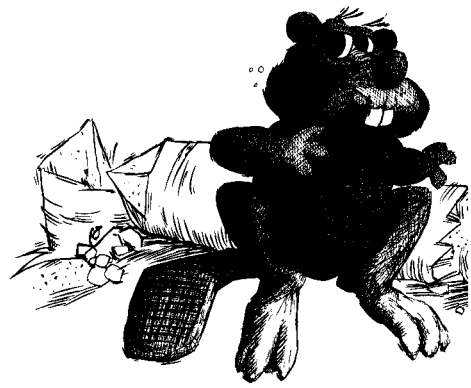
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT



Partitioning and Separating Sets of Orthogonal Polygons

Thomas Ottmann
Eljas Soisalon-Soininen
Derick Wood

Data Structuring Group
CS-86-53

October, 1986

Partitioning and Separating Sets of Orthogonal Polygons *

Thomas Ottmann † Eljas Soisalon-Soininen ‡ Derick Wood §

September 5, 1986

Abstract

In this paper we consider the computation of the orthogonal convex hull of a set of orthogonal simple polygons, where by orthogonal we mean that only horizontal and vertical orientations of edges are allowed. This, as we show, induces a partition of the set of orthogonal polygons. We provide an algorithm to compute the orthogonal-convex-hull partition of a set of p orthogonal polygons in $O(n \log p)$ time and $O(n)$ space, where n is the total number of vertices of the p polygons. Moreover we prove that this is both time and space optimal. For the case $p = 1$, an $O(n)$ time- and space-optimal algorithm is also presented. We simplify the description of the algorithms by making essential use of a decomposition theorem which we also prove.

As we shall demonstrate these results enable us to solve a number of separability problems for polygons. In particular, the group 4-way iso-separability of p polygons with a total of n vertices can be solved in $O(n \log p)$ time and $O(n)$ space.

1 INTRODUCTION

Orthogonal simple polygons, that is, simple polygons whose edges are parallel to the x- or y-axes, occur naturally as images on rectangular grids in image processing, see [7,20], or as building blocks in VLSI layout and wire

*The work of the second author was supported partially by the Academy of Finland and partially by the Alexander von Humboldt Foundation while visiting the first author, and that of the third author was supported under Natural Sciences and Engineering Research Council of Canada Grant No. A-5692.

†Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, Postfach 6980, D-7500 Karlsruhe, W. Germany.

‡Department of Computer Science, University of Helsinki, Tukholmankatu 2, SF-00250 Helsinki, Finland.

§Data Structuring Group, Computer Science Department, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

routing, see [11]. Despite this most studies have concentrated on points, orthogonal line segments, and orthogonal rectangles, see [2,3,5]. Recently however, [10] and [11] have both, independently, investigated the construction of orthogonal convex hulls for sets of orthogonal polygons. In both presentations, as pointed out in [13], some difficulties in the definition of orthogonal convex hulls for sets of polygons or points have been overlooked. However, it turns out that these definitional difficulties disappear if we form the orthogonal-convex-hull partition of a set of orthogonal polygons, rather than attempting to form a single connected orthogonal convex hull. A confirmation of this approach is to be found in [15] and [16]. It is this alternative approach that we take in the present paper which we feel gives, in the practical situations mentioned above, a more natural grouping of images and components.

This method of partitioning sets of objects is one of the most basic from the geometrical point of view and may be extended to include the convex-hull partition of arbitrary sets of objects in arbitrarily dimensioned space. The grouping or clustering of finite point sets has, of course, been studied intensively in the areas of statistics and classification, however little seems to have been done with regard to finite sets of objects other than points.

We illustrate the usefulness of the concept by solving some separability problems for orthogonal polygons. The study of separability in this sense was initiated in [4]; however, it can be considered to be a special case of the more general problem of motion planning, see [18,19,17], or of translation in graphics, see [8].

The paper consists of a further four sections. In Section 2 we discuss some previous work on the partitioning of sets of geometrical objects, while in Section 3 we provide the basic notions and terminology used throughout the paper. In Section 4 we first describe an algorithm to compute the orthogonal convex hull of a single orthogonal polygon. Both [10] and [11] have described similar algorithms to compute the orthogonal convex hull of a single orthogonal polygon; therefore, ours is included, primarily, to ensure the paper is self-contained. Second, we provide algorithms to compute the orthogonal convex-hull and orthogonal convex-hull partition of a set of orthogonal polygons before proving their space and time optimality. Finally, in Section 5 we define various notions of separability and apply the results of Section 4 to obtain time- and space-optimal algorithms for their associated decision problems.

2 PREVIOUS WORK ON PARTITIONING

Although the study of ortho-convex-hull partitioning is, to the best of our knowledge, new, there has been some previous work on partitioning sets of

geometrical objects. In this section we briefly survey these results.

Perhaps the most basic partition is the *connected partition* of a set of geometrical objects. We say a set of objects is *connected* if their union is a connected point set. The connected partition of a set S of geometrical objects is the maximal partition of S such that each element of the partition is a connected set, and is disjoint from all other elements. The elements of this partition are also called *connected components.*

In [6] the connected partitions of sets of orthogonal line segments and sets of orthogonal rectangles are shown to be computable in $O(n \log n)$ time and $O(n)$ space. Moreover the extension to three and higher dimensional orthogonal ranges, that is orthogonal "hyper-rectangles", is also considered.

A second partition stems from the study of safety in locked transaction systems initiated in [22,14]. They, essentially, define the *closure-connected partition* of a set of rectangles in the plane.

A point set in the plane is *diagonally closed* if for any two points (x, y) and (x', y') in the set such that $x' < x$ and $y' > y$ the points (x', y) and (x, y') also belong to the set. The *diagonal closure* of a set S of geometrical objects in the plane is the smallest diagonally-closed point set which contains all elements of S. The diagonal closure naturally induces a partition on S, called the *closure-connected partition*: Two objects are *closure connected* if they are contained in the same component of the diagonal closure.

In [9] and [21] the closure-connected partition of a set of orthogonal rectangles in the plane is computed and in the latter an $O(n \log n)$ time and $O(n)$ space algorithm is presented.

Third, in [21] the notion of a *rectangular partition* is introduced. Given a set of geometrical objects in the plane, its *minimal enclosing orthogonal rectangle*, or *bounding box*, is determined by the minimum and maximum x-coordinate and y-coordinate values of the corresponding point set. Again, given a set S of geometrical objects in the plane its rectangular partition is the maximal partition which satisfies the condition: For every pair of distinct elements of the partition their bounding boxes have no point in common. [21] provide an $O(n \log n)$ time and space algorithm to compute this partition. The space bound can, almost certainly, be reduced to $O(n)$.

3 BASIC NOTIONS

A geometrical object in the plane is said to be orthogonal if its edges are parallel to the x- or y-axes. A *polygon* is, for the purposes of this paper, an orthogonal simple polygon. We will speak of *orthogonal (straight) lines, orthogonal line segments*, and *orthogonal curves*, that is a line consisting of orthogonal line segments. We introduce immediately the central notion:

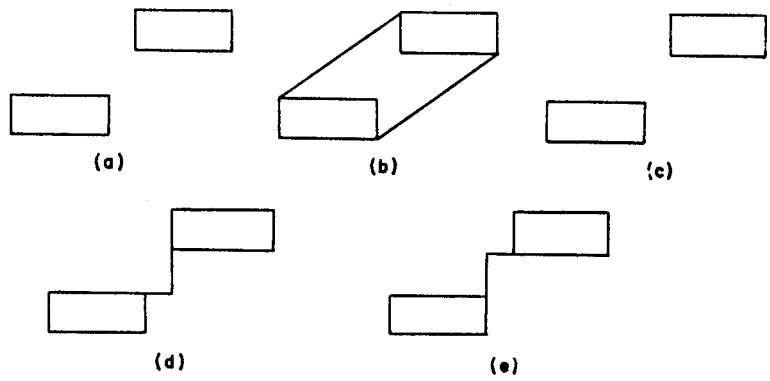


Figure 1: Various hulls of two rectangles

Definition 3.1 *We say a point set is* orthogonal convex *(or* ortho-convex*) if for every two points in the set that are the endpoints of an orthogonal line segment, the line segment lies wholly in the set.*

Given a point set its orthogonal convex hull *(or* ortho-convex hull*) is a smallest ortho-convex set containing the given set.*

Similarly, given a point set its connected orthogonal convex hull *(or* connected ortho-convex hull*) is a smallest connected ortho-convex set containing the given set.*

The notion of ortho-convexity is analogous to the usual notion of convexity, while the notions of ortho-convex hull (introduced in [13]) and connected ortho-convex hull (introduced in [10] and [11]) both, in their own way, capture some of the essence of the usual convex hull. These two notions are discussed in detail in [13], where their advantages and disadvantages are pointed out. The ortho-convex hull of a set may be a disconnected set, while the connected ortho-convex hull, though connected, is not necessarily unique. In Figures 1 (a)-(e) we display two rectangles, their convex hull, their ortho-convex hull, and two example connected ortho-convex hulls. On the other hand in Figure 2 we illustrate the unique connected ortho-convex hull of a set of polygons. In [16] it is shown that ortho-convexity and convexity are unified under restriction-orientation convexity; therefore, it appears that ortho-convexity is the appropriate notion of convexity in an orthogonal framework.

The "arms" occurring in Figures 1 (d) and (e) are, in a sense, unnatural. They occur simply because we require connectedness in the connected ortho-convex hull. These arms have, for the application we have in mind, little

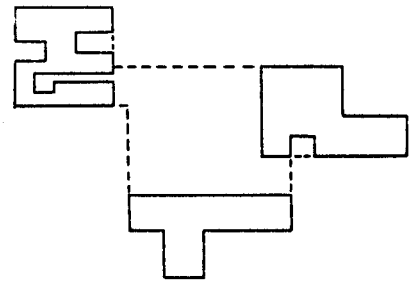


Figure 2: The connected ortho-convex hull of three polygons

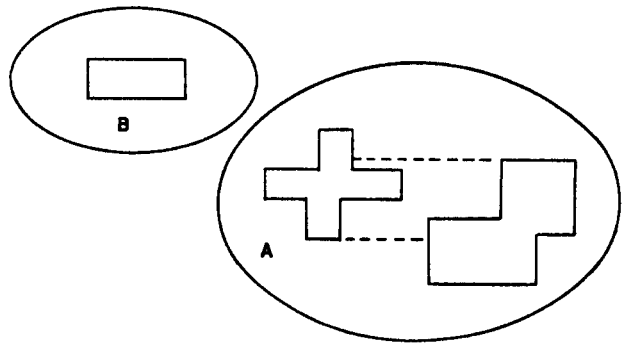


Figure 3: An ortho-convex-hull partition

or no meaning. Indeed they indicate that the objects they connect are really unrelated. For this reason we examine the ortho-convex hull as an alternative to the connected ortho-convex hull of polygons studied in [10] and [11]. The ortho-convex-hull of a set of polygons is the smallest ortho-convex point set containing the polygons. It naturally induces a partition on the set of polygons into pairwise disjoint components. We call this the *ortho-convex-hull partition*: Two polygons belong to the same component of the ortho-convex-hull partition if and only if they are contained in the same (connected ortho-convex) component of the ortho-convex-hull of the set of polygons. Clearly this partition is, in general, coarser than the connected partition of the set of polygons discussed in the previous section. In Figure 3 we display the ortho-convex-hull partition of a set of polygons; observe that element A of the ortho-convex-hull partition consists of two elements of the connected partition.

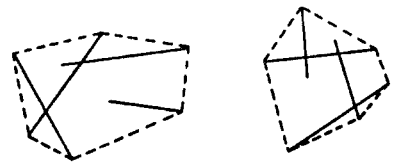


Figure 4: Convex hull partition of a set of line segments

As pointed out in [13] and proved in [16] the ortho-convex hull is not only unique, but also has a number of equivalent definitions as does the usual convex hull (this does not hold for connected ortho-convex hulls). We may define, in analogy, the *convex-hull closure* of a set as the smallest set which contains the given set such that each maximal connected subset of the set is convex. In Figure 4 the convex hull closure of a set of line segments is displayed. We leave this partition for future investigation.

An alternative formulation of the ortho-convex-hull partition of a set S of polygons, which is easily proved, is: *The ortho-convex-hull partition of S is the maximal partition of S such that the connected ortho-convex hulls of its elements are disjoint.*

In the next section, we present algorithms to compute the ortho-convex hull of an individual polygon and a set of polygons, as well as to compute the ortho-convex-hull partition in the latter case.

4 COMPUTING THE ORTHO-CONVEX HULL AND PARTITION

Rather than computing the ortho-convex hull or partition directly, we first demonstrate that it can be decomposed into simpler closure operations, which are easier to compute. For convenience throughout this section a set of polygons often means the point set given by their union. This should be clear from the context.

We begin by defining the simpler closure operations.

Definition 4.1 *Given a set S of polygons we say that S is* NX-closed, *if for every two points (x_1, y) and (x_2, y) which are connected in the set $S \cap \{(x', y') : y' \geq y\}$ and satisfy $x_1 \leq x_2$, then all points (x, y) are in S, where $x_1 \leq x \leq x_2$. In other words the horizontal line segment joining the two points is in S, if they are connected above the horizontal line passing through them. Similarly, S is* SX-closed *if for every two points (x_1, y) and (x_2, y)*

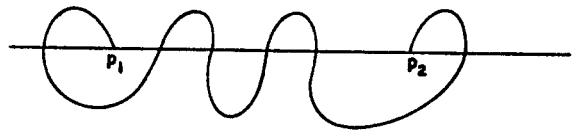


Figure 5: Illustrating the reason for the definition of NX-closure

which are connected in the set $S \cap \{(x', y') : y' \leq y\}$ *and satisfy* $x_1 \leq x_2$, *then all points* (x, y) *are in* S, *where* $x_1 \leq x \leq x_2$. *Further we say that* S *is* X-closed *if it is both* NX- *and* SX-*closed.*

The reader might wonder why we have introduced the notion of an X-closed set in this way and not as follows: A set S is X-closed if for all pairs of points on the same horizontal line which are connected in S the line segment joining the two points also belongs to S. In order to show that this latter condition is no stronger than the one chosen in Definition 4.1 assume that p_1 and p_2 are two points on the same horizontal line which are connected in S; see, for example, Figure 5.

Any curve connecting p_1 and p_2 in S, which does not lie either wholly to the north or wholly to the south of the straight line through p_1 and p_2, cuts this line into fragments. If S is both NX-closed and SX-closed according to Definition 4.1, each of these fragments must belong to S. Thus the whole line segment joining p_1 and p_2 must also belong to S.

Analogously we also have:

Definition 4.2 *Given a set* S *of polygons, we say that* S *is* WY-closed, *if for every two points* (x, y_1) *and* (x, y_2) *which are connected in the set* $S \cap \{(x', y') : x' \leq x\}$ *and satisfy* $y_1 \leq y_2$, *then all points* (x, y) *are in* S, *where* $y_1 \leq y \leq y_2$. S *is* EY-closed *is defined in a similar manner, and we have that* S *is* Y-closed *if it is both* EY- *and* WY-*closed. Finally we say that* S *is* XY-closed *if it is both* X- *and* Y-*closed.*

We now turn to the various notions of closure:

Definition 4.3 *Given a set* S *of polygons,* the NX-closure of S, *denoted by* $NX(S)$, *is the smallest* NX-*closed set containing* S. SX-, EY-, WY-, X-, Y-, *and* XY-closures *are defined similarly.*

The importance of these notions stems from:

Proposition 4.1 *Let* S *be a set of polygons. Then* $XY(S)$ *is the ortho-convex hull of* S.

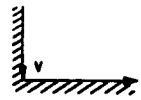


Figure 6: A down-right vertex

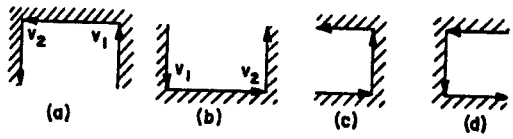


Figure 7: The four kinds of cavities

Proof: Clearly each connected component of $XY(S)$ is connected ortho-convex, and, moreover, $XY(S)$ is the smallest such set. Hence $XY(S)$ equals the ortho-convex hull of S. □

The following decomposition results are now obtained:

Lemma 4.2 *Let S be a set of polygons. Then*
(a) $SX(NX(S)) \subseteq X(S)$ and $NX(SX(S)) \subseteq X(S)$,
(b) $EY(WY(S)) \subseteq Y(S)$ and $WY(EY(S)) \subseteq Y(S)$, and
(c) $X(Y(S)) \subseteq XY(S)$ and $Y(X(S)) \subseteq XY(S)$.

Proof: (a) By definition $S \subseteq X(S)$ and thus $SX(NX(S)) \subseteq SX(NX(X(S)))$. But $SX(NX(X(S))) = X(S)$, because $X(S)$ is both NX- and SX-closed. Similar arguments apply for $NX(SX(S)) \subseteq X(S)$, (b), and (c). □

We next demonstrate that the inclusions converse to those given in Lemma 4.2 also hold. To do this we characterize the form of the contour of the different closure operators.

A vertex v of a polygon is said to be a *down-right vertex*, if the preceding vertex (in the clockwise order of vertices) is above v in the polygon, and the succeeding vertex is to the right of v, see Figure 6. In a similar way *down-left*, *up-right*, *up-left*, *left-up*, *left-down*, *right-up*, and *right-down* vertices are defined. A polygon is said to have an *upwards cavity* at vertices v_1 and v_2, if v_1 and v_2 are consecutive vertices, v_1 is an up-left vertex, and v_2 is a left-down vertex, see Figure 7(a). Note that we assume the interior of the polygon lies to the right when traversing the vertices in clockwise order. A polygon has a *downwards cavity* at v_1 and v_2, if v_1 and v_2 are consecutive

vertices, v_1 is a down-right vertex, and v_2 is a right-up vertex, see Figure 7(b). Similarly, *rightwards* and *leftwards cavities* at vertices v_1 and v_2 are defined, see Figures 7(c) and (d).

The following lemma summarizes the rôle of these cavities.

Lemma 4.3 *A polygon is:*

(a) NX-closed iff it has no upwards cavities,
(b) SX-closed iff it has no downwards cavities,
(c) X-closed iff it has no upwards nor downwards cavities,
(d) WY-closed iff it has no leftwards cavities,
(e) EY-closed iff it has no rightwards cavities,
(f) Y-closed iff it has no leftwards nor rightwards cavities, and
(g) XY-closed iff it has no cavities.

Proof: Immediate from the definitions. □

Theorem 4.4 Decomposition Theorem *Let S be a set of polygons; then:*

(a) $SX(NX(S)) = NX(SX(S)) = X(S)$,
(b) $EY(WY(S)) = WY(EY(S)) = Y(S)$, *and*
(c) $X(Y(S)) = Y(X(S)) = XY(S)$.

Proof: Consider (a) only, the others follow by similar arguments. Each NX-closed set containing S must also be a set S' of polygons composed of at most as many polygons as S. By the definition of NX-closure, each polygon in S' is NX-closed and, thus, has no upwards cavities (Lemma 4.3(a)). Similarly Lemma 4.3(b) implies that $SX(NX(S)) = SX(S')$ is a set of polygons each having no downwards cavities. We now conclude that $SX(S')$ has neither downwards nor upwards cavities, because SX-closure does not introduce new upwards cavities, it only preserves existing ones. Hence each polygon in $SX(S')$ is X-closed (Lemma 4.3(c)), which means that $SX(S')$ is X-closed. Finally, because $SX(NX(S)) \subseteq X(S)$ by Lemma 4.2(a), and $X(S)$ is the smallest X-closed set containing S, we have equality. That $NX(SX(S)) = X(S)$ follows similarly. □

We are now in a position to describe algorithms for the computation of the ortho-convex hull of, first, a single polygon, and, second, of a set of polygons. Although both [10] and [11] present optimal algorithms for the first, our algorithm is simple in that it is based on Theorem 4.4. We first divide the given polygon into quarters, for example the *NW-quarter* is the portion of the polygon from the leftmost vertex (the topmost one if there is more than one leftmost vertex) to the topmost vertex (the rightmost one if there is more than one). Similarly we have the NE-, SE-, and SW-quarters.

The crucial idea behind this quartering is that computing the SX-closure of the polygon, for example, can be considered to be the SX-closure of the NW-quarter, the EY-closure of the NE-quarter, the NX-closure of the SE-quarter, and the WY-closure of the SW-quarter, *when* we assume the polygon is rotated by a quarter turn in a counter-clockwise direction after each operation has been carried out. Similar sequences are obtained for the SX-, EY-, and WY-closures, leading, by Theorem 4.4, to the XY-closure of the given polygon.

For this reason we only describe an algorithm to compute the SX-closure of the NW-quarter of a polygon, and then mention how it can be modified to compute the NX-, EY-, and WY-closures.

Algorithm *SX-Closure of the NW-Quarter of an R-Polygon*;

Input A sequence of vertices $v_1, \ldots, v_m$ in clockwise order, called the input list, determining the NW-quarter of a polygon as defined above. In other words v_1 is the leftmost and topmost vertex in the polygon, and v_m is the topmost and rightmost vertex.

Output A sequence of vertices $v_1', \ldots, v_p'$ in clockwise order where $v_1' = v_1$, and $v_p' = v_m$, called the output list; the vertices of the SX-closure of $v_1, \ldots, v_m$.

begin Scan the input list producing an output list of vertices (the vertices of the SX-closure). On meeting each vertex in the input list, it is either copied into the output list or discarded according to the following rules:

(a) If the current input vertex is a right-down vertex it is discarded together with all following vertices until one is met which is at least as high as the current output vertex. If it is the same height simply add it to the output list. Otherwise add a new vertex to the output list having the same y-coordinate as the current output vertex and having the same x-coordinate as the current input vertex, and add the current input vertex to the output list.

(b) If the current input vertex is not a right-down vertex, it is added to the output list, to become the current output vertex.

end of Algorithm.

To obtain, for example the NX-closure of the NW-quarter, we scan the vertices in reverse order and begin to discard them when a down-left vertex is met and continue discarding until a vertex no higher than the current output vertex is reached.

Thus we have proved:

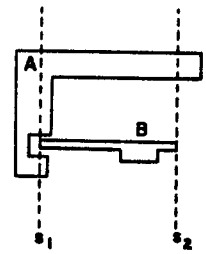


Figure 8: Illustrating how cavities are filled

Theorem 4.5 *Given an arbitrary polygon with n vertices, $n \geq 4$. Then the Z-closure of the polygon, where $Z \in \{NX, SX, EY, WY, X, Y, XY\}$, can be computed in $O(n)$ time and space, and this is optimal.*

Having demonstrated how the SX-closure of a single polygon can be computed efficiently, we consider now an algorithm to compute one of the basic closures of a set of polygons, using the plane-sweep paradigm, (see [12]). If we sweep from bottom-to-top then we will compute the SX-closure, and to compute the NX-closure we will sweep from top to bottom. We choose to sweep from left-to-right, which enables us to compute the WY-closure efficiently. Apart from the sweeping direction each of the basic closures is computed in an identical manner. Alternatively one may consider that the set of polygons is rotated by quarter turns, the WY-closure algorithm is executed, and the output set of polygons is rotated in the reverse direction for the same number of turns to compute any of the other closures.

Observe that in the definition of an WY-closed set of polygons, a vertical line segment joining two points is in the set only if the two points are connected to the left of the vertical line they determine. This definition was made with plane-sweeping in mind. As the vertical sweep-line sweeps the plane from left to right, we will sweep through and fill in leftwards cavities found at the current position of the sweep-line. In Figure 8 we see that at position S_1, A and B are connected because the leftwards cavity in A will have been filled in. This in turn will lead to the filling in of the new cavity between A and B by the time S_2 is reached. This is a very similar approach to that taken in [21] to compute a similar closure operator, cf. Section 2.

Essentially at any position of the sweep-line we need to maintain the connected components formed so far. Clearly we only need sweep in discrete (rather than continuous) steps through the set of polygons, since changes to the connected components only occur when vertices of the polygons are

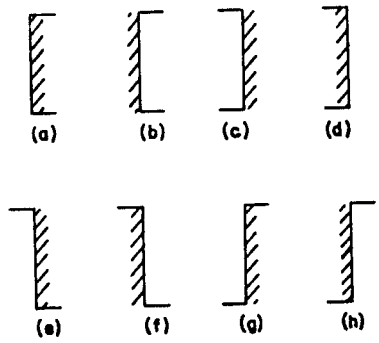


Figure 9: The eight kinds of vertical edges

met.

To simplify somewhat the exposition of the WY-closure algorithm we will first only compute the partition of the set of polygons induced by the WY-closure. After giving this algorithm we will then explain how it should be modified to produce the actual WY-closure.

This means that we are required to maintain the current elements of the partition at each position of the sweep-line. and we need to update these elements correctly as we pass each *sweep-point*. We say an element is *active* if it is cut by the sweep-line. Each active element determines a number of disjoint *active y-intervals* or *waves*, its intersection with the sweep-line at its current position.

We will now discuss the algorithm in a little more detail before specifying it completely. As the sweep-line passes through the polygons, it is only their vertical edges which are of interest, their horizontal edges may be ignored. For this reason, when determining the sorted sweep-points we, essentially, consider the set of polygons to be a set of vertical edges. However, we retain, with each vertical edge, its type and the polygon to which it belongs. There are eight types of vertical edges, specified by the two adjacent horizontal edges and the inside of its polygon, see Figure 9. New intersections need to be detected on meeting edges of Types (a), (c), (e), and (g), that is starting edges, in analogy with the rectangle case. Each edge type also affects the current waves, for example Type (a) serves to begin new waves, while Type (d) ends waves. Note that Type (b), corresponding to a leftwards cavity, does not affect the current wave.

Since the algorithm must keep track of the current connected components under WY-closure, then one of the many available data structures

for the UNION-FIND problem (see [1]) is used. We call this the *union-find* structure. We only require that each operation takes $O(\log n)$ time, where n is the total number of vertices in the p polygons, since this will ensure that a worst case bound of $O(n \log n)$ time is obtained for a complete sweep.

Now as in [6] we need to keep track not only of the current waves associated with each component, but also of the detailed composition of each wave, that is which polygons gave rise to it. Each wave is a connected set, unlike the waves in [6] where they can have a nested structure. We choose to represent waves in two structures, the first, the *wave structure*, simply holds all the current waves, and the second keeps the finer details for each wave separately; we call this the *background structure*. Now consider the operations that these structures are required to support.

On meeting a Type (a), (c), (e), or (g) edge we need to:

(i) Determine any intersections, that is query the wave structure for the waves which intersect the given vertical edge, and

(ii) Merge the components corresponding to the waves, if necessary.

(iii) Merge these waves with the wave specified by the given vertical edge.

On meeting a Type (b), (d), (f) or (h) edge we need to:

(i) Determine which wave it belongs to,

(ii) Determine what shrinkage, if any, it will cause in this wave. Since waves are represented by disjoint intervals on the sweep-line, we represent them in a balanced leaf search tree with the y-coordinates of their endpoints. This is our wave structure. The background structure for each wave may also be represented by a balanced search tree. For each polygon contributing to the wave, it holds all y-coordinate values of its currently active horizontal edges.

Algorithm *WY-Closure Partition*;

Input A set of p polygons in the plane with a total of n vertices, $n, p \geq 1$.

Output The WY-closure partition of the polygons.

begin

Step 1 If $p = 1$ then apply the single polygon algorithm. Otherwise, considering each polygon to be a collection of vertical edges, determine the sorted sweep points.

Step 2 Scan through the sweep-points in sorted order, and at each sweep point carry out one of the following operation sequences according to the type of the edge: (a), (c), (e), (g): Treating the edge as a wave w, query the wave structure for any intersections; let these be $w_1, \ldots, w_r$, $r \geq 0$. Determine the components to which w, $w_1, \ldots, w_r$

belong. Merge the corresponding components if necessary. Replace the waves $w_1, \ldots, w_r$ by a new wave determined by their intersection with w, w' say, construct a new background structure for w', from the corresponding background structures for $w_1, \ldots, w_r$ and, possibly, w. (b), (d), (f), (h): Query the wave structure to determine which wave w it belongs to. If it is a Type (d), (f) or (h) edge then determine, from the background structure, whether or not its removal will either cause w to shrink or have no effect on w. In the former cases update the wave structure. In all cases update the background structure.

end of Algorithm.

Let us examine the time taken by each step in the above algorithm. Step 1 clearly takes, at worst, $O(n \log n)$ time, thus consider Step 2.

Step 2 (a), (c), (e), (g): Obtaining all intersections with w takes $O(\log n + r)$ time, where $r = O(n)$, in the worst case, is the number of intersecting waves. We carry out $r+1$ FIND instructions, and at most r UNION operations; we charge these to the algorithm as a whole, rather than to this step of it. We require r waves to be deleted and one new one to be inserted; again we charge the deletions to the algorithm as a whole; the insertion can be carried out in $O(\log n)$ time. Clearly the new wave w' can be computed in $O(1)$ time. The background structure for w' can be formed by catenating the background structures for $w_1, \ldots, w_r$, since they are consecutive waves, and updating it with w. We know from [1] that if the background structures are 2-3 trees, for example, then two such structures can be catenated in $O(\log n)$ time. We, once more, charge the catenation of background structures to the whole algorithm.

Step 2 (b), (d), (f), (h): The initial wave query, is in this case, a simpler one than in the previous step, we may simply query the wave structure with a single y-coordinate value of the line segment. This takes $O(\log n)$ time, at most. If it is Type (b) we simply add its endpoint y-coordinates to the background structure, which requires $O(\log n)$ time. For each of the other types shrinkage of w may occur, and this is determined by deleting one or two points from the background structure for w and checking for shrinkage. This takes $O(\log n)$ time. Finally at most one point is added to the background structure, again taking $O(\log n)$ time.

Apart from the operations charged to the algorithm, a single step requires $O(\log n)$ time, and hence $O(n \log n)$ time for the $O(n)$ steps required by Steps 1 and 2. It remains to compute the time charged to the algorithm as a whole in Step 2. We claim that overall there can be, at worst, $O(n)$ FINDs, UNIONs, wave deletions, and wave catenations giving an additional cost of $O(n \log n)$, that is algorithm WY-closure requires $O(n \log n)$ time overall. To see this, first observe that only Type (a) edges introduce new

waves and there can be at most n of them. Second, whenever waves are joined, they are never separated again; they may only shrink. Therefore there can only be at most $n-1$ joins or UNIONs.

We can summarize the above discussion in:

Theorem 4.6 *Given a set of polygons in the plane, with a total of n vertices, $n \geq 1$. Then their Z-closure partition can be computed in $O(n \log n)$ time and $O(n)$ space in the worst case, where $Z \in \{NX, SX, EY, WY\}$.*

The space requirement follows immediately by observing that all supporting structures require only $O(n)$ space.

To extend this result to X- and Y-closure (and XY-closure) we may either use a second scan (and a third scan) as was done in [21], or modify the above algorithm to output the new set of WY-closed polygons. We take this latter course, since it leads to a uniform approach to all the closure operators, namely polygons in and polygons out.

Algorithm *WY-Closure*;

Input A set of polygons in the plane with a total of n vertices, $n \geq 1$.

Output The polygons constituting the WY-closure of the input polygons.

begin When an incoming vertical edge in *WY-Closure-Partition* cuts some of the current waves, in at most two points, the end points of this edge are added to a vertex list L. Otherwise, if an incoming edge causes a change in some wave, both the old and the new end point of the wave are added to L. (If some wave disappears altogether, both its end points are added.) As is easily seen, the vertex list L contains all the vertices of the polygons in the WY-closure and maybe some additional points. These extra points occur, however, only when three or more points in L lie on the same vertical or horizontal line, and they can be easily identified and discarded when the final sorted lists of vertices determining polygons have been produced. Vertices in L are partitioned into lists $L_1, \ldots, L_m$ according to the m elements in the partition produced by *WY-Closure-Partition*. Vertex v is in L_i, if the i-th element in the partition corresponds to a polygon having v on its contour. This classification is performed by consulting the union-find structure built by *WY-Closure-Partition* with the edge which caused v to be in L. The lists $L_1, \ldots, L_m$ are sorted to obtain a clockwise ordering of the vertices in the corresponding polygons. Finally, if some list contains more than two consecutive elements lying in the same vertical or horizontal line, only the first and the last of these are left in the list and the others are discarded.

end of Algorithm

The time needed for all of the above tasks is bounded by $O(n \log n)$. We now have:

Theorem 4.7 *Given a set of p polygons in the plane, with a total of n vertices, $p \geq 1$, their Z-closure and their Z-closure partition can be computed in $O(n \log p)$ time and $O(n)$ space in the worst case, where $Z \in \{X, Y, XY\}$.*

Proof: Consider the Y-closure of the set of polygons, the bounds for the X- and XY-closure follow directly by the same arguments.

If $p = 1$ we require $O(n)$ time and space by Theorem 4.5. Otherwise $p > 1$ and we proceed as follows. First compute the Y-closure of each of the p polygons in turn. This requires $O(n)$ time and space once more. Second, execute *WY-Closure-Partition* on the resulting set, modifying Step 1 so as to perform the sorting of sweep-points by merging $2p$ upper and lower contour sorted lists of points. By Lemma 4.3 (f) the vertices in the upper and lower contours of each Y-closed polygon appear in sorted x-order. Thus these $2p$ lists are obtainable in $O(n)$ time and space, while performing their merger requires $O(n \log p)$ time and $O(n)$ space. Apart from this minor change *WY-Closure-Partition* proceeds as before. But in analyzing its time bounds the remaining steps now require only $O(n \log p)$ time and $O(n)$ space. This is because there can never be more than p waves and no more than two active horizontal edges for each polygon at any one time, since each polygon is Y-closed. Third, compute the Y-closure of the, at most p, polygons obtained as output from *WY-Closure*, and then execute *EY-Closure* using the same merge sorting technique in Step 1. This, by Theorem 4.4, results in the Y-closure partition of the set of p polygons and also their Y-closure in $O(n \log p)$ time and $O(n)$ space. □

5 SEPARABILITY

Given a set of p orthogonal polygons in the plane we ask whether they can be separated(or translated) to infinity without collisions [4]. This problem has been studied as part of motion planning, see [18,19,17], and as a graphics problem, see [8]. If we require the polygons to be moved *one at a time* to infinity, *in the same direction*, we call this the iso-separability problem. It is clear that iso-separation to the east(eastern separation) is possible if and only if western separation is possible, and similarly for north and south. Define a set to be *x-convex* if, for every two points in the set that are the endpoints of a horizontal line segment, the line segment is in the set. Then, the *x-convexhull* of a set is the smallest x-convex set containing it. We define

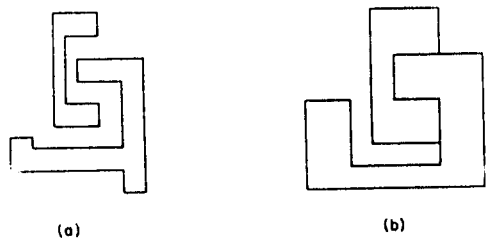


Figure 10: Eastern separable and nonseparable examples

y-convex and *y-convexhull* similarly. Now, eastern separation is possible if and only if the x-convex hulls of the p polygons do not intersect; see Figure 10 for eastern separable and nonseparable examples. But these observations imply:

Theorem 5.1 *Given a set of p polygons with a total of n vertices, iso-separability can be decided in $O(n \log p)$ time and $O(n)$ space.*

Proof: Using the algorithm of [3] adapted for finding one intersection in a set of orthogonal polygons. □

If we require the polygons to be moved, *in any one of the four possible directions*, one at a time to infinity, we call this the *4-way iso-separation problem.* A set of polygons is 4-way iso-separable if and only if it is both x- and y-separable. Thus it is 4-way iso-separable if and only if the x-convex hulls of the polygons do not intersect and the y-convex hulls do not intersect. Hence we have:

Theorem 5.2 *Given a set of p polygons with a total of n vertices, 4-way iso-separability can be decided in $O(n \log p)$ time and $O(n)$ space.*

But what if a set of polygons is neither iso-separable nor 4-way iso-separable? In these cases we ask for the maximal partition into either iso-separable or 4-way iso-separable clusters. The associated problems are called the *group iso-separability* and the *group 4-way iso-separability* problems. Let us consider them one at a time.

If we can find the maximal x- and y-partitions, we can clearly compute the maximal iso-partition. But this problem falls to the same method of attack used above, namely, first, we find the x-convex hulls of each polygon and, second, we compute the connected components of the result. Since the connected components algorithm of [6] requires $O(n \log p)$ time and $O(n)$ space for p x-convex polygons, we have:

Theorem 5.3 *Given p orthogonal polygons in the plane with a total of n vertices, group iso-separability can be solved in $O(n \log p)$ time and $O(n)$ space.*

Of course, we may be left with the trivial partition, that is one connected component.

Second, the group 4-way iso-separability partition corresponds exactly to the XY-closure partition we have computed in Section 4. Clearly the elements of the XY-closure partition are, by definition, iso-separable in any of the four directions. Moreover, no larger partition can be 4-way iso-separable, since the x- or y-convex hulls of two elements of such a partition must intersect. Thus our final separability result can be stated as:

Theorem 5.4 *Given p orthogonal polygons in the plane with a total of n vertices, group 4-way iso-separability can be solved in $O(n \log p)$ time and $O(n)$ space.*

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading,Mass., 1974.

[2] J.L. Bentley, D. Haken, and R.W. Hon. Fast geometric algorithms for VLSI tasks. In *Proceedings of the Spring COMPCON*, pages 88–92, 1981.

[3] J.L. Bentley and D. Wood. An optimal worst case algorithm for reporting intersections of rectangles. *IEEE Transactions on Computers*, EC-29:571–576, 1980.

[4] B. Chazelle, Th. Ottmann, E. Soisalon-Soininen, and D. Wood. The complexity and decidability of SEPARATION^{TM}. In *ICALP'84,J. Paradaens(ed.),Springer-Verlag Lecture Notes in Computer Science 172*, pages 119–127, 1984.

[5] D. Dolev, K. Karplus, A. Siegel, A. Strong, and J.D. Ullman. Optimal wiring between rectangles. In *Proceedings of the Thirteenth ACM Symposium on Theory of Computing*, pages 312–317, 1981.

[6] H. Edelsbrunner, J. van Leeuwen, Th. Ottmann, and D. Wood. Computing the connected components of simple rectilinear geometrical objects in d-space. *RAIRO Informatique théoretique*, 18:171–183, 1984.

[7] L. Ferrari, P.V. Sonka, and J. Sklansky. Minimal rectangular partitions of digitized blobs. In *Proceedings of the Fifth International Conference on Pattern Recognition*, pages 1040–1043, 1980.

[8] L.J. Guibas and F.F. Yao. On translating a set of rectangles. In *Proceedings of the Tenth Annual ACM-SIGACT Symposium on Theory of Computing*, pages 154–160, 1980.

[9] W. Lipski and C.H. Papadimitriou. A fast algorithm for testing for safety and detecting deadlocks in locked transaction systems. *Journal of Algorithms*, 2:211–226, 1981.

[10] D.Y. Montuno and A. Fournier. *Finding the x - y Convex Hull of a Set of x - y Polygons.* Technical Report 182, University of Toronto, CSRG, 1982.

[11] T.M. Nicholl, D.T. Lee, Y.Z. Liao, and C.K. Wong. *Constructing the X-Y Convex Hull of a Set of X-Y Polygons.* Technical Report, IBM Research Center, 1982.

[12] J. Nievergelt and F.P. Preparata. Plane-sweep algorithms for intersecting geometric figures. *Communications of the ACM*, 25:739–747, 1982.

[13] Th. Ottmann, E. Soisalon-Soininen, and D. Wood. On the definition and computation of rectilinear convex hulls. *Information Sciences*, 33:157–171, 1984.

[14] C.H. Papadimitriou. Concurrency control by locking. *SIAM Journal on Computing*, 12:215–226, 1983.

[15] G.J.E. Rawlins and D. Wood. Optimal computation of finitely-oriented convex hulls. *Information and Control*, 1986.

[16] G.J.E. Rawlins and D. Wood. *Restricted-Orientation Convexity.* Technical Report, University of Waterloo, Computer Science, 1986.

[17] J.T. Schwartz and M. Sharir. On the piano mover's problem: I. The special case of a rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, to appear, 1983.

[18] J.T. Schwartz and M. Sharir. On the piano mover's problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, to appear, 1983.

[19] J.T. Schwartz and M. Sharir. On the piano mover's problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. to appear, 1983.

[20] J. Sklansky. Measuring concavity on a rectangular mosaic. *IEEE Transactions on Computers*, EC-21:1355–1364, 1972.

[21] E. Soisalon-Soininen and D. Wood. Optimal algorithms to compute the closure of a set of iso-rectangles. *Journal of Algorithms*, 5:199–214, 1984.

[22] M. Yannakakis, C.H. Papadimitriou, and H.T. Kung. Locking policies: safety and freedom from deadlock. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, pages 286–297, 1979.