new polygon positions via linear programming that are at a local minimum of the objective. The new model yields a translational compaction algorithm that runs two orders of magnitude faster than physical simulation methods. We also consider the problem of separating overlapping polygons using a minimal amount of motion and show it to be NP-complete. Although this *separation* problem looks quite different from the compaction problem, our new model also yields an efficient algorithm to solve it. The compaction/separation algorithms have been applied to marker making: the task of packing polygonal pieces on a sheet of cloth of fixed width so that total length is minimized. The compaction algorithm has improved cloth utilization of human generated pants markers. The separation algorithm together with a database of human-generated markers can be used for automatic generation of markers that approach human performance.

Keywords: Packing, Cutting, Compaction, Automated layout generation, Minkowski sums, Computational geometry, Physically based simulation, Database driven automatic layout generation, Garment manufacturing.

---

*A previous version of this paper was presented in the Proceedings of the Ninth Annual Symposium on Computational Geometry, ACM Press

†This research was funded by the Textile/Clothing Technology Corporation from funds awarded to them by the Alfred P. Sloan Foundation. Author's current address: GTE Laboratories Incorporated, 40 Sylvan Road, Waltham, MA02254

‡This research was funded by the Textile/Clothing Technology Corporation from funds awarded to them by the Alfred P. Sloan Foundation and by NSF grants CCR-91-157993 and CCR-90-09272.

1 Introduction

1.1 Definitions

In this paper, a two dimensional *layout* is defined to be the placement of a set of polygonal pieces on a rectangular sheet of material. If the pieces are to be cut from the material, it is important that they not be allowed to overlap each other or to cross the boundary of the material. The efficiency (utilization) of a non-overlapping layout is the ratio of the area covered by the pieces to the total area of the material. In the apparel industry, the layout sheet has fixed width (corresponding to the width of a bolt of cloth) and arbitrary length, and a non-overlapping layout is called a *marker*. Figure 1 shows a pants marker generated by a human with efficiency of 89.62%. Optimal marker making, finding a placement of a given set of pieces with highest efficiency, is a hard problem. This paper considers the simpler problem of planning motions of the pieces in an already existing layout to improve the layout in some fashion: increase efficiency by shortening the length, open up space for new pieces without increasing the length, or eliminate overlap among pieces. We use the term *compaction* for any motion planning task that stays entirely within the set of feasible (non-overlapping) configurations, such as the first two tasks. We use the term *separation* for a task that moves the configuration from an overlapping to a non-overlapping configuration. Ultimately, we introduce a compaction/separation optimization model that encompasses both types of motions and other interesting combinations.

1.2 Applications

Compaction and separation can be applied to perform a variety of marker layout tasks. Leftward or downward compaction reduces the area of a layout consisting of a fixed number of pieces. This process involves moving the right boundary leftward to shorten the length of the layout in the x-direction, or moving the top boundary down to shorten the length in the y-direction, or both. Figure 2 shows the leftward compaction of the marker in Figure 1, which improves the efficiency of the marker by .94%. Another application is opening up gaps (free space between pieces) as an aid to the task of fitting as many pieces as possible on fixed area material. One repeatedly finds gaps, and in case a gap is not large enough to hold a piece, one opens up the gap by pushing away the surrounding pieces. Figure 3 shows compaction opening up a gap as a result of applied forces shown by the arrows. Two new polygons (which were not part of the motion planning but are shown for illustration purposes) can be placed without overlap after compaction is applied. The same result can be obtained by putting the polygons into the overlapped positions and requesting a *separation* without specifying forces. In fact, our algorithm does both separation and compaction: it will eliminate overlaps and then optimize motion in the direction of all specified forces. These compaction/separation functions are very important when the pieces in the layout are to be cut from expensive materials, as in aircraft, automobile and apparel manufacturing.

This research grew from our current project [20] to generate pants markers automatically with an efficiency competitive with those generated manually by experienced humans. Humans with many years of experience can reliably generate pants markers whose efficiency is in the range 89-91% and probably within 1% of optimal. Our algorithm for panel (large piece) placement [21] can exceed human performance at least 60% of the time, and the overall average efficiency after automatic panel placement and compaction is slightly better than the human average. We plan to use the compaction techniques described here to improve the efficiency of our machine generated markers and also to open up gaps during trim (small piece) placement. Currently, compaction is being put to use in industry to improve human-generated layouts.

Since many garment pieces will be cut using the same marker, a small percentage increase in efficiency per marker results in a large savings in material cost. For pants manufacturing, each time a marker is used, sixty layers of cloth are cut simultaneously, and a 1% increase in efficiency represents a savings of about $25. A representative of a large pants manufacturer estimates that a 0.1% average improvement in efficiency will save his company about two million dollars per year. Experienced human marker makers can generate high efficiency markers. However, our experiments show that almost all production quality markers, pants or other garments, can be improved in efficiency by our compaction algorithm.

The task of improving efficiency proves to be extremely hard for humans. Production quality markers

are packed very tightly: every piece is touching its surrounding pieces, so there is no room to move just one piece. To shorten the marker length, one needs to find a coordinated motion of all the pieces.

Another difficult task is to move pieces to open up a gap somewhere in a marker without changing the bounding box. Human marker makers need to open up gaps to place trim pieces. Even experienced marker makers find the task of opening gaps in tightly packed markers time-consuming and awkward. It is unlikely that they are finding the maximal sized gap since they can only move one piece at a time. A compaction algorithm that can efficiently open up a gap can be a valuable tool for human marker makers.

The above mentioned compaction tasks all address tightening of a layout. In marker making applications, we would encounter the needs for *separation*. More specifically, there can be two occasions when a separation is necessary or desirable. The first relates to resolving overlaps in markers. Some markers may have overlapping pieces and we want to eliminate the overlaps by moving pieces away from each other. When necessary, we would even increase the current length of the marker to give more space for resolving overlaps. The second occasion involves distributing the free space uniformly among the pieces. Frequently, the overall length of a marker is determined by a critical path connecting a piece on the left boundary to a piece on the right boundary. It is difficult or sometimes impossible to improve the critical path. However, some of the other pieces can have relatively large free spaces around them. These free spaces can be used to increase the minimum distance among adjacent pieces. This action is given the term *floating*. Floating can facilitate the cutting process. Figure 4 shows an example of floating.

Separation for overlap resolution in turn has an application in database-driven automated marker making. The underlying idea is to collect high quality markers generated by experienced human marker makers into a database. When a new marker making order comes in, we look at the database for a similar marker. Then we match the pieces and do a piecewise substitution of the corresponding pieces. The substitution process will inevitably introduce overlaps. By applying separation and leftward compaction, we can generate a marker of sufficient quality automatically.

1.3 Related Work

As far as we know, no efficient algorithm has been previously published for compaction or separation of a large number of non-convex polygons (see [31] for a survey of current results). In industry, several CAD firms have tried "one piece at a time" methods which are not successful in practice.

Previous research efforts in compaction are largely concentrated in the field of VLSI design [2] [14] [19] [23] [28] [32]. The research is focused on compacting a layout consisting of rectangles and sometimes variable-length rectilinear wires. We found the techniques not readily extendible to non-rectangles or not applicable in our case in which the layouts are already tightly packed.

Physically based simulation methods provide another approach to compaction. They simulate the pieces as rigid bodies and apply a set of forces to move the bodies in the desired directions. A local minimum is reached when the bodies cannot be moved further. These simulation methods fall into two types: *spring* model and *contact force* model. The difficulty we have found with physically based simulation methods is that these algorithms run very slowly.

Spring model methods (also called *penalty methods*) [22] [25] allow the pieces to inter-penetrate slightly. A restoring force is determined by the amount of inter-penetration. From these forces, the motion of the pieces is computed by numerical integration. This integration is carried out by cutting time into steps. Small steps are required to ensure accuracy and numerical stability of the integration; however, the smaller the step, the greater the computational cost. It is generally very difficult to choose the correct step size, especially when many different forces are involved. Layout compaction involves hundreds of frequently changing contacts, and for this reason, the spring model is not suitable for compaction.

Contact force model methods (also called *analytical methods*) [3] [4] have been recently studied in computer graphics. Contact forces are introduced at contact points to prevent inter-penetration. A system of differential equations is solved to find a set of consistent contact forces. The bodies then move with constant velocities in accordance with the contact forces. A collision detection algorithm is used to find the time-interval until the next collision time. The simulation proceeds from time-interval to time-interval until the local minimum is reached. Contact force methods have the advantage that the time-intervals are much larger and more easily determined than the time-steps of the spring model methods. However, there

still are difficulties. Solving the system of differential equations for determining the contact forces is time-consuming for problems of non-trivial size. Moreover, the determination of consistent contact forces is itself NP-complete if there are vertex-vertex contacts [3] [24]. Finally, we find that in the case of many contacts the local minimum is only reached after many time-intervals because the pieces tend to "rattle" against each other.

Stoyan[29] [30] uses an optimization approach similar to our first velocity-based optimization model (see next section). For polygons with a large number of vertices (more than twenty), he simplifies the polygons and the configuration space to overcome the costs inherent in the velocity-based approach. Thus his technique generates an approximate compaction.

1.4 Results

The main contributions of this paper are two optimization models which yield algorithms for performing various compaction and/or separation tasks: a *velocity-based* model and a *position-based* model.

The velocity-based model allows us to remove the concept of mass and force and directly compute velocities. This model yields a new, more efficient time-interval based simulation technique for translational compaction. We had expected that direct computation of velocities would allow our algorithm to solve the compaction problem in a reasonable amount of time. Unfortunately, even though our new simulation techniques reduced the running time by at least an order of magnitude over previous methods, the algorithm still took hours to run on typical markers on a 28MIPS workstation because of the number of times it has to recompute velocities. This can be viewed as a important negative result that demonstrates the inherent limitation of the physically based approach.

Our second, position-based model yields a compaction algorithm that directly computes a new set of positions for the polygons without using velocities, time-steps, time-intervals or any sort of simulation. Instead, this new model contains artificial constraints that select a convex feasible subset of the solution space and thus make a direct calculation possible via linear programming. The artificial constraints are based on the configuration space concept from the field of robotics plus a new *locality heuristic*. Because of the extra constraints, the algorithm does not immediately reach a local minimum of the original compaction problem, but a small number (less than five in practice) of repetitions suffices to reach the local minimum. Using this model, our compaction algorithm runs in almost real time in practice and solves industrially significant problems.

Our new position-based model also yields a separation algorithm that eliminates overlaps in a marker. Originally, compaction was our only goal, but we noticed that the compaction algorithm also accomplished separation. This occurred because the slack variables for each non-overlapping constraint in our model exactly correspond to the amount of overlap between pairs of polygons. We realized that we could make these variables explicit in the constraints and add them to the original objective of our model. The resulting position-based model can eliminate overlaps, apply forces, and even float pieces away from each other (negative overlap) in any desired combination. The separation algorithm has been applied in database driven automated marker generation. A CAD company in the textile industry is now vigorously pursuing the database driven automated marker making idea.

1.5 Organization of the Paper

The following section formalizes the compaction problem and discusses its complexity. Section 3 describes a physically based simulation method for compaction. This technique solves a velocity-based optimization model to generate a set of velocities for the set of polygons and then simulates motion until a new contact occurs. We discuss its long running time for tightly packed markers and why that is an inherent drawback of physically-based methods. Section 4 gives background knowledge about configuration spaces and Minkowski sums which play a central role of in the position-based model. Section 5 describes this new model and gives an algorithm based on it that directly determines new positions for the polygons. As a result, this algorithm does not use simulation nor does it have time as an explicit parameter. Section 6 defines the problem of separating overlapping polygons, studies its complexity, and shows how the position-based model yields a separation algorithm that finds a locally optimal solution to this problem. Section 7 demonstrates that by

combining the separation algorithm and a database of human generated markers, we can have an automated marker making system that very quickly generates markers close to human performance. Finally, Section 8 gives empirical evidence for the efficiency of the compaction algorithm in marker making. It also describes how it fits into our current project for automatically generating pants markers.

2 Statement of Compaction Problem

In order to study its complexity, we find it convenient to define the two-dimensional compaction problem as a motion planning problem for polygons in a rectangular container in which all the polygons can move simultaneously and only the right boundary of the container is movable. Inter-penetration between the polygons or between the polygons and the container is not allowed during the motion. The goal is to minimize the area of the container. A translational compaction problem is a compaction problem in which the polygons are not allowed to rotate.

Theorem 2.1 *Two-dimensional translational compaction is* NP-hard *even for rectangles.*

Proof: We reduce the NP-hard PARTITION problem to compaction. The PARTITION problem is defined as follows: given a set S of integers, decide if S can be partitioned into two subsets S_1 and S_2 such that the sum of the elements of S_1 equals the sum of the elements of S_2. We reduce PARTITION to compaction by the construction shown in figure 5. For an instance $\{a_1, a_2, \ldots, a_n\}$ of PARTITION, we build rectangles of height 1 and width a_i, $(i = 1, \ldots, n)$. We put the rectangles into a container of height 2. PARTITION has a solution if and only if the blocks can be compacted to the length $\sum a_i/2$. ■

To answer the question of whether the translational compaction problem is in NP, we must specify the form of the output of compaction. If we specify that the output is only the final configuration, then we show later in this section that translational compaction is not in NP: the actual motion might require an exponential size description, and there may be no polynomial time verification that the final configuration is reachable by a non-overlapping motion. If we specify that the output must include the description of the motion, then translational compaction is in NP. For the types of layouts we have encountered, the two specifications are equivalent because the layouts are "rightward disassemblable": they can be taken apart by repeatedly moving some piece to the right. For such layout problems, a reachable final state always has a motion with a polynomial size description.

The following lemma shows that if there is a translational compaction whose motion has a polynomial sized description, then there is a compaction which we can verify in polynomial time. This implies that the set of layouts which can be compacted with a polynomial sized motion is NP-complete.

Theorem 2.2 *For a layout of polygons, if there is a motion along piecewise algebraic curves with a polynomial size description (in other words, this motion can be described by polynomial spline curves) from one configuration of a layout to another, then there is a piecewise linear motion with a polynomial sized description which results in a configuration with the same or better efficiency.*

Proof: Given a layout with n polygons, the configuration space for non-overlapping translational motion is $2n$-dimensional. For each pair of polygons P and Q, the constraint that vertex A of P lies on the "correct" side of edge BC of Q represents a half-space in the configuration space. There is only a polynomial number of such vertex-edge half-spaces. The configuration space of non-overlapping positions can be partitioned into convex regions each of which is an intersection of a polynomial number of vertex-edge half-spaces (although there is an exponential number of such convex regions). A piecewise algebraic motion enters and leaves each half-space a polynomial number of times. Therefore, the path visits a polynomial number of convex regions. Given the list of regions visited, one can "straighten out" the motion. When the original path moves from region R_i to R_j, the straightened path passes through the vertex common to R_i and R_j. The final position of the straightened path is the vertex of the final region R_{final} which has the best value of the linear functional that the compaction is supposed to maximize. Each vertex of a convex region is an intersection of $2n$ hyper-planes with polynomial sized descriptions. Each such vertex has a polynomial sized rational representation. Therefore, the correctness of the straightened path can be verified in polynomial time. ■

This result shows that a broad class of compaction problems are in NP.

The second specification of compaction, in which the output of compaction does not include the description of the motion, is proven to be much harder. The reason is that moving from one configuration to another might involve complicated motions of many pieces. A closely related coordinated motion planning problem is the *Warehouseman* problem, which is the problem of deciding whether a set of rectangular objects within a rectangular region can reach a goal configuration from an initial configuration through translations. Hopcroft *et.al,* showed that the warehouseman problem is PSPACE-hard [12].

It may seem that the Warehouseman problem is harder than the compaction problem because the compaction problem need not force a particular final configuration, just one which optimizes the objective. In other words, there might be many configurations that satisfy the goal of compaction which is to shorten the container's length. However, the authors [17] have shown that the Warehouseman problem can indeed be reduced to the compaction problem. Thus, the second specification of compaction is PSPACE-hard.

Theorem 2.3 ([17]) *Compaction (without motion description) is PSPACE-hard even for a collection of rectangles when the right boundary of the container is not allowed to move to the right.*

The PSPACE-hardness of the problem shows that moving from one configuration to another by translation alone might still require an exponential number of steps. We have successfully constructed an explicit example that requires exponential number of moves for solving this extended version of the warehouseman problem and the compaction problem [15].

Remark: In principle, Canny's algorithm([5]) can be used to solve the leftward compaction problem exactly in time exponential in the number of degrees of freedom. However, the number of degrees of freedom here is linear in the input size. Hence his algorithm requires exponential time for our application.

3 Compaction Using a Velocity-Based Optimization Model

In this section, we give a compaction algorithm based on a simulation of the translational motion of rigid bodies. This algorithm computes the velocities of the bodies using a velocity-based optimization model. By allowing us to ignore the dynamic properties of the rigid bodies and to concentrate on the kinematics, the optimization model yields a compaction algorithm which is simpler than current contact force model algorithms (see Section 1.3).

A polygon in the layout is viewed as a rigid polygonal body which is governed by two geometric properties: (1) every point on the body has the same velocity; and (2) the body is not penetrable by other bodies. The following optimization model, in which the motions are determined by implicit "pseudo" forces, replaces the usual the Newtonian physics formulation.

We start by assigning a set of velocities v_i, where

$$0 \leq |v_i| \leq 1,$$

and desired directions (the pseudo-forces) f_i to each piece i. The model objective is to maximize

$$\sum_{i=1}^{n} f_i \cdot v_i,$$

i.e., find the velocities that move the pieces in the desired directions as fast as possible. The velocities must satisfy a set of non-penetration constraints. If two pieces are already touching, a non-penetration constraint is set up as follows. Assume vertex A of polygon P is touching edge BC of polygon Q. Polygon P is to be moved with velocity u and polygon Q with velocity v. The points on the boundary of a polygon are ordered counterclockwise, i.e. interior of polygon Q is on the left side of the directed edge BC. After time t, A is moved to $A' = A + ut$ and edge BC is moved to $B'C'$ where $B' = (B + vt)$ and $C' = (C + vt)$. The condition that A' does not penetrate $B'C'$ is:

$$A'B' \times C'B' \geq 0,$$

that is

$$((B + vt) - (A + ut)) \times ((B + vt) - (C + vt)) \geq 0.$$

This simplifies to

$$(B - A) \times (B - C) + (B - A) \times (u - v)t \geq 0$$

and

$$AB \times CB + AB \times (u - v)t \geq 0.$$

Since A was touching BC,

$$AB \times CB = 0,$$

and since $t > 0$, the non-penetration constraint is

$$AB \times (u - v) \geq 0,$$

which is a linear constraint on the velocity. The vertex-vertex touching case is broken into several vertex-edge touching cases.

We build a velocity-based model containing linear constraints for all vertex-edge touching pairs. The resulting linear program is then solved to get a new set of velocities which, at an infinitesimal time-interval, do not cause inter-penetration among the polygons. Next, a collision detection algorithm is used to find the time t_{min} at which the first collision occurs between a vertex-edge pair which was not previously part of the constraints. The simulation proceeds by the time interval t_{min}: polygon i is moved by $v_i \cdot t_{min}$. At this point, some original vertex-edge touching pairs break up and some new pairs are formed. The algorithm repeats the above process until t_{min} found by the collision detection algorithm is 0 which means the polygons cannot move further.

The expensive step in the contact force model methods (see Section 1.3) of solving a system of differential equations is replaced by solving a linear program. Even so, this algorithm is still very slow on practical problems involving more than one hundred polygons, and it is not numerically robust. The main reason for the slow running time is the large number of time intervals it takes before reaching the local minimum. When the layout is getting tight, the chance of collision increases dramatically and the time interval for each step becomes very small. And the more polygons there are in a layout, the more frequently the algorithm takes small time intervals. The small time step can also cause numerical difficulties in setting up constraints and collision detection. For a marker of 45 polygons, the running time is about 40-50 minutes on a 28 mips Sun SparcStation™.

However, we observe that the algorithm is relatively fast in bringing a set of sparsely layed polygons into a relatively tight state. Therefore, it can still serve as a good alternative to the existing physically based simulation methods when dealing with loosely packed layouts. If a relatively tight state is good enough, one could also replace the polygons by simpler outer approximations, as Stoyan has done with his compaction technique (Section 1.3). Our observation is that humans generate very tightly packed layouts, and so the velocity-based algorithm is not a practical way to compact these layouts.

4 The Theory of Minkowski Sum

Our next optimization model is based on the concept of *Minkowski sums*. Minkowski sums are widely used in robot motion planning [5] [18] [26] and in image analysis [27]. Throughout the automated marker making project [15] [20] [21], we have successfully demonstrated the great utility of Minkowski sum in packing and placement problems. Basically, by using Minkowski sums and differences, we can convert a polygon-polygon intersection (overlap) query and a polygon-polygon containment query into point in polygon location queries.

In this paper, the algorithms presented use Minkowski sum only. Therefore, this section concentrates on the concept and properties of Minkowski sum.

4.1 Definitions and Properties

We present the definition of Minkowski sum in set theoretic terms. The lemmas and theorems in this section are valid for discrete or continuous point sets in the Euclidean d–space.

Definition 4.1 Let A and B be two point sets in the Euclidean space. The *Minkowski Sum* of A and B, denoted by $A \oplus B$, is a point set:

$$A \oplus B = \bigcup_{b \in B} A^b$$

where A^b is A translated by b:

$$A^b = \{a + b \mid a \in A\}$$

Lemma 4.1 shows that $A \oplus B$ can be written equivalently as an algebraic sum of A and B. The algebraic sum is sometimes easier to apply. Hence, this lemma can also serve as an alternative definition of Minkowski Sum.

Lemma 4.1

$$A \oplus B = \{a + b \mid a \in A \wedge b \in B\}$$

Proof: ($\Rightarrow$) By definition, $x \in A \oplus B$ implies that there exists $b \in B$ such that $x \in A^b$. It follows that there exists $a \in A$ such that $x = a + b$. Therefore x is also an element of the set on the right hand side.
($\Leftarrow$) Similar. ■

Corollary 4.2 $A \oplus B = B \oplus A$

The following lemma shows that the "shape" of the Minkowski sum is translationally invariant. As a consequence, if the point sets A and B can only change location but not orientation, we need to compute their Minkowski sum just once. When A and B are placed in new locations, their Minkowski sum is translated accordingly.

Lemma 4.3 *Let A and B be two point sets. Let s and t be two points. Then*

$$A^s \oplus B^t = (A \oplus B)^{s+t}$$

Proof: The proof is straightforward from Lemma 4.1. ■

4.2 Application: Intersection Detection

Theorem 4.4 *Let A and B be two point sets and x be a point in the plane. Then $A \bigcap B^x \neq \emptyset$ if and only if $x \in A \oplus (-B)$, where $(-B) = \{-b \mid b \in B\}$ is the reflective image of B with respect to the origin of the global coordinate system.*

Proof: ($\Rightarrow$) Let $y \in A \bigcap B^x$. We have $y \in A$ and $y \in B^x$. That is, there exists $b \in B$ such that $y = b + x$. Therefore, we have $x = y + (-b)$. Note that $y + (-b)$ is a point in $A \oplus (-B)$. Thus, $x \in A \oplus (-B)$.
($\Leftarrow$) If $x \in A \oplus (-B)$, then we have $x = a + (-b)$ for $a \in A$ and $b \in B$. Rewrite $x = a + (-b)$ as $a = x + b$. Therefore, a belongs to both A and B^x which implies $A \bigcap B^x \neq \emptyset$ ■

Corollary 4.5 $A^s \bigcap B^t \neq \emptyset$ *if and only if* $(t - s) \in A \oplus (-B)$.

Proof: A^s and B^t intersect if and only if they intersect after they are both translated by $(-s)$, *i.e.* if and only if $A \bigcap B^{t-s} \neq \emptyset$. ■

From Corollary 4.5, we immediately obtain a useful fact which states that A and B intersect if and only $A \oplus (-B)$ contains the origin.

In the marker making applications, there is a local coordinate frame attached to each of the polygons. The coordinates of the vertices of the polygon are given in the local coordinate system. The location of a polygon in the global coordinate system is given by the global coordinates of the polygon's local origin. Let $P(s)$ denote the fact that the local origin of polygon P is placed at point s in the global coordinate system. Without confusion, when location for P is not specified, we will assume P is placed at the global origin. The geometric interpretation of Corollary 4.5 for two polygons P and Q is: $P(s)$ and $Q(t)$ intersect if and only if $t - s$ is in the Minkowski sum polygon $P \oplus (-Q)$. Since Q's local origin coincides with the global origin, $(-Q)$ is simply the polygon Q rotated 180 degrees around its location origin.

Figure 6 shows that as long as the local origin of polygon Q is outside the Minkowski Sum of P and $(-Q)$, Q cannot overlap P.

4.3 Algorithms for Computing Minkowski Sums

Previously proposed algorithms for computing Minkowski sums have been limited to convex polygons and simple polygons (a polygon is a simple polygon if it is non self-intersecting and without holes). Guibas *et. al.* [9] observed that the Minkowski sum of two convex polygons can be computed in linear time by merging the edge segments of the two polygons. In general, it is easy to show that an edge segment on the boundary of the Minkowski sum polygon of P and Q is part of an edge segment formed as the sum of a vertex in P and an edge in Q or *vice versa*. Let us call the edges formed by the sum of a vertex in one polygon and an edge of the other polygon candidate edges. If there are n vertices in P and m vertices in Q, then there are $O(mn)$ candidate edges. A natural idea for generating the Minkowski Sum is to calculate the arrangement [8] of the candidate edges in $O(m^2n^2 \log nm)$ time. The algorithms in [13] and [1] for calculating the Minkowski sum of two simple polygons followed this idea. Kaul *et. al* [13] introduced the concept of vertex edge supporting pairs which reduces the number of candidate edges. In the worst case, the Minkowski sum of two simple polygon can have $O(m^2n^2)$ edges and the same number of holes.

There is a class of polygons called *starshaped* polygons which are not as restricted as convex polygons but also easier to compute than simple polygons. A polygon is a starshaped polygon if there exists a point such that the line segment connecting this point and an arbitrary point in the polygon is completely contained in the polygon. The point is called a *kernel* point of the polygon. We now show how to compute the Minkowski sum of starshaped polygons. First, we prove a crucial property for the Minkowski Sum of two starshaped polygons that greatly simplifies the computing of the Minkowski sum.

Theorem 4.6 *The Minkowski sum of two starshaped polygons is also a starshaped polygon.*

Proof: Let P and Q be two starshaped polygons. Let k_1 be a kernel point of P and k_2 be a kernel point of Q. It suffices to show the $k_0 = k_1 + k_2$ is a kernel of $P \oplus Q$. To see this, let u be an arbitrary point in $P \oplus Q$ and by definition $u = v + w$ for $v \in P$ and $w \in Q$. Since P and Q are starshaped, we have that $k_1 v$ is totally contained in P and $k_2 w$ is totally contained in Q. Therefore, the Minkowski sum $k_1 v \oplus k_2 w$ is totally contained in $P \oplus Q$. $k_1 v \oplus k_2 w$ is a parallelogram with k_0 at one end of a diagonal and $u = v + w$ at the other diagonal. Hence, $k_0 u$ is totally contained in $P \oplus Q$. ■

Remark: The theorem shows that starshaped polygons are "closed" under Minkowski sum operations. The only previously known class of polygons that is closed under Minkowski sum is convex polygons.

It follows from Theorem 4.6 that the Minkowski sum of two starshaped polygons can not have holes. Thus, the computation of Minkowski sum is reduced to calculating the outer envelope [8] of the arrangement of the $O(mn)$ candidates by an angular sweepline algorithm. The outer envelope of $O(mn)$ segments can have $O(mn\alpha(mn))$ [10] segments where $\alpha()$ is the extremely slowly growing inverse of the Ackermann's function. For practical purposes, it can be considered a constant. The straightforward implementation of the angular sweepline algorithm runs in $O(mn\alpha(mn) \log mn)$ time. Hershberger [11] presented an algorithm for calculating the outer envelope of n line segments in $O(n \log n)$ time. Therefore, we have

Theorem 4.7 *The Minkowski sum of two starshaped polygons can be computed in $O(mn \log mn)$ time.*

Currently we are using a numerically robust implementation of angular sweepline algorithm for computing Minkowski sum for starshaped polygon as inputs. The algorithm is based on the observation that if we are unsure if a point q lies on the Minkowski sum, we can always project from the kernel point p through q and take the farthest intersection with a candidate edge. We have encountered data containing a few non-starshaped polygons but our studies have shown that all these polygons can be expressed as a union of two and very rarely three starshaped polygons. For those non-starshaped polygons, we have a decomposition algorithm to decompose the polygon into a small number of starshaped ones.

5 Compaction Using a Position-Based Optimization Model

In this section, we describe a compaction algorithm that directly solves for the positions of the polygons without the use of time-intervals or simulation. Instead, this algorithm uses a *position-based* optimization model. The model contains artificial constraints that restrict the solution space to a convex feasible region. These artificial constraints are generated using a *locality heuristic* which in turn relies on Minkowski sums of polygonal regions (boundary plus interior of each polygon) to create an explicit representation of the set of non-overlapping positions of each pair of polygons. The objective in the model is a linear function of the positions of the polygons. The algorithm solves this model using linear programming, generating a new set of positions for the polygons. Since artificial constraints were added, the new set of positions may not be a local minimum of the objective for the original compaction problem. The algorithm computes a new model and repeats, terminating when two consecutive models are identical. In practice, very few iterations are required to find a local minimum for the original compaction problem. This improved algorithm runs two orders of magnitude faster than the previous velocity-based algorithm.

5.1 A Locality Heuristic

We represent each polygon in a local coordinate system. The position of polygon P is given by the global coordinate c_P of its local origin. We assume that the local origin of each polygon is a kernel point of that polygon. (If any polygon is not starshaped, then we decompose it into starshaped polygons and add the constraint that these polygons must move as one unit.) Recall that $P(c_P)$ is a copy of P with its local origin positioned at c_P. From the previous section, we have that polygon $Q(c_Q)$ intersects polygon $P(c_P)$ iff $c_Q - c_P \in P \oplus (-Q)$. The purpose of the *locality heuristic* is to find a large convex region in the exterior of the Minkowski sum $P \oplus (-Q)$ which contains the point $c_Q - c_P$.

The *locality heuristic* finds such a convex region by first determining a point on the boundary of the Minkowski sum that is "nearest" to $c_Q - c_P$ (described below). Starting from that point, it walks clockwise and counterclockwise along the boundary of the Minkowski sum. When walking clockwise (with respect to the origin of the Minkowski sum), it always makes left turns. It follows the next edge if it turns to the left (at a concave vertex of the Minkowski sum), otherwise, it extends the current edge, finds its intersection with the Minkowski sum and resumes the walk from that point. This procedure continues until the current edge can be extended to infinity. It proceeds analogously in the counterclockwise direction, making right turns instead of left turns (see Figure 7). The heuristic outputs a set of constraints consisting of the half-planes to the "outside" of each Minkowski edge it encounters. The intersection of these half-planes is a convex subset of the feasible solution space.

Under the locality heuristic, the "nearest" point is not the boundary point with the closest Euclidean distance to $c_Q - c_P$. Instead, it is the intersection of the segment from the origin to point $c_Q - c_P$ with the boundary of the Minkowski sum. If $c_Q - c_P$ is inside the Minkowski sum, as it would when we are separating overlapping polygons, the nearest point is obtained by extending the ray from the origin through the point $c_Q - c_P$ until it intersects the boundary. This choice of nearest point is important for the correct operation of the separation algorithm described in Section 6.

5.2 The Compaction Algorithm

As in the algorithm in Section 3, we assign a set of desired directions f_i to each polygon; however, instead of assigning velocities, we use a set p_i of variables representing the positions of the polygons. The non-

overlapping constraints of the position-based model are built in the following way. We first use a sweepline algorithm to find all the adjacent pairs of polygons. For each adjacent pair, we find the the set of half-plane constraints generated by the locality heuristic in the previous section.

The optimization model is set up with all the constraints from each adjacent pair and the objective function

$$\text{maximize} \quad \sum_{i=1}^{n} f_i \cdot p_i$$

This model is a linear program, and its solution is a set of positions which represent maximal motion in the desired directions that satisfies the current set of constraints. After the motions are applied, the system reaches a minimum with respect to the set of constraints generated by the heuristic. The algorithm may need to iterate because when the polygons are moved into their new positions, some of the convex regions generated by the locality heuristic might change. The algorithm stops iterating when the set of convex regions calculated by the heuristic is the same as the previous iteration, at which point the local minimum for the compaction problem is reached. The final position is a local minimum because the convex regions contain a feasible open ball about the current set of positions. Furthermore, each step of the algorithm moves the polygons from one point to another in the same convex subset of the non-convex feasible (non-overlapping) solution space. Therefore, the straight line motion for each step stays within the solution space, and the total motion is a piecewise linear subset of the solution space (actually, manufacturers would not mind if the polygons "leap-frogged" over each other on the way to a more compact layout. We have recently developed a generalization of our compaction algorithm based on mixed integer programming which does exactly that.)

6 Separation of Overlapping Polygons

In this section, we consider the problem of separating overlapping polygons. Given a set of overlapping polygons, the problem is to find a set of translations of the polygons such that, after the translations, the polygons are in non-overlapping positions and such that the total translation of all the polygons is minimized.

First we show that finding a global minimum for the total translation is NP-complete. Next we show how to modify the compaction algorithm of the previous section to find a local minimum of the separation problem.

6.1 Lower Bounds

Theorem 6.1 *The separation of overlapping polygons is NP-complete even for rectangles.*

Proof: Since rotation is not allowed, we can use the same argument as in [20] to show that the problem is in NP. To show it is NP-hard we again reduce PARTITION to the problem. Figure 8 shows the construction of the reduction. Let $(a_1, a_2, \ldots, a_n)$ be the integers in an instance of PARTITION. Let $B = \sum_{i=1}^{n} a_i/2$. Place n rectangular pieces of height 1 and length a_i $(i = 1, \ldots, n)$ inside a rectangular space of size $2 \times B$ in overlapping positions. The rectangular space is then surrounded by many additional rectangular "blocks" to make sure its size is not easily enlarged. For one piece to move from one position to another position inside the rectangular space, it needs to move no more than 1 unit vertically and B units horizontally. The total motion of the pieces is less than $B + B^2$. The surrounding blocks are built in such a way that to increase the length or the width of the rectangular space, they have to move a total of $B + B^2$ units distance. Therefore, PARTITION has a solution if and only if the total displacement in the separation is less than $B + B^2$. The construction can be done in polynomial time. ■

For the cases we are dealing with, the polygons just slightly overlap. We can show that separating slightly overlapping polygons is no easier. For simplicity, we define the degree of overlap r for two overlapping polygons P and Q as:

$$r = \max\left(\frac{\text{area}(P \cap Q)}{\text{area}(P)}, \frac{\text{area}(P \cap Q)}{\text{area}(Q)}\right)$$

And we say the two polygons are slightly overlapping if $r < 1/c$ for some constant integer $c > 1$.

Theorem 6.2 *The separation of overlapping polygons is NP-complete even if the polygons just slightly overlap each other.*

Proof: We will reduce the NP-hard 2-PARTITION problem to this problem. The 2-PARTITION problem is defined as follows: given nonnegative integers $n, a_1, \ldots, a_n$ with $\sum_{j=1}^{n} a_j = 2B$, can these integers be partitioned into two subsets such that each subset contains $n/2$ integers and the sums of the two subsets are equal. We use a construction similar to that in the previous proof. Each piece still has height 1 but width $2Bc + a_i$. The rectangular space in the middle now has size $2 \times (2Bnc + B)$. We divide the n pieces into two rows each containing $n/2$ pieces within the middle space. The two rows do not overlap vertically. Within each row, the pieces are spread out evenly. The degree of overlap between two adjacent pieces in each row is at most

$$\frac{2B}{2Bc + a_i} \leq 1/c$$

because $2B$ is the total overlap. If there is a solution to 2-PARTITION, the moves needed to make the pieces non-overlapping are as follows: exchange two vertically adjacent pieces when necessary and do horizontal adjustment within each row. The vertical exchanges have a total displacement of at most $2n$. The total horizontal displacement within each row is at most nB because each piece has at most B horizontal displacement. So with $< 2n(B+1)$ total displacement, we can solve the separation problem. We will put $2n(B+1)$ surrounding blocks on each side to restrict the movement of the pieces within the middle space. ■

It is easy to see that the constant c can be replaced by a polynomial $F(B, n)$ and the proof still works.

6.2 The Separation Algorithm

This section describes a modification of the compaction algorithm that finds a local minimum for the separation problem using a similar position-based optimization model.

For two overlapping polygons P and Q, if we displace P by d_P and Q by d_Q, then the two polygons will not overlap in their new positions iff $c_Q - c_P + d_Q - d_P$ is outside of the Minkowski sum $P \oplus (-Q)$. The vector from $c_Q - c_P$ to the closest boundary point on the Minkowski sum yields the shortest vector $d_Q - d_P$ that separates the two polygons. Suppose we constrain $c_Q - c_P + d_Q - d_P$ to remain within a convex subset of the exterior of $P \oplus (-Q)$ that touches the closest boundary point. If we so constrain every pair of overlapping polygons, then if a feasible solution with respect to the constraints exists, the solution will give displacements that separate the polygons.

Remark: we use the "nearest" point as defined by the locality heuristic instead of the Euclidean closest point. This choice guarantees a feasible solution if the following three conditions hold:

- the polygons are not restricted to stay within a bounding box;
- all polygons are starshaped;
- no two polygons have their local origins at the same global position.

To see this is the case, consider shrinking all the polygons towards their local origins the same amount until they are not overlapping. Then scale up the whole layout the amount that the polygons were shrunk. The result is a non-overlapping layout. However, this feasibility proof depends on the fact that each "nearest" point stays at the same spot on each shrinking polygon. This holds for the given definition of "nearest" point. It does not hold for the Euclidean closest point. Naturally, there usually is a bounding box, but at least the other two conditions hold for the layouts we consider.

As shown in Figure 9, if two polygons are slightly overlapping, then the difference between the polygon positions will be slightly inside the Minkowski sum. The convex region found by the locality heuristic still gives a good indication of the direction of motions to separate the two polygons. Thus, we can use the locality heuristic from the last section to find the convex regions. The constraints are built similarly as in the compaction algorithm of the previous section. The objective function is modified to:

$$\text{minimize} \quad \sum_{i=1}^{n} |q_i|$$

where $|q_i|$ is the displacement of polygon i. We can use a standard technique in linear programming to eliminate the absolute values. If the set of constraints of this linear program is feasible, then the solution gives the locally minimal displacements that will separate the polygons.

7 Layout Made Easy

The separation algorithm of the previous section solves an outstanding problem in database driven systems for automated marker making. Such systems, developed by several CAD/CAM firms, are based on a database of high quality human generated markers.

Some human marker makers with 20 to 30 years of experience can generate extremely high quality markers. The markers generated by marker makers with only two or three years of experience are often 1% lower in efficiency than the markers made by the most experienced marker makers. Commercially available systems automatically generate markers at least 5% below the efficiency of the best human generated markers. It is thought that if these systems could somehow start with the top quality human generated markers as an a initial configuration, then they could generate much more efficient markers. This is the idea behind a database marker making system.

Given a set of polygons to be placed, the typical database matching system applies a set of shape similarity criteria to find the marker in the database that has the same size and shape combinations. Each polygon to be placed in the new marker is matched to a polygon in the human generated marker. Once such a match is found, it uses a "one polygon at a time" technique to place each polygon in the new marker at the position of its matching polygon in the human marker. If the polygon overlaps previously placed polygons in the new marker, its position must be "corrected" so that it does not overlap these polygons. Once a polygon is correctly placed, its position is frozen. Without a coordinated overlap correction method, this form of "one polygon at a time" correction can grossly alter the layout of the marker and make it harder or even impossible to place the rest of the polygons anywhere near the position of their matching pieces. In some cases the correction algorithm fails to find a non-overlapping position for the polygon.

With our separation algorithm, we can give a simpler and more reliable layout algorithm. First, lay out each polygon at the corresponding position of its matching polygon in the human generated layout. This layout is created regardless of whether the polygons overlap each other. Next, apply the separation algorithm to find a nearby feasible non-overlapping placement if one exists. If no such placement exists, we can increase the length of marker to allow such a placement and apply the leftward compaction algorithm to shorten the length.

If we have freedom in both dimensions of the marker, we can always find a nearby non-overlapping placement (see the remark in Section 6.2). However, if the width is fixed, our separation algorithm can fail to find a feasible placement. However, this will only happen when gross alterations are required to create a valid marker. We can not expect to do better because the general problem is NP-complete.

8 Performance

We have combined our algorithms into one separation/compaction algorithm with multiple user-selected options. This section describes the performance of our algorithm and its application to database driven automatic marker making.

8.1 Running Time and Robustness

Because the neighbor-relation is a planar graph, a linear number of neighboring polygon pairs is determined by a sweepline algorithm. The convex region found by the locality heuristic usually contains a small number of edges. Thus in practice, the total number of constraints is linear in the input size. The algorithm typically runs in two to five iterations before it stops. Leftward compaction for a marker of 120 polygons, with the largest polygons having nearly 100 vertices, runs in 20 seconds on a 28 mips Sun SparcStation.

The algorithm is also numerically very robust. It handles degenerate cases, slightly overlapped inputs and (slightly) inaccurate Minkowski sums quite well. In particular, the vertices of the Minkowski sums we use have been rounded to the nearest integer lattice point.

8.2 Improvement in Cloth Utilization

The algorithm has improved the cloth utilization of many production quality human generated markers (markers that actually go to the cutting room). The average improvement for the markers from one of the worlds largest jeans manufacturers is 0.32%; the average savings in material is 1.36 inch per marker. As Section 1.2 indicated, a 0.1% improvement in efficiency would save about two million dollars in material for the manufacturer. In the leftward compaction example shown in Figure 1, the efficiency of the marker has increased by nearly 1% and total length shortened by 5 inches.

8.3 Database Driven Marker Making

Using a naive polygon matching algorithm, we match a new set of polygons to the set of polygons in a human generated marker. Figure 10 shows the marker generated by substituting matching polygons. Using our separation/compaction algorithm, we eliminate the overlaps in the marker and then compact leftward. The resulting marker has an efficiency of 88.89% which is comparable to human marker makers with two or three years of experience. By moving three small polygons manually to the gap on the lower right corner of the marker and running leftward compaction again, the efficiency increases to 89.49%, only 0.65% lower than the marker generated by an expert human (Figure 13) for the same set of polygons. This shows that starting from a good initial configuration can greatly reduce the complexity of marker making and demonstrates the applicability of our separation algorithm for database-driven marker making.

8.4 Extensions

In recent work, we have extended our position-based model to allow compaction with small rotations of the polygons [16]. In fact, we have two different methods. One extension involves relaxation of individual polygons: finding an orientation of each that gives it the largest free space. In another extension, we consider the exact non-linear optimization model for the entire set of polygons involving both translations and rotations. Instead of solving this model directly, we solve a linearized version. Ordinary translation-only separation is used to eliminate small overlaps resulting from the linearization.

We have also created a mixed integer programming version of position-based optimization that can be used to find optimal layouts of small numbers of polygons [6]. Using the MINTO optimization package, we have sometimes been able to create optimal layouts of up to nine polygons in a few minutes.

9 Concluding Remarks

We have presented here the first fast and practical algorithm for separating and compacting a layout of convex and non-convex polygons. The algorithm has been shown to be every effective in a real world application: marker making in apparel manufacturing. The algorithm is general and versatile. We can use the algorithm to perform different tasks in all stages of marker making, such as compaction, opening up gaps, and database oriented automatic marker making, by simply tailoring the objective functions. The separation/compaction algorithm uses a new position-based model that we have recently been able to extend to compaction with rotation and optimal layout.

We have seen several researchers in the area of cutting and packing introduce the same concept using many different names: configuration space, Minkowski sum, hodeograph, non-overlap space, and so forth. We hope our work will help others realize that these are all the same concept–the Minkowski sum–and that the field of computational geometry offers both theoretical and practical results on the Minkowski sum that they can draw on. It is hoped that by demonstrating their great utility as we have done here, interest might be generated in the use of these and other algorithms from the field of computational geometry.

Acknowledgements

We would like to thank Prof. Fred Abernathy, Karen Daniels and Dr. Richard Szeliski for helpful discussions. Members of our research team, especially Karen Daniels, Lee Wexler and Venkatesh Reddy, have facilitated the implementation of this work by building and maintaining Harvard Automated Marker Generation Testbed. Murray Daniels of MITRE Corporation provided the user interface software. Karen Daniels has carefully read the earlier drafts of this paper and offered many suggestions that enhanced the presentation of this paper. Dr. Veljko Milenkovic has kindly proofread the drafts of this paper and point out several typographical and grammatical errors.

References

[1] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, pages 72–82, 1992.

[2] R. Anderson, S. Kahan, and M. Schlag. An $O(n \log n)$ algorithm for 1-D tile compaction. In M. Nagl, editor, *Graph-Theoretic Concepts in Computer Science*, volume 411 of *Lecture Notes in Computer Science*, pages 287–301. Springer-Verlag, 1990.

[3] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics (Proc. SIGGRAPH)*, 23(3):223–232, 1989.

[4] R. Barzel and A. H. Barr. A modeling system based on dynamics constraints. *Computer Graphics (Proc. SIGGRAPH)*, 22(4):179–187, 1988.

[5] J. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1987.

[6] K. Daniels, Z. Li, and V. Milenkovic. Multiple containment methods and applications in cloth manufacture. *Army Research Office and MSI Stony Brook Workshop on Computational Geometry*, October 1993.

[7] D. Dobkin, J. Hershberger, D. Kirkpatrik, and S. Suri. Implicitly searching convolutions and computing depth of collision. In *Proceedings of 2nd SIGAL*, pages 165–180, 1990.

[8] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Heidelberg, West Germany, 1987.

[9] L. Guibas, L. Ramshaw, and J. Stolfi. A Kinetic Framework for Computational Geometry. In *IEEE 24th Annual Symposium on Foundations of Computer Science*, 1983.

[10] S. Hart and M. Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6:151–177, 1986.

[11] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.

[12] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the warehouseman's problem. *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[13] A. Kaul, M.A. O'Connor, and V. Srinivasan. Computing Minkowski Sums of Regular Polygons. In *Proceedings of the Third Canadian Conference on Computational Geometry*, Vancouver, British Columbia, 1991.

[14] T. Lengauer. On the solution of inequality systems relevant to ic-layout. *Journal of Algorithms*, (5):408–421, 1984.

[15] Z. Li and V. Milenkovic. A compaction algorithm for non-convex polygons and its application. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 153–162, 1993.

[16] Z. Li and V. Milenkovic. Compaction of a 2D layout of non-convex shapes and applications. *SIAM Conference on Geometric Design*, November 1993.

[17] Z. Li and V. Milenkovic. On the complexity of the compaction problem. In *Proc. 5th Canadian Conference on Computational Geometry*, pages 153–162, 1993.

[18] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22:560–570, 1979.

[19] D. Maple. A hierarchy preserving hierarchical compaction. In *Proc. 27th Design Automation Conference*, pages 375–381, 1990.

[20] V. Milenkovic, K. Daniels, and Z. Li. Automatic Marker Making. In *Proc. 3rd Canadian Conference on Computational Geometry*, 1991.

[21] V. Milenkovic, K. Daniels, and Z. Li. Placement and Compaction of Nonconvex Polygons for Clothing Manufacture. In *Proc. 4th Canadian Conference on Computational Geometry*, 1992.

[22] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proc. SIGGRAPH)*, 22(4):289–298, 1988.

[23] R. C. Mosteller, A. H. Frey, and R. Suaya. 2-d compaction a monte carlo method. In P. Lasleber, editor, *Advanced Research in VLSI*, pages 173–197. MIT Press, 1987.

[24] R. S. Palmer. *Computational Complexity of Motion and Stability of Polygons.* PhD thesis, Cornell University, Jan., 1987.

[25] J. C. Platt and A. H. Barr. Constraint methods for flexible models. *Computer Graphics (Proc. SIGGRAPH)*, 22(4):279–287, 1988.

[26] J. T. Schwartz and M. Sharir. Algorithmic motion planning in robotics. In J. van Leeuwen, editor, *Algorithms and Complexity*, volume A of *Handbook of Theoretical Computer Science*, pages 391–430. Elsevier, Amsterdam, 1990.

[27] J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, New York, 1982.

[28] H. Shin, A. L. Sangiovanni-Vincentelli, and C. H. Séquin. Two dimensional compaction by 'zone refining'. In *Proc. 23rd Design Automation Conference*, pages 115–122, 1986.

[29] Y. G. Stoyan. A basic problem of optimal cutting of materials for blanks of arbitrary 3D geometry space form. *SICUP Workshop on Cutting and Packing at the ORSA/TIM Joint National Meeting*, November 1992.

[30] Y. G. Stoyan. Precise and approximate methods of solution of problems in irregular allocation of nonconvex polygons in a strip of minimal length. *SICUP Workshop on Cutting and Packing at the IFORS Conference*, July 1993.

[31] P. E. Sweeney and E. R. Paternoster. Cutting and packing problems: A categorized, application-orientated research bibliography. *J. Opl Res. Soc.*, 43(7):691–706, 1992.

[32] C. K. Wong. An optimal two-dimensional compaction scheme. In P. Bertolazzi and F. Luccio, editors, *VLSI: Algorithms and Architectures*, pages 205–211. Elsevier (North-Holland), 1985.

Name: Human generated marker
Width: 59.75 in
Length: 543.16 in
Pieces: 192
Efficiency: 89.62%

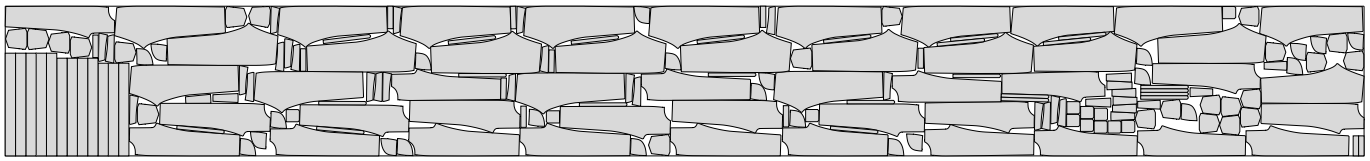

Figure 1: A human generated pants marker.

Name: Compacted marker
Width: 59.75 in
Length: 537.52 in
Pieces: 192
Efficiency: 90.56%

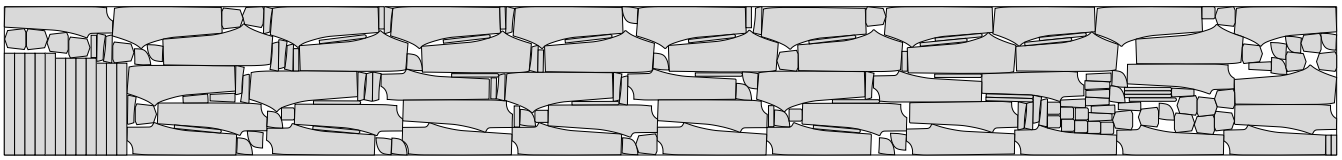

Figure 2: The human generated marker after compaction.

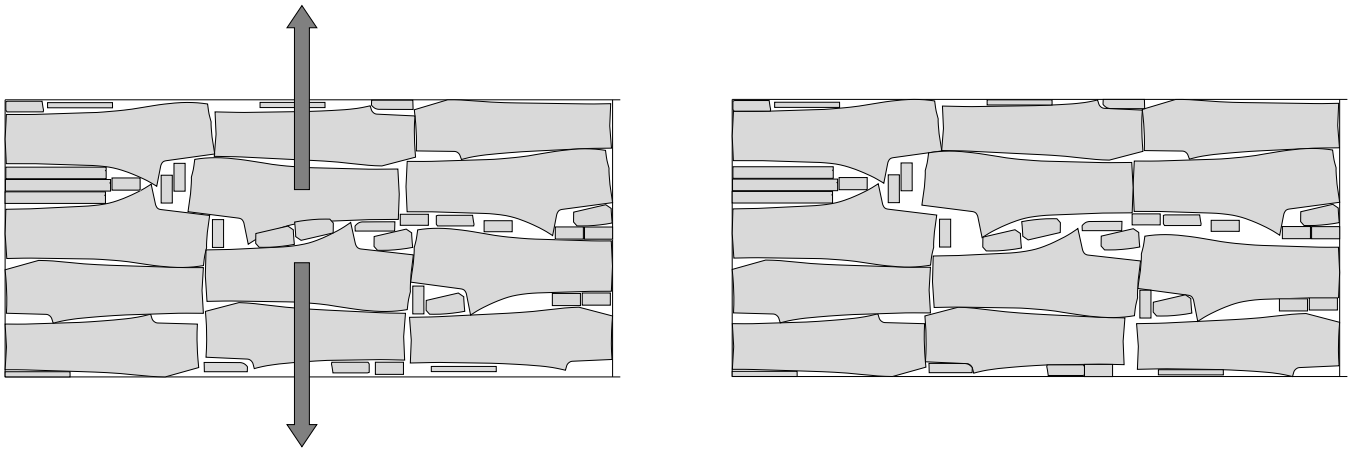

Figure 3: Opening up a gap.

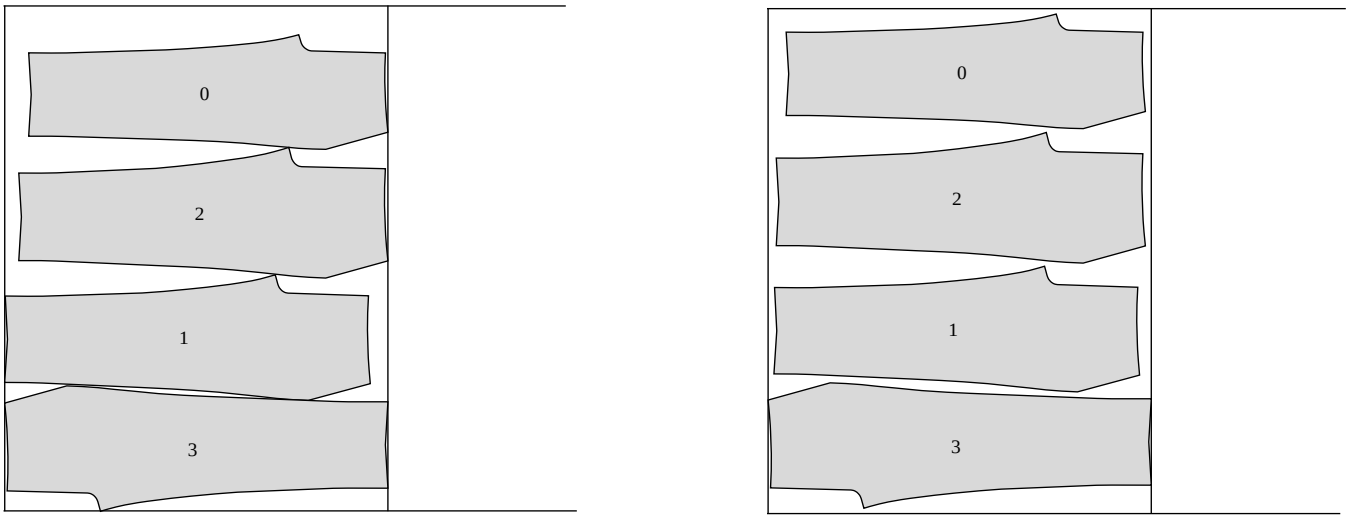

Figure 4: Example of floating.

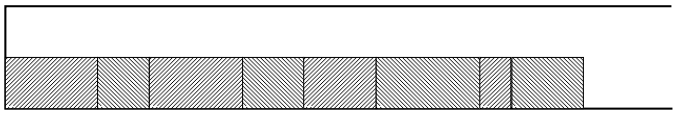

Figure 5: Reduction of PARTITION to compaction.

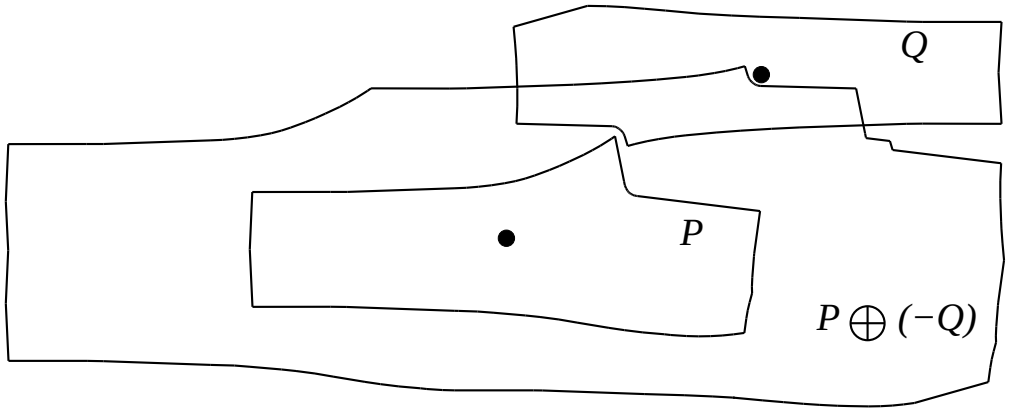

Figure 6: Minkowski sum and non-overlapping placement.

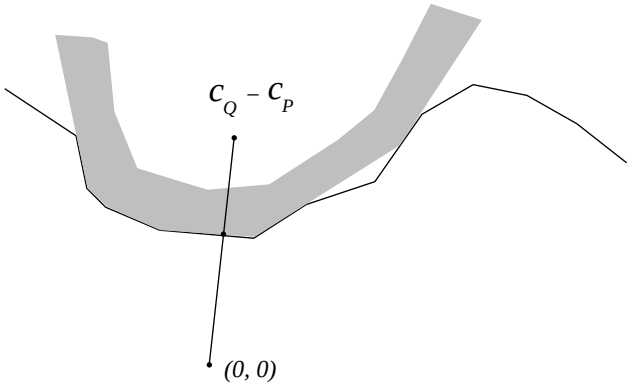

Figure 7: The "nearest" convex region in the exterior of the Minkowski sum.

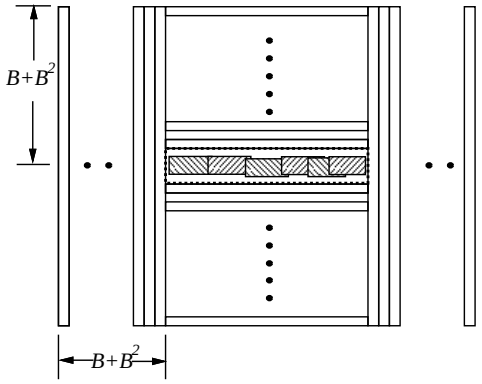

Figure 8: Reduction of PARTITION to separation of overlapping polygons.

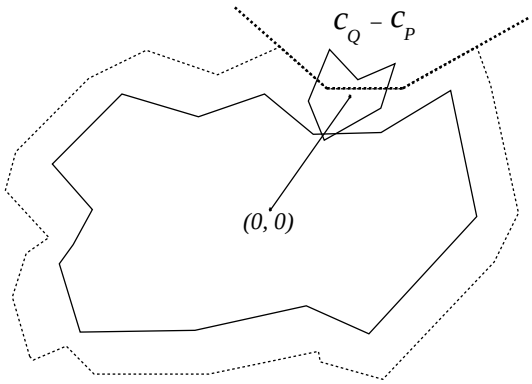

Figure 9: Minkowski sum of two slightly overlapped polygons.

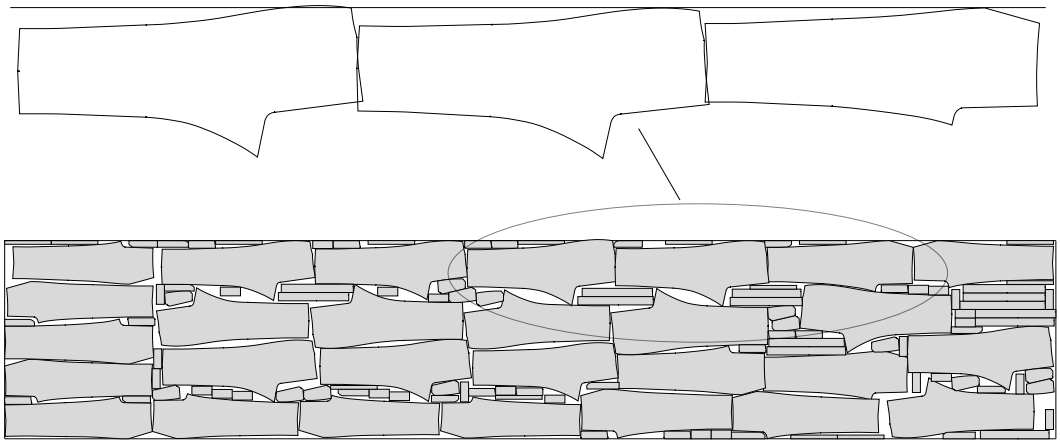

Figure 10: Marker generated by overlays.

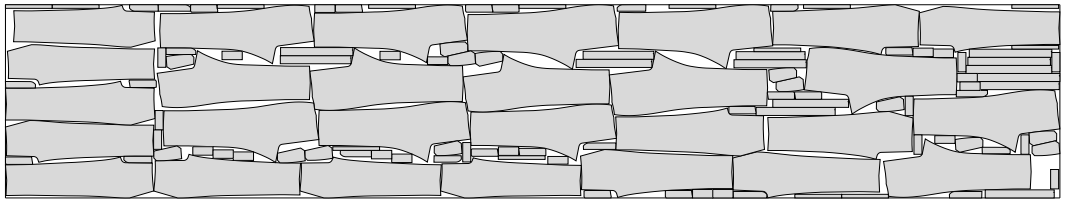

Figure 11: Marker after elimination of overlaps and leftward compaction. Length = 312.64 in, efficiency = 88.98%

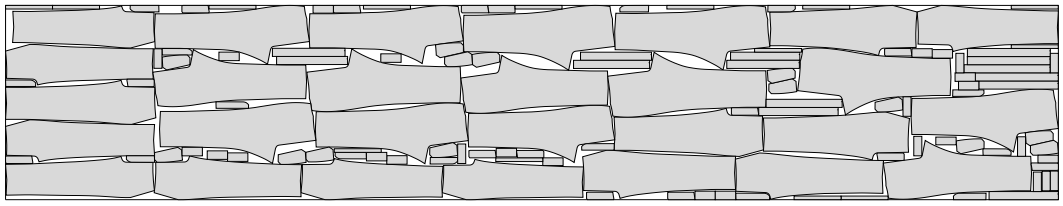

Figure 12: Marker after adjustment on 3 small polygons. Length = 310.87 in, efficiency = 89.48%

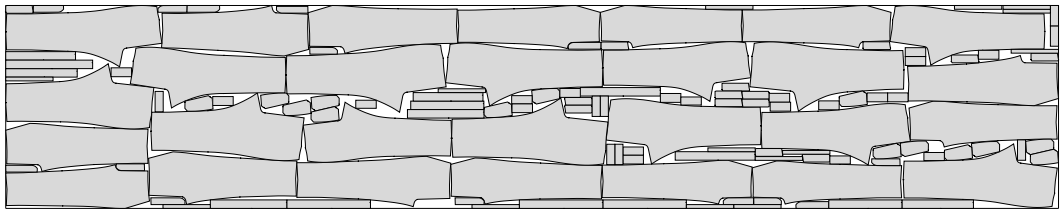

Figure 13: The corresponding marker generated by a human. Length = 308.61, efficiency = 90.14%