Placement and Compaction of Nonconvex Polygons for Clothing Manufacture *

Victor Milenkovic, Karen Daniels, Zhenyu Li
Harvard University
Center for Research in Computing Technology
Cambridge, MA 02138

1 Introduction

Marker making is a task essential to the manufacture of clothing. Figure 1 is a marker containing the parts for twelve pairs of men's pants of assorted lengths and waist measurements. This marker, or cutting plan, was created by a human in about 45 minutes. The large panels are the front and back pieces of the pants. The smaller pieces are *trim*: waist-bands, pocket-facings, belt loops, etc. The pieces are cut out of up to sixty layers of cloth on a long cutting bed by a reciprocating blade. The *efficiency* of a marker is the percentage of area that is utilized: this particular marker has 90.56% efficiency. Current software for automatically generating markers falls short of human performance by five to ten percent. A single percent of efficiency pays for the labor involved in creating the marker, and therefore the software is not used in large scale manufacturing (although it might be used for small runs).

We are currently engaged in the second year of a three year project in automatic marker making. The goal of this project is to match the efficiency of human-generated pants markers at least 80% of the time. As we reported last year [6], finding the optimal marker is NP-hard. However, we only have to match what a human can generate routinely in 45 minutes. Published research in automatic marker making is scarce [10] [7] [5].

1.1 Problem Description

The marker width[1] represents the width of the bolt of cloth from which the pants will be cut. At present, there is no standard width for a bolt of cloth. The demand for different sizes and styles also varies in an unpredictable fashion. The pieces are polygons, and the goal is to place the pieces in a non-overlapping configuration that minimizes the length of the marker. Our previous report [6] describes certain other placement restrictions that we do not consider here. Pieces can be flipped about the x or y axis and they can be rotated 180 degrees. In some cases, the cloth has a direction or "nap", and only a flip about the x axis is allowed. Finally, small rotations of up to 3 degrees are sometimes permitted.

From our interviews and observations of people who create markers, we have determined that they place the large panels first. The large panel placement determines the length of the final marker and hence its efficiency. Very rarely will a worker increase the length of the marker to fit the trim pieces in. In fact, an experienced worker will not move the panels at all once he or she starts placing the trim. Instead, the worker will anticipate the needs of the trim during the panel placement phase: guided by experience, he or she will shift or tilt the panels here and there in a way that opens useful gaps between them.

*This research was funded by the Textile/Clothing Technology Corporation from funds awarded to them by the Alfred P. Sloan Foundation.

[1] As commonly depicted, markers have their long dimension, the length, in the x direction, and they have their short dimension, the width, in the y direction.

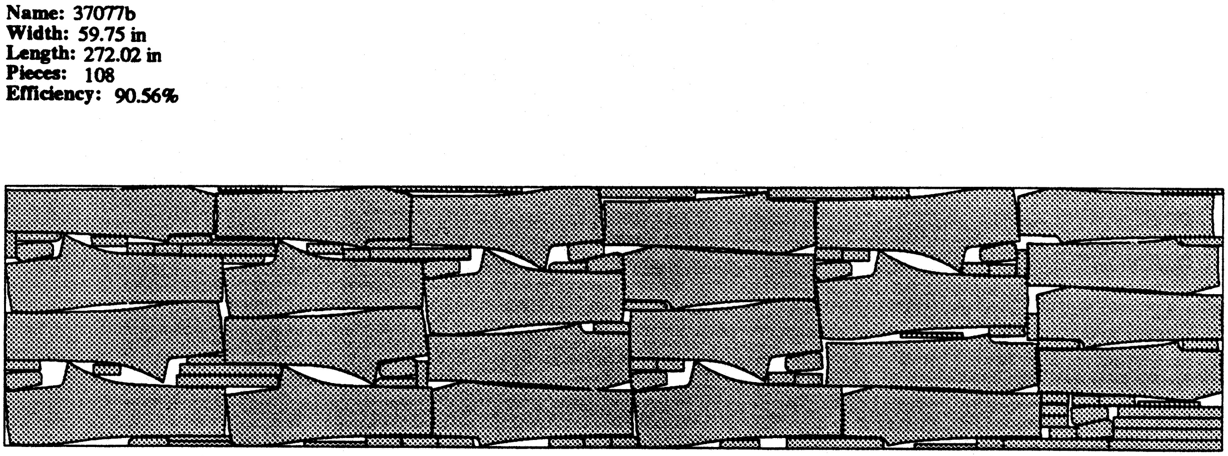

Figure 1: Typical Pants Marker

1.2 Outline

We have partitioned the marker-making task into three parts: panel placement, compaction, and trim placement. We can currently generate panel placements for the most common *grid-like* configuration: columns of four with corresponding panels in adjacent columns abutting. Our panel placement algorithms do not anticipate the needs of trim placement. Instead, we intend to rely on a local optimization technique we call *compaction* which acts on the pieces as if they are frictionless solids. Various types of forces can be applied to the pieces: a constant leftward "gravity" field to compact the overall marker or a repulsion force between neighboring pieces to open up a gap between them. We are currently working on trim placement algorithms. These will utilize compaction to open up gaps where trim pieces almost fit. This is a computer-oriented way of solving the problem that humans solve by experience.

2 The Minkowski Sum

The Minkowski sum [3] [2] [8] [9] [1] [4] is an important part of the preprocessing necessary for fast panel placement and compaction. Given two planar point sets A and B, the Minkowski sum and difference are defined as follows:

$$A + B = \{a + b \mid a \in A \text{ and } b \in B\} \qquad \text{and} \qquad A - B = \{a - b \mid a \in A \text{ and } b \in B\}.$$

The Minkowski sum of two polygonal regions is a polygonal region.

Let $A + u$ and $B + v$ be copies of A and B translated by u and v, respectively. For what values of u and v do these two sets overlap? They overlap if and only if there is a point p in common: $p = a + u = b + v$, where $a \in A$ and $b \in B$. Thus,

$$v - u = a - b.$$

Thus it is easy to see that $A + u$ and $B + v$ overlap if and only if $v - u$ lies in $A - B$. When placing a piece into a marker, the Minkowski difference can be used to determine the region in which its center can be placed without causing an overlap (shown in black in Figure 2).

For the data we have, each piece is represented as a polygon with the origin at the center of its bounding box (minimum-area axis-parallel rectangle). Thus $A + u$ is simply a copy of A centered at u. Using the Minkowski difference, one can rapidly answer the question: Given the Δx between the centers of A and B, what is the minimum Δy such that B is above A? This is easily solved by finding the intersection of the vertical line $x = \Delta x$ with the boundary of $A - B$.

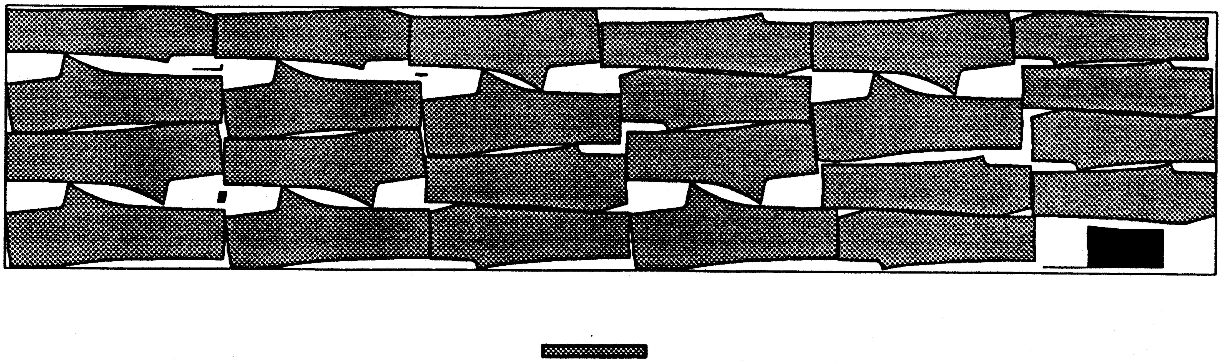

Figure 2: Regions of Valid Placement for Unplaced Piece

We are currently using a Minkowki sum algorithm which acts on general polygons. We plan to implement a faster algorithm which exploits the near-convex nature of the pieces.

3 Panel Placement

For a pants marker containing n pairs of pants (typically 9 to 14), there are $2n$ large pieces corresponding to the front and back panels of the pants.[2] Our panel placement algorithm is based on a general strategy. Partition the marker into several regions. Select a region of the marker to pack, and pack it as tightly as possible. Higher priority is given to packings that use hard-to-place pieces. As each region is considered in turn, there are fewer remaining unplaced pieces from which to select. However, the remaining pieces are easier to place. The hope is that the pieces get easier fast enough to allow the placement to be done without backtracking.

Based on our observations of human-generated markers, we make the assumption that panels appear in columns of four as depicted in Figure 3. These columns correspond to the regions in the general strategy. In the optimal placement under this assumption, each panel abuts the panels to its left and right; the panels in the first (leftmost) column abut the left margin; the panels in the last column abut the right margin. In this configuration, the minimum length of the marker is one-quarter the total length of all the panels. We call such a configuration *grid-like*. More than half of the human-generated markers we have examined have a length equal to or greater than the grid-like configuration.

3.1 The Placement Algorithm

Panel placement is performed one column at a time from left to right. The algorithm will backtrack if a column of panels cannot be placed. If there is no backtracking, the algorithm runs in $O(n^5)$ time. If there is backtracking, the running time can be exponential. To avoid backtracking, we use the strategy of placing difficult pieces first. We consider the height (Δy) of a panel to be a measure of difficulty.

As each new column is placed, the algorithm considers every possible stack of four panels with every possible orientation for each panel. There are 4! orderings of the panels within each column, and for each panel, there are four possible orientations. (For a marker with 24 panels, there are about 65 million combinations for the first column.) For a particular stack, the x-coordinate of each panel is set so that its x-interval abuts the x-interval of the panel to its left. For the first column, the panels abut the left margin. In order to be considered, a particular stack of four panels with orientations must satisfy the following conditions:

1. There must be some choice of y-coordinates for all the panels (with orientations) including the four

[2] The number of panels to be cut is $2n$ instead of $4n$ because the manufacturers exploit the symmetry of pants (the layers of cloth alternate between face-up and face-down).

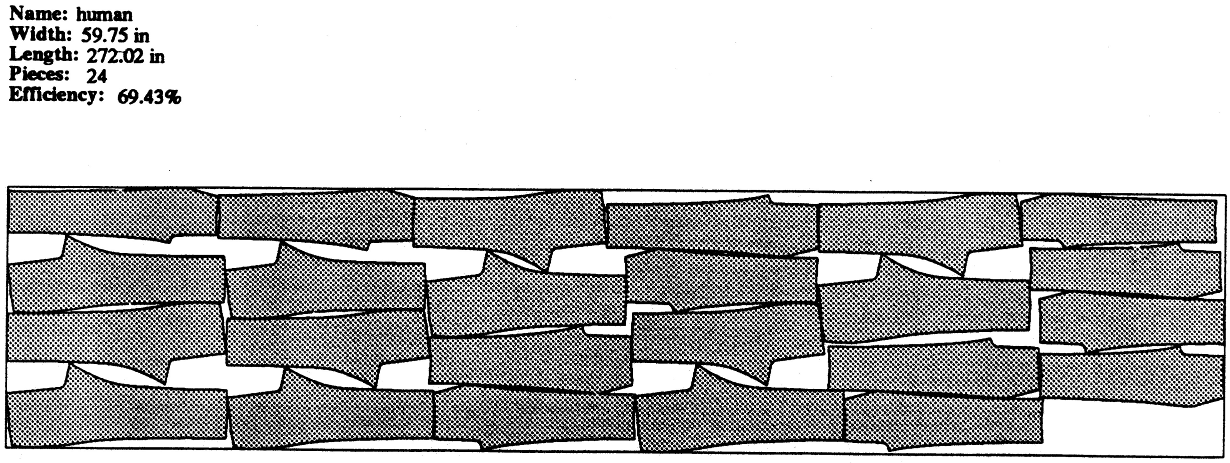

Figure 3: Panel Placement of Figure 1

tentatively placed panels such that the panels lie inside the marker and do not overlap.

2. The lengths (Δx) of the remaining panels must not be inconsistent with an even right margin.

Note that the y-coordinates for the panels in the first column may not be determined until we place the very last column of panels. On the other hand, once a panel is placed, its x-coordinate (and orientation) is fixed.

The algorithm chooses the stack which maximizes the sum of the heights of the four panels. Ties are broken by assigning a lower priority to columns with a jagged right boundary, where jaggedness is the difference between the maximum and minimum rightmost extent of the four panels.

3.2 Finding Feasible Y-Coordinates

When testing condition (1) above, the algorithm has the following information: a placement of panels into columns and the x-coordinate and orientation of each panel. It must determine if there are y-coordinates for all the panels that result in a non-overlapping placement. We observe that in a grid-like configuration each panel can interact with up to six other panels: the three pieces above and the three pieces below it in the same, preceding, and succeeding columns. In addition, a panel can interact with the top or bottom of the marker.

For each pair of interacting panels, there is a minimum Δy between them. This leads to a set of constraints of the form,

$$y_i - y_j \geq c_{ij},$$

where y_i is the y-coordinate of the center of the ith panel. The values of the constants c_{ij} are determined using the precomputed Minkowski differences. Two special variables, y_{bot} and y_{top}, represent the bottom and top of the marker. These must satisfy an additional constraint,

$$y_{\text{top}} - y_{bot} \leq w,$$

where w is the width of the marker.

When the mth column is placed, the constraints on all the panels can be condensed into a set of constraints on the y-coordinates of the panels in the mth column and y_{bot} and y_{top}. From these alone, the set of constraints on the $(m+1)$st column can be deduced, and so forth. The technique is essentially dynamic programming, and it has a running time in $O(k^2)$ per column, where k is the number of panels in a column. Since $k = 4$, a constant, the algorithm can test condition (1) for a particular stack of panels with orientations in constant time.

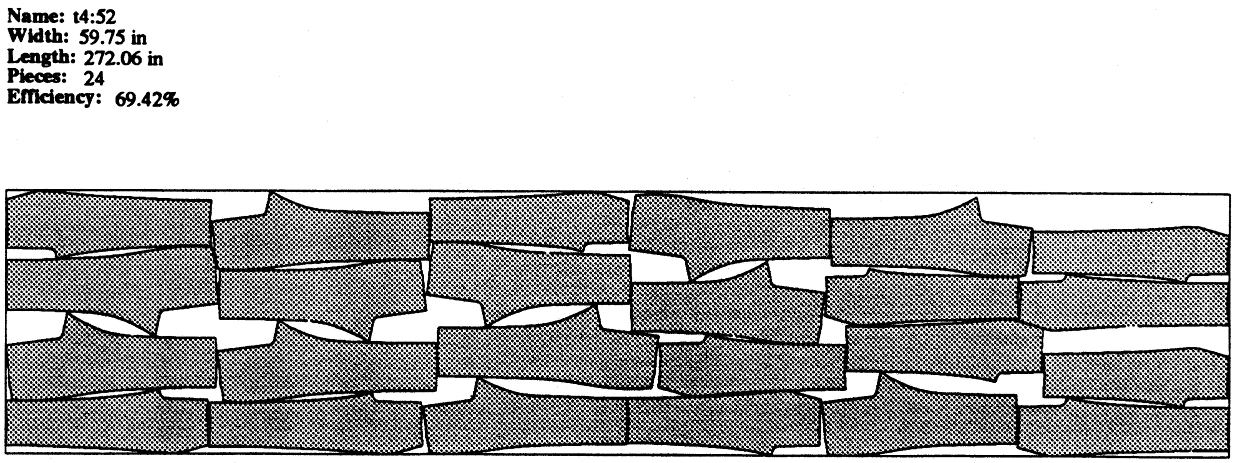

Figure 4: Automatically Generated Panel Placement

3.3 Checking for an Even Right Margin

In order to test condition (2), the algorithm must determine if the remaining panels can be placed into four rows so that the final right margin is even. This is purely a computation on the lengths of the remaining panels–there is no check to see if the putative placement will fit within the upper and lower bounds of the marker.

We choose a specific upper bound (0.25 inch) on the jaggedness of the right margin, but there are more sophisticated ways of defining evenness: for example, the permitted jaggedness could be bounded by twice the minimum jaggedness seen so far.

Unfortunately, testing condition (2) is equivalent to bin-packing, and so we must use a heuristic. Our algorithm divides the remaining panels into four sets, corresponding to the four rows of the marker, and repeatedly tests to see if swapping any two of these panels will diminish the sum of the squares of the x-coordinates of the rightmost points in the four rows.

3.4 Results

The algorithm can generate a panel placement for a 24-panel marker in slightly under five minutes on a 28 MIPS Sparc station. Figure 3 depicts a human-generated panel placement, and Figure 4 gives the output of our program on the same set of panels. The difference in length is negligible. The human marker is more orderly in the final column, but it should not be a problem to alter the placement algorithm to generate similar results.

4 Compaction

The process of *compaction* can be thought of as a physical action on the marker. It can apply leftward pressure on the right margin to shorten the marker if possible. Or it can "reach in" and open up a gap by pushing outward on pieces surrounding the gap. Our trim placement algorithm will be based on compaction: each gap will be opened as widely as possible before trim is placed inside it.

We accomplish compaction through the use of linear programming. This version allows translation but no rotation. We are currently developing a version with rotation.

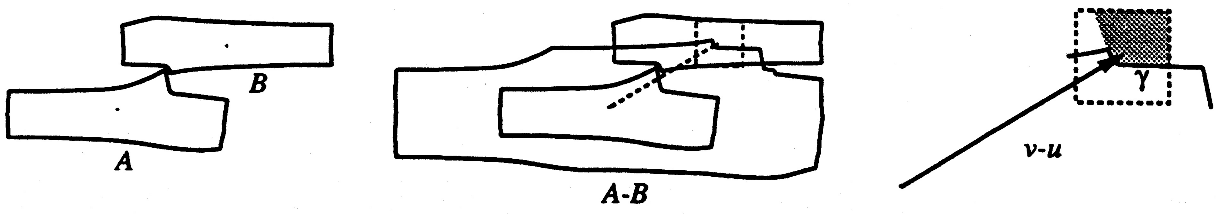

Figure 5: Pieces A and B, the Minkowski Difference (translated to the center of A), and the Convex Constraint Region for $v - u$ (Magnified 2x)

4.1 Compaction Algorithm

Each piece can be assigned a potential field vector. If piece P_i, centered at v_i, is assigned the vector f_i, then the potential energy of the system is,

$$E = -\sum f_i \cdot v_i.$$

Any translational motion of the pieces which diminishes this potential energy without introducing an overlap is desirable. The vector f_i represents the force applied to piece P_i. For example, to compress the marker from the right, a leftward force can be applied to the right margin, which can be considered to be a degenerate polygon.

Overlap is avoided by means of a pairwise constraint on the centers of neighboring pieces. Let u and v be the current translations of pieces A and B respectively. The point $v - u$ is constrained to lie outside the Minkowski difference $A - B$. This represents a non-convex constraint on the value of $v - u$, and finding the minimum potential energy for a set of non-convex constraints is NP-hard. However, selecting a convex subset of the exterior of $A - B$ in which to constrain $v - u$ results in a problem that can be solved by linear programming. A judicious choice of the convex subset assures that some progress will always be made if any progress is possible (except in certain degenerate cases).

The compaction algorithm selects the convex subset as follows. Join the center of $A - B$ to the point $v - u$ by a line segment. Let e be the edge of the boundary of $A - B$ through which this segment passes. Determine the longest convex chain γ of the boundary of $A - B$ which contains e. The point $v - u$ is constrained to lie in the convex set bounded by γ with its first and last segments extended into rays. Pants pieces have the property that this convex set is always contained in the exterior of $A - B$. The process of selecting a convex subset is illustrated in Figure 5.

The objective of the linear program is to minimize the potential energy. After finding a local minimum, the point $v - u$ may have moved so that its position corresponds to a different convex subset. In this case, the linear program must be run again. Since each step diminishes the potential energy, this process cannot loop indefinitely. In practice, only one or two steps are necessary.

4.2 Efficient Determination of Neighbors

Each pair of neighboring pieces A and B adds a set of linear constraints to the linear programming problem. The compaction algorithm determines neighbors as follows. It puts an upper bound of one inch on the amount any piece can move in the x or y direction (in a single LP stage). Under this bound, two pieces cannot interact unless their bounding boxes, expanded by one inch in each dimension, overlap. A straightforward sweep technique determines the intersections among n rectangles in $O(n \log n + l)$ time, where l is the number of intersections. In most cases, the marker is fairly tightly packed to start with, and the bound of one inch results in at most a few more iterations of the compaction procedure.

Since the pieces are larger than one inch in size, the neighbor graph is planar, and thus l is linear in n. Each center is confined to a two-inch by two-inch square, and this constraint is given to the linear programming package. The square constraints result in a confinement of each center difference $v - u$ to a four by four

Name: z2out
Width: 59.75 in
Length: 271.12 in
Pieces: 24
Efficiency: 69.66%

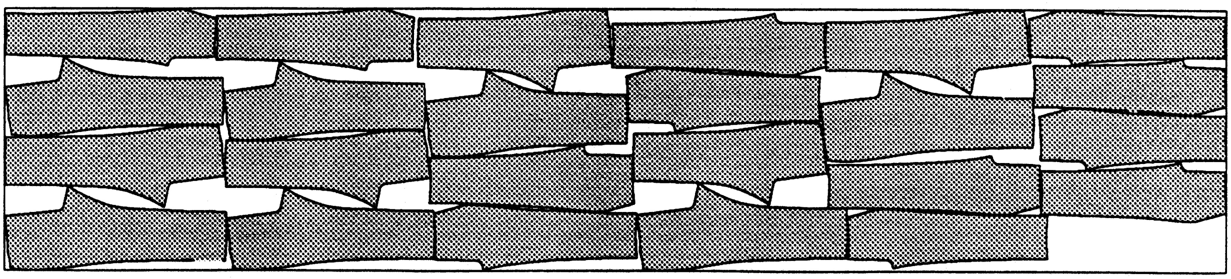

Figure 6: Leftward Compaction of Human-Generated Marker

Name: z1out
Width: 59.75 in
Length: 271.05 in
Pieces: 24
Efficiency: 69.68%

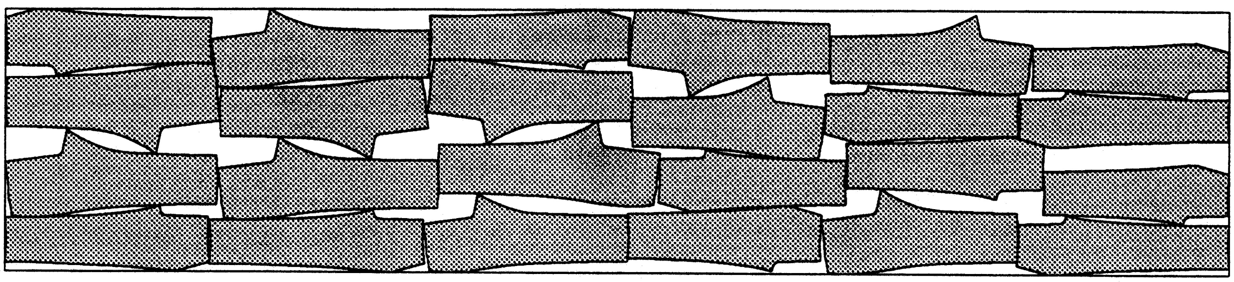

Figure 7: Leftward Compaction of Machine-Generated Marker

square. Only the portion of the chain γ which lies in this square need be passed to the linear programming package. Hence the number of constraints is quite small, essentially linear in n.

4.3 Results

Since the number of constraints is small, less than a second is required for the calls to the linear programming package. Figures 6 and 7 show the result of a leftward compaction on the panel placements in Figures 3 and 4, respectively, resulting in an improvement of about one inch or 0.2% in each case. Figure 8 demonstrates how compaction can open a gap between neighboring pieces without affecting the efficiency.

5 Conclusion

We can perform panel placement for the majority of cases in an acceptable amount of time. We can also perform compaction swiftly enough to consider using it as a subroutine to the trim placement procedure. Without the precomputation of Minkowski sums, the running times would be grossly unacceptable. At the very least, CAD/CAM software designers should perform such precomputations, and provide panel

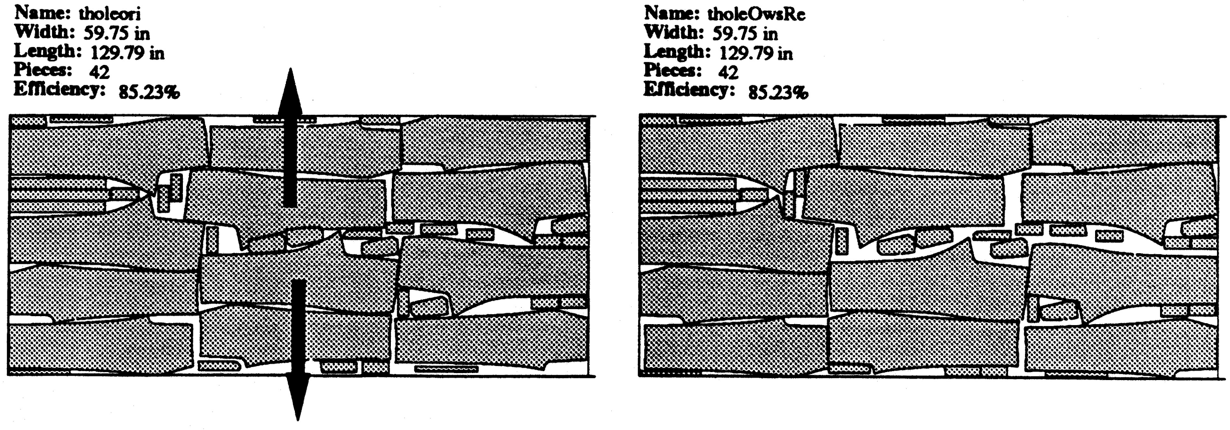

Figure 8: Gap Opened between Neighboring Panels in order to Fit Trim Pieces

placement and compaction as procedures.

We expect to be able to compute non-grid-like panel placements (Section 3) in the near future, and we hope to solve the trim placement problem using the same general strategy as we used for panel placement. Any suggestions and ideas are welcome.

References

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

[2] P.K. Ghosh. *A Computational Theoretic Framework for Shape Representation and Analysis using the Minkowski Addition and Decomposition Operators*. PhD thesis, Tata Institute of Fundamental Research, Bombay, India, 1986.

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

[4] 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.

[5] Alain Mangen and Nadine Lasudry. Search for the Intersection Polygon of any Two Polygons: Application to the Garment Industry. *Computer Graphics Forum*, 10:195–208, 1991.

[6] Victor Milenkovic, Karen Daniels, and Zhenyu Li. Automatic Marker Making. In *Proceedings of the Third Canadian Conference on Computational Geometry*, 1991.

[7] M. Nakajima and K. Hayashi. Automated Parts Layout Planning in Garment Industry By Using Group Technology. In *Computers in the World of Textiles*, pages 297–310. Textile Institute Annual World Conference, 1984.

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

[9] J. Serra, editor. *Image Analysis and Mathematical Morphology*, volume 2: Theoretical Advances. Academic Press, New York, 1988.

[10] Michikazu Tanaka and Takao Wachi. Computerized Marker Making. *Journal of The Textile Machinery Society of Japan*, 26(7):74–81, 1973.