

R.A. CUNINGHAME-GREEN

AN INDUSTRIAL CUTTING PROBLEM

Summary. An algorithm is described, for replacing two-dimensional polygonal shapes by thin shells consisting of triangles. This simplifies the computation of configuration space obstacles, an essential step in the generation of cutting layouts in an industrial context.

Keywords Cutting layouts; computational geometry.

1. CUTTING LAYOUTS

A shoe is constructed from a number of different components, many of which are irregular, non-convex 2-dimensional shapes which must be cut from sheets of material. Formerly, shoes were designed by hand using a physical model and the components were cut by hand from hides or skins. Nowadays, designs increasingly use CAD systems and the components are cut by computer-controlled machines from rolls or sheets of artificial material.

A cutting layout is a geometrical design showing how large numbers of copies of a particular component may be cut from 2-dimensional material. Thus, in Fig. 1, cutting layouts are shown for a horseshoe-shaped component known as a *vamp*. These particular layouts use the material with markedly different efficiencies: 54%, 67% and 80% respectively for the layouts of Fig. 1(a), 1(b), 1(c).

Cutting layouts are needed in order to program automatic cutting machines, but are also used to define efficiency standards for manual cutting. The layouts must be in the form of simple, repetitive patterns because the cutting is carried out by a machine which is relatively limited in its operation. A cutting head travels across the material punching out a row of copies of the shape, then travels back punching out another row. On some machines, the cutting head is able to rotate through 180° and follow a cutting layout like those of Fig. 1(b), 1(c); on others, a layout as in Fig. 1(a) is produced.

The material is usually not isotropic and the copies of the shape must therefore be cut so as to respect the grain, which may require them all to like with the same orientation as in Fig. 1(a), or may permit rotation through 180° as in Fig. 1(b), 1(c).

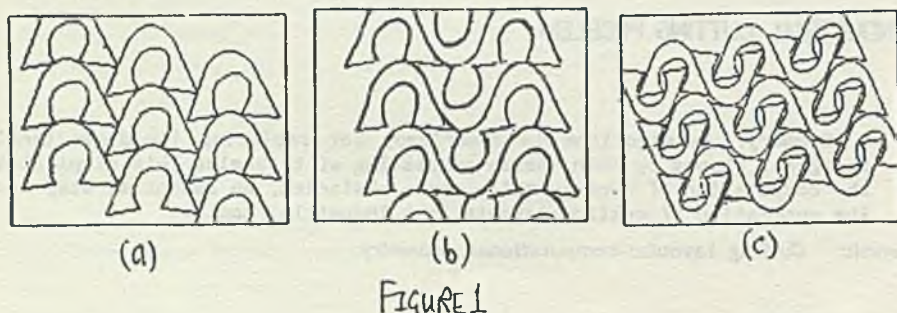


Fig. 1. Cutting layout for a horseshoe - shaped component

Rys. 1. Trasowanie cięć dla składników o kształcie podkowy

2. PROBLEM OF OVERLAP

The computer program PAX, which produced the layouts of Fig. 1, accepts a specification of a shape in the form of a suitably close polygonal approximation and generates a large number of feasible layouts, choosing one for which the efficiency of material utilization is highest. Because of the regular nature of the layouts, this process is basically straightforward. Thus in Fig. 2, the layout is generated by the two translations \underline{u} and \underline{v} , and the rate of material usage is measured by the area of the parallelogram which they span. However, the translations \underline{u} and \underline{v} must be feasible, i.e. the layout they generate must be free of overlapping shapes. That is to say each pair of shapes in the layout must have disjoint interiors. This leads us to the following problem.

Two polygonal shapes are given, each with a datum point (A and B respectively) labelled in its interior. If the shapes are placed, without change of orientation, so that the datum points fall on points A_1, B_1 of the plane respectively, will the interiors of the shapes have non-empty intersection? In other words, is the spatial relationship identified by the vector $A_1 B_1$ permissible for these two shapes?

In systematically generating feasible translation-pairs, the program PAX needs an efficient way of answering this question for arbitrary spatial relationships, without making a global investigation each time. It does this by calculating a so-called configuration space obstacle (CSO) before layout-generation begins.

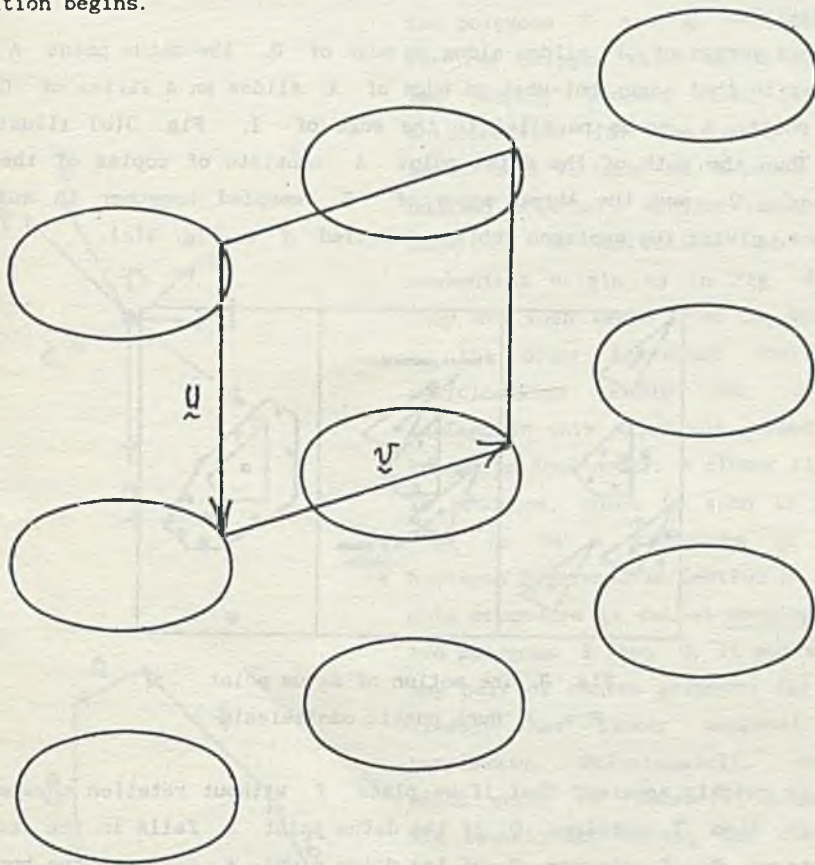


Fig. 2. Layout generated by two translations

Rys. 2. Trasowanie generowane przez 2 translacje

3. CONFIGURATION SPACE OBSTACLES

To simplify the discussion, suppose first that the shapes in question are respectively a triangle T and a quadrilateral Q , with Q initially assumed fixed with its datum point B at the origin and T free to translate without rotation.

Now take triangle T and move it without rotation to some position where it just touches Q , then "wipe" T round Q . That is to say, move T round Q without rotation, constantly remaining in contact with Q but not penetrating its interior. Consider the motion of datum point A of the triangle T .

When a vertex of T slides along an edge of Q , the datum point A moves parallel to that edge; but when an edge of T slides on a vertex of Q the datum point A moves parallel to the edge of T . Fig. 3(b) illustrates this. Thus the path of the datum point A consists of copies of the four edges of Q and the three edges of T coupled together in suitable sequence, giving the heptagon H labelled in Fig. 3(c).

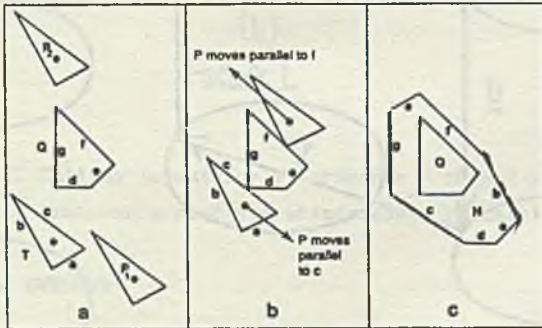
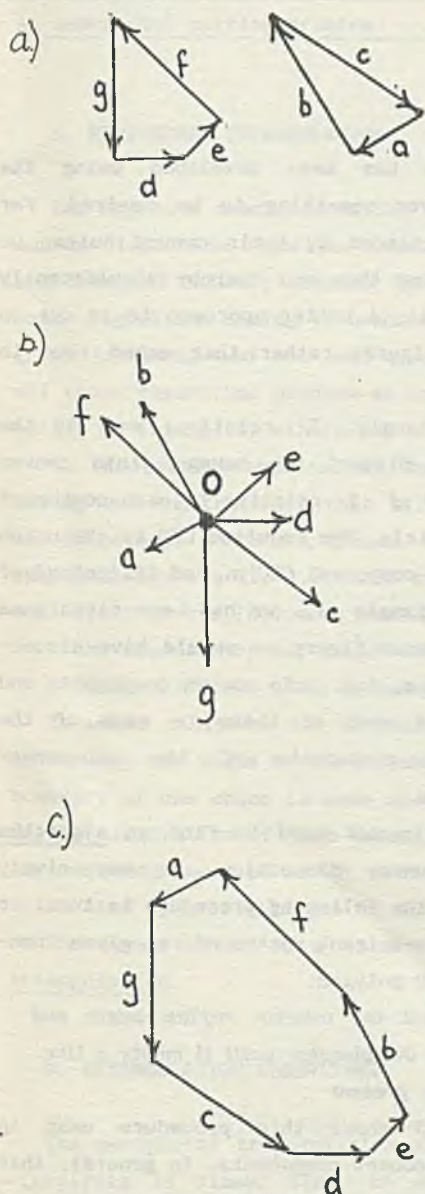


Fig. 3. The motion of datum point

Rys. 3. Ruch punktu odniesienia

It is quickly apparent that if we place T without rotation anywhere on the page, then T overlaps Q if the datum point A falls in the interior of heptagon H ; T touches Q if the datum point A falls on the boundary of H , and T and Q do not meet if the datum point A falls in the exterior of H .

The heptagon H is called the configuration space obstacle (CSO). Evidently all CSO's which arise if quadrilateral Q is initially translated to some other position are merely translates of H so it suffices to calculate any one of these translates. A method of calculation (as distinct from the physical process of wiping T round Q) is described next.



4. Calculating COS's

In Fig. 4(a), we have given directionality to the sides of the two polygons T and Q - clockwise for the polygon which moved during the "wiping" process (i.e. T) and anticlockwise for the other. The seven edge-vectors so created are bundled together, without change of magnitude or direction, at some convenient origin as in Fig. 4(b). They are then taken from the bundle in the order in which they lie anticlockwise about the origin (defagc in this case) and joined end to end in that order. A closed figure is produced, which is seen in this case to be a translate of the heptagon discussed in Section 2.

This procedure is called merging the two polygons T and Q. It works for any pair of convex polygons [2] and clearly has linear computational complexity. Unfortunately, shapes which occur in industrial problems are usually non-convex, but one way out of this problem is to replace shapes by their convex hulls, a computation which may be carried out in $O(n \log n)$ time for an n -gon [2].

Fig. 4. Calculation of the configuration space obstacles

Rys. 4. Wyznaczanie konfiguracyjnej przestrzeni przeszkód

5. CONVEX DISSECTION

In the industry, practical software has been developed using the principles so far explained but it leaves something to be desired. For example, if the vamps in Fig. 1 were replaced by their convex hulls, we should lose the possibility of interlocking them and thereby significantly reduce the level of utilization of material. A better approach is to cut up non-convex figures into smaller convex figures rather than embed them in larger ones.

Thus, to calculate the CSO of our triangle T relative, say, to the non-convex hexagon of Fig. 5, we first dissect the hexagon into convex components as shown, and calculate the CSO of T relative to each component

separately. The required CSO is the union of the component CSO's. And if, instead of the triangle T , we had been given some non-convex figure we should have dissected this, too, into convex components and related each of these to each of the convex components of the non-convex hexagon.

It is not hard to find an algorithm for convex dissection. We may simply apply the following procedure in turn to each re-entrant vertex of a given non-convex polygon:

Bisect the interior reflex angle and extend the bisector until it meets a line already present

Fig. 5. The procedure used to dissect the non-convex hexagon
Rys. 5. Zastosowanie procedury rozcięcia do niewypukłego sześciokąta

Fig. 5 shows this procedure used to dissect the non-convex hexagon into three convex components. In general, this algorithm produces $(v + 1)$ components if the original figure has v re-entrant vertices. In this general way we may arrive at a technique for cutting and packing highly irregular, non-convex shapes - such as the vamps of Fig. 1.

6. PERIPHERAL TRIANGULATION

The approach described so far still has some computational drawbacks. The convex dissection algorithm described in Section 4 has quadratic complexity and does not usually produce the least number of components. On the other hand, algorithms aiming to produce the least number of components are of very great intricacy and not well-adapted to practical computing [2]. Moreover, all these algorithms produce an unpredictable mixture of components: some may merely be triangles, others may be, say, 50-gons.

In the present application, however, we may avoid these difficulties by making use of the fact that all the given shapes are congruent. For two non-coincident congruent shapes in given positions, the condition that the shapes should overlap, i.e. have intersecting interiors, is easily seen to be equivalent to the following condition (C):

The boundary of one shape meets the interior of the other shape arbitrarily close to its boundary

To check condition (C) between two shapes in given positions it suffices to replace one shape by its boundary and the other shape by a thin shell. The boundary of one shape is made up of line-segments and the shell of the other may be constructed of thin triangles, as in Fig. 6. Then the CSO is made up of components each of which is the CSO of a line segment and a triangle, calculated by a trivial use of the merging procedure already described.

The construction of a thin shell out of triangles we call peripheral triangulation.

7. TRIANGULATION ALGORITHMS

The peripheral triangulation of a convex shape may be achieved straightforwardly in linear time, as suggested by Fig. 6. Let the vertices be X_1, \dots, X_n taken in anticlockwise sequence and define $X_0 = X_n$; $X_{n+1} = X_1$; $X_{n+2} = X_2$. On each edge $X_{r+1} X_{r+2}$ choose a point Y_{r+1} close to X_{r+1} ($r = 1, \dots, n$). Then the required shell is the union of the n triangles $\{X_r X_{r+1} Y_{r+1}\}$.

If the shape is not convex, the construction must be more carefully designed. For example, in Fig. 7, the shell produced by the chain of triangles has zero thickness at two points, so that the boundary of the other shape may cross without intersecting the interior of the shell.

In the appendix, we present a linear-time algorithm for peripheral triangulation. With each side $X_r X_{r+1}$ of the given shape, the algorithm associates a thin triangle $B_r C_r D_r$, such that line-segment $B_r C_r$ contains line-segment $X_r X_{r+1}$ with B_r close to X_r and C_r at X_{r+1} ; D_r is collinear with X_{r+1}, X_{r+2} and close to X_{r+1} .

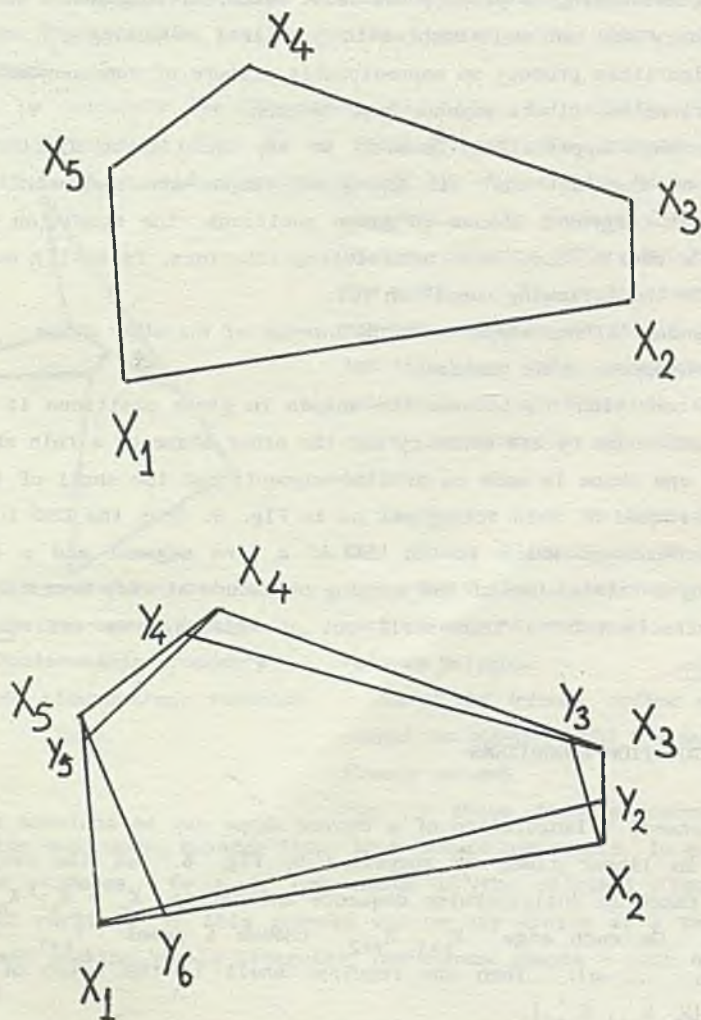


Fig. 6. Triangulation of a convex shape

Rys. 6. Triangulizacja wypukłych kształtów

Specifically, if X_r is a convex vertex then B_r is taken at X_r , otherwise B_r is taken exterior to the line-segment $X_r X_{r+1}$; similarly if X_{r+1} is convex vertex then D_r is taken interior to the line-segment $X_{r+1} X_{r+2}$, otherwise exterior. Fig. 8 shows the four cases of convexity/concavity of two consecutive vertices X_{r-1}, X_r , from which it will be seen that the shell does not have zero thickness at any vertex X_r .

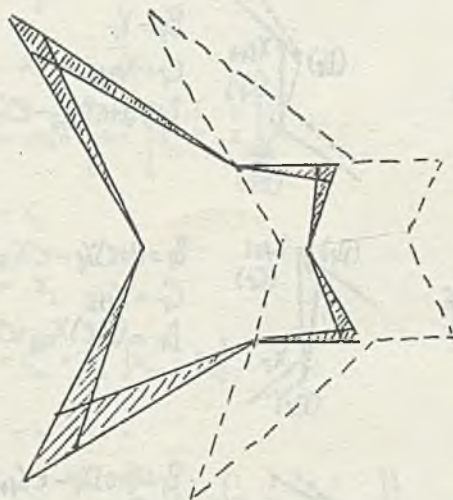


Fig. 7. The shell produced by the chain of triangles

Rys. 7. Powłoka wyznaczona przez łańcuch trójkątów

8. CONCLUSIONS

Until recently, industrial software for producing cutting layouts was rather slow, and occasionally unreliable. Above all, being frequently based on convex hulls, it was not capable of generating interlocking layouts of the kind shown in Fig. 1.

The program PAX which generated these interlocking layouts was written in interpreted BASIC for an Archimedes 310. It uses the above principle of peripheral triangulation for the calculation of CSO's and will generate a fully-interlocking efficient layout of this kind, from a polygonal specification, in something between 10 and 30 seconds.

FIGURE 8

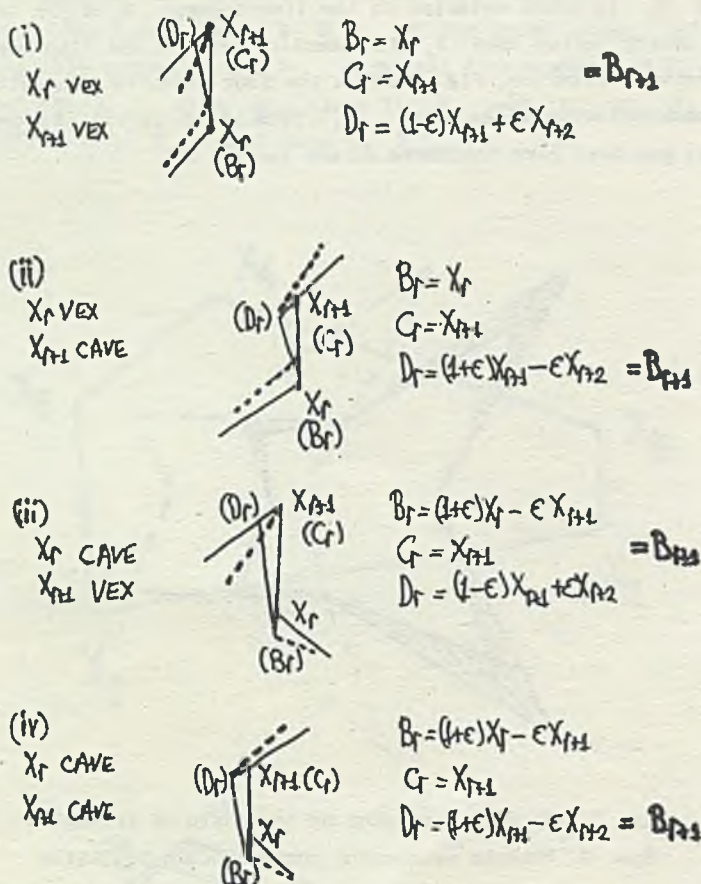


Fig. 8. Four cases of convexity/concavity of two consecutive vertices

Rys. 8. Cztery przypadki wypukłości/wklęsłości dla dwu przyległych wierzchołków

APPENDIX: PERIPHERAL TRIANGULATION

For simplicity, it is assumed that no three consecutive vertices are collinear, $\epsilon > 0$ is small.

INITIALISE $X_{n+1} : X_1$

$X_{n+2} : X_2$

$e_1 : = \epsilon \text{Sgn} \left(\det \begin{bmatrix} 1 & 1 & 1 \\ X_n & X_1 & X_2 \end{bmatrix} \right)$

$A_o : = X_2$

$C_o : = X_1$

$D_o : = C_o + e_1 (A_o - C_o)$

ALGORITHM For $r = 1$ TO n

$e_{r+1} : = \epsilon \text{Sgn} \left(\det \begin{bmatrix} 1 & 1 & 1 \\ X_r & X_{r+1} & X_{r+2} \end{bmatrix} \right)$

$A_r : = X_{r+2}$

$C_r : = A_{r-1}$

$D_r : = C_r + e_{r+1} (A_r - C_r)$

$B_r : C_{r-1} \quad \text{if } e_r > 0, \quad \text{ELSE} = D_{r-1}$

REFERENCES

- [1] Lyusternik L. A., Convex Figures and Polyhedra, D.C. Heath & Co, Boston 1966.
- [2] Toussaint G. (ed), Computational Geometry, N. Holland 1985
- [3] Papadimitriou C. H., Steiglitz K. "Combinatorial Optimization" Printice - Hall, New Jersey 1982.
- [4] Garfinkel R. S. Nemhauser G. L. "Integer Programming", John Wiley and Sons, New York 1972.

Recenzent: Doc.dr hab.inż. Andrzej Świerniak

Wpłynęło do Redakcji 2.11.1989 r.

PRZEMYSŁOWY PROBLEM CIĘĆ

S t r e s z c z e n i e

W pracy opisano algorytm zastępujący dwuwymiarowe kształty wielokątne wąskimi powłokami składającymi się z trójkątów. Upraszcza to wyznaczanie konfiguracyjnej przestrzeni przeszkód, która stanowi podstawowy krok w generacji rozmieszczenia cięć w zagadnieniach przemysłowych.

ИНДУСТРИАЛЬНАЯ ПРОБЛЕМА РЕЗКИ

Р е з ю м е

В работе представлен алгоритм заменяющий двучленные многоугольные формы узкими покровами состоящими из треугольников. Это упрощает определение конфигурационного пространства препятствий, которое представляет основной шаг в области генерации расположения резаний для промышленных вопросов.