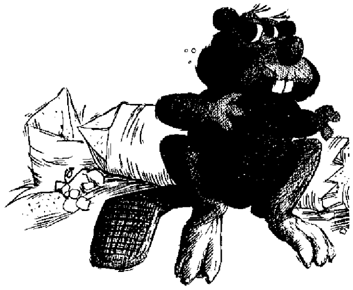
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT



An Isothetic View of Computational Geometry

Derick Wood

Data Structuring Group
CS-84-01

January, 1984

AN ISOTHETIC VIEW OF COMPUTATIONAL GEOMETRY[†]

Derick Wood[‡]

ABSTRACT

We survey a number of results, some new, for isothetic polygons in the plane. Specifically we consider intersection, convexity, combinational, clustering and visibility problems. These problems occur in image processing, VLSI design and layout, circuit testing, transaction systems, and pattern recognition.

1. INTRODUCTION

Since Michael Shamos [Sh] carried out his pioneering work in computational geometry the field has blossomed considerably. The purpose of this chapter is to review some of the developments in computational geometry since its inception. In order to do justice to the field I have been obliged, albeit not unwillingly, to restrict my attention to the plane and to isothetic or rectilinearly-oriented polygons. Since the first problem I studied in computational geometry was the rectangle intersection problem with Jon Bentley [BW] and my most recent work [Wo] involves isothetic polygons, this restriction is a happy one. This survey provides an opportunity to draw together the variety of problems for isothetic objects into some coherent whole, while also allowing the formal treatment of isothetic polygons in the style of Shamos [Sh].

Isothetic polygons occur in VLSI design, in floor and street plans, in data bases, in CAD/CAM, in NC, in circuit routing, in circuit testing, in robotics, in locked transaction systems, in image processing, in pattern recognition, and in Saskatchewan. Often they can be viewed as discrete approximations to polygons in the real plane, for example in image processing and pattern recognition, however just as often they appear in their own right, for example in VLSI designs. From a theoretical viewpoint isothetic polygons

[†] Work carried out under a Natural Sciences and Engineering Research Council of Canada Grant No. A-5692.

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

are usually (but not always) easier to handle than arbitrary polygons. Because of this it is often possible to obtain time- and space-optimal algorithms for isothetic polygons, when this is not the case in general. Finally, isothetic polygons are the source of many beautiful problems - an important issue for a researcher.

Section 2 provides the basic definitions for isothetic objects. In Sections 3 through 7 we study intersection, convexity, combinational, clustering, and visibility problems, respectively. Section 8 is a closing discussion of the results, the methodology, and some of the omissions.

2. DEFINITIONS

We begin by defining our objects of study, namely isothetic lines, line segments, curves and polygons. We assume all objects are in the plane except when explicitly stated otherwise. We assume the plane has an embedded Cartesian coordinate system.

An *isothetic* (*straight*) *line* is a straight line parallel to either the x-axis or y-axis. Similarly, an *isothetic line segment* is a line segment parallel to either the x- or y-axis, and an *isothetic ray* is a ray parallel to either the x- or y-axis. See Figure 2.1(a), (b) and (d).

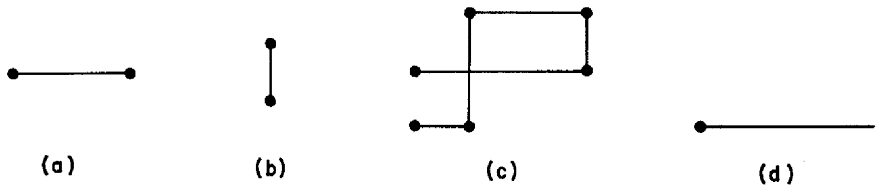


Figure 2.1

An *isothetic curve* is a connected sequence of isothetic line segments, see Figure 2.1(c).

An isothetic curve is *closed* if starting at any point p, on the curve, a path can be traced along the curve, visiting every point once and only once, which leads to p.

An isothetic curve is *simple* if:

(i) self-intersections only occur at endpoints, and

(ii) no more than two segments intersect at each endpoint.

A non-closed simple isothetic curve is a *staircase* if either for all points p

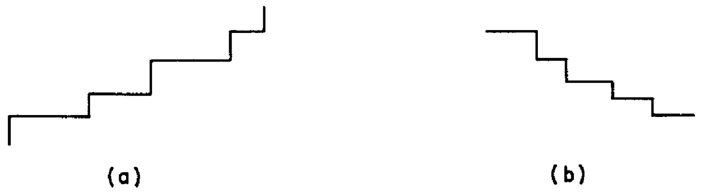


Figure 2.2

and q on the curve, with p to the left of q, p is not above q, or for all points p and q on the curve, with p to the left of q, p is not below q. See Figure 2.2.

An *isothetic polygon* is a closed isothetic curve, which forms its boundary. We also consider a polygon to be the closed finite region of the plane enclosed by the closed curve. Note that this definition excludes the possibility of a polygon having *holes*. A *simple isothetic polygon* is a closed simple isothetic curve. See Figure 2.3(a) for a simple polygon and 3(b) for a non-simple polygon.

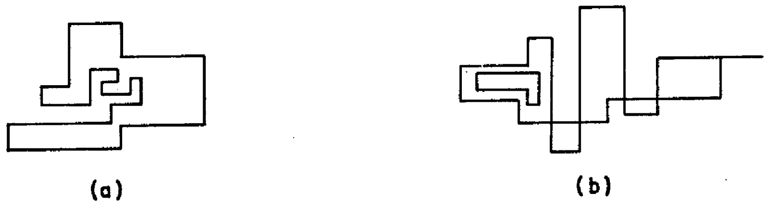


Figure 2.3

We often wish to distinguish polygons not only with respect to their simpleness, but also with respect to their isothetic convexity.

A simple isothetic polygon is *isothetically convex* if every isothetic line has either an empty intersection with the polygon or an isothetic line segment as its intersection. Figure 2.3(a) is a non-convex polygon, while Figure 2.4 displays a convex polygon. A special case of a convex polygon is a *rectangle*.

Observation *Every convex isothetic polygon can be decomposed into at most four staircases. In Figure 2.4 there are three:* 1 *to* 2, 2 *to* 3, *and* 3 *to* 1.

If a polygon is non-convex we often wish to refer to its convex hull, which is defined as follows:

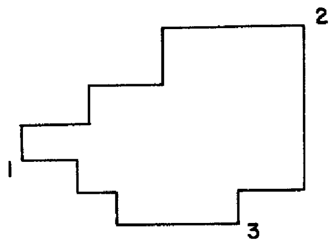


Figure 2.4

The *isothetic convex hull* of a simple isothetic polygon is the smallest, in area, convex simple isothetic polygon containing the given polygon. Figure 2.5 displays the convex hull of Figure 2.3(a).

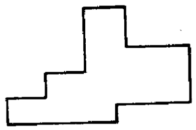


Figure 2.5

Already our definitions lead to three natural problems:

Problem 1 *Determine if a given isothetic polygon is simple.*

Problem 2 *Determine if a given simple isothetic polygon is convex.*

Problem 3 *Determine the convex hull of a given simple isothetic polygon.*

In the next section we will describe an algorithm to determine whether or not an isothetic polygon is simple, while the following section will discuss convexity.

3. INTERSECTION PROBLEMS

Throughout this and the following sections we will omit the adjectives *isothetic* and *simple*. All objects are isothetic and all polygons are simple unless it is explicitly stated otherwise. We first state:

Problem 4 *Given a collection of polygons determine if there are two* intersecting *polygons in the collection, that is two polygons having a common point.*

Problem 5 *Given a collection of polygons determine all pairwise intersecting polygons.*

Problem 6 *Given a collection of polygons and an integer* $k \geq 2$, *determine if there is a* k-intersection, *that is* k *polygons having a common point. In particular is there a p-intersection?*

Problem 7 *Given a collection of polygons determine the maximal* k *for which there is a k-intersection, that is the* thickness *of the given polygons.*

Using the *sweep line* paradigm, Shamos and Hoey [ShH] gave an $O(p\log p)$ time and $O(p)$ space algorithm for Problem 4 for the case of p line segments. This was extended by [BO1] and [Br] to provide an $O(p\log n+k)$ time, where there are k pairwise intersections, and $O(p)$ space algorithm for Problem 5 for line segments.

The algorithm of [BO1] with some minor modifications gives us a fast algorithm for simplicity testing, that is Problem 1. There are two checks to be carried out. First, we need to check whether or not there are two edges which intersect at some point which is not an endpoint. Such non-endpoint intersections are caused (a) by a vertical edge crossing a horizontal edge, (b) by two vertical edges overlapping, or (c) by two horizontal edges overlapping. The algorithm of [ShH] will detect intersections of type (a), while those of types (b) and (c) can be found by sorting the edges. Second, we need to check that each endpoint intersection is formed by a unique pair of edges. Clearly there should only be n such pairs for an n-vertex simple polygon. Hence we count the endpoint intersections. If their number exceeds n then the given polygon is not simple. The [BO1] and [Br] algorithm can be modified to perform this count, without affecting its time and space bounds, hence we obtain:

Theorem 3.1 *Given a polygon with* n *vertices it can be tested for simpleness in* $O(n \log n)$ *time and* $O(n)$ *space.*

The results for Problem 4 were extended to rectangles in [BW] who gave an $O(p\log p+k)$ time algorithm. This was improved in [E] and [Mc1] to also require only $O(p)$ space. Since it was already proved in [BW] that these were lower bounds, this results in:

Theorem 3.2 **[BW], [E], [Mc1]**
Problem 4 for rectangles can be solved in $O(p\log p)$ *time and* $O(p)$ *space. Problem 5 for rectangles can be solved in* $O(p\log p+k)$ *time and* $O(p)$ *space, where there are* k *intersecting pairs. Moreover these are both time- and space-optimal in a comparison-based model of computation.*

Observe that $k = O(n^2)$ see Figure 3.1.

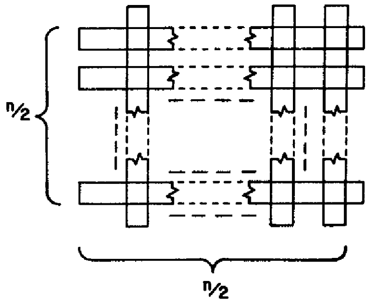


Figure 3.1

More recently time- and space-optimal divide-and-conquer algorithms for this problem have appeared, see [GüW] and [EO]. While [GBT] have a new approach based on resolution.

These results can be extended to polygons fairly easily. We sketch the sweep-line approach for them.

The basic idea is to sweep, conceptually, a vertical line from the left of the polygons to the right of the polygons and, by this means, reduce a static two-dimensional problem to a dynamic one-dimensional problem. At each position of the sweep line there are polygons which it intersects, *active* polygons, polygons completely to its left, *dead* polygons, and polygons completely to its right, *inactive* polygons. The intersection of each active polygon with the sweep line results in a number of disjoint intervals, called *active intervals*. See position (b) of the sweep line in Figure 3.2. There are two crucial observations about the sweep line. First there are at most $2n$ different sets of active intervals, since the set of active intervals only changes when the sweep line meets a vertical edge of a polygon. Second two polygons intersect if and only if there is a position of the sweep line for which their corresponding active intervals intersect.

The first observation reduces a continuous sweep of the sweep line to $2n$ discrete jumps, the second reduces the two-dimensional problem to a one-dimensional one. Letting the x-projections of the vertices in ascending sorted order be known as *sweep points* we can describe the intersection algorithm as:

```
For each sweep point $x$ do
    if $x$ corresponds to a left vertical edge then
        query active intervals with the edge for intersections, and
        insert the edge to form or extend an active interval else
```

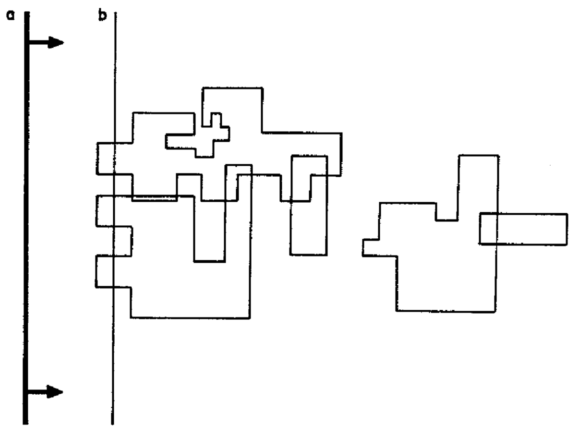


Figure 3.2

```
{x corresponds to a right vertical edge}
    delete the edge from its corresponding active interval
    yielding zero, one or two new active intervals.
```

A *left* vertical edge is one on the left side of the polygon, while a *right* vertical edge is on the right side.

The dynamic one-dimensional problem can be solved with either a *segment tree*, requiring O($n\log n$) space, or a *1-fold interval tree* requiring O(n) space. We choose the former, since it has many other applications, a description of the latter can be found in [E] and [Mc1]. The segment tree is a binary leaf search tree, that is keys are to be found at external nodes while internal nodes contain *signposts*, *separating keys*, or *routers* as they are variously called. Moreover each external node not only represents a point in the key space, but it also represents the interval in the key space defined by the successor key in the tree. This implies that each internal node can also be considered to represent a key-space interval, namely the interval determined by its external nodes. So each interval specified by a pair of keys is associated with a unique node of the tree. In our setting the keys are y-values corresponding to the endpoints of vertical edges. In Figure 3.3 there are 7 vertical edges giving 7 y-values. The corresponding empty segment tree is displayed in Figure 3.4.

External node y_i represents the closed-open interval $[y_i, y_{i+1})$ for $0 \leq i < 6$ and y_6 represents the closed interval $[y_6, y_6]$. Therefore the interval of node u, $int(u)$, is the interval $[y_0, y_4)$. To each node u is associated a set of intervals $cover(u)$; each interval I in it satisfies $int(u) \subseteq I$

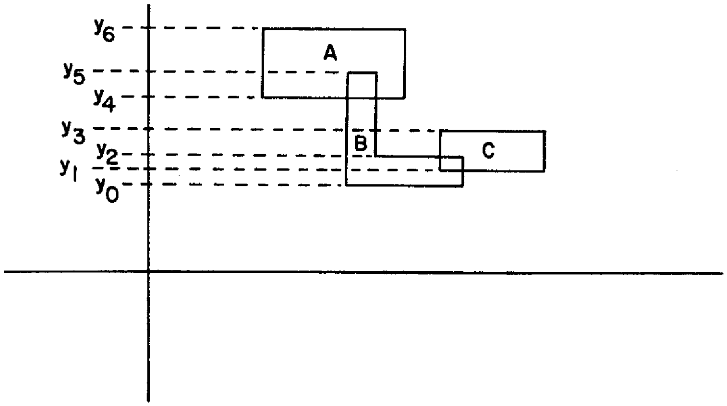


Figure 3.3

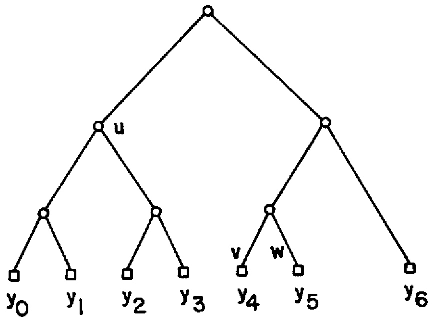


Figure 3.4

and $int(\pi u) \not\subseteq I$, where πu is the parent of u. Initially $cover(u) = \varnothing$, for all nodes u in the tree. In contrast to binary search trees the structure does not change during insertions and deletions, only the cover sets of the nodes change. Hence we can construct the segment tree to be a minimal height tree initially. For example inserting the edge $[y_0, y_5]$ of B into the segment tree of Figure 3.4 causes $cover(u) := \{B\}$, $cover(v) := \{B\}$ and $cover(w) := \{B\}$. The latter assignment does not satisfy the rule given above since $int(w) = [y_5, y_6)$. However because $int(w) \cap [y_0, y_5] \neq \varnothing$ and w is an external node we assign B to $cover(w)$ anyway. If a segment tree has n external nodes, then it has height bounded by $\lceil \log_2 n \rceil$ and it is easily seen that the insertion of an interval only affects the cover sets of at most

$4\lceil \log_2 n \rceil$ nodes. Deletion is not quite as simple, see [BW] for details, but it can still be carried out in $O(\log n)$ time.

Disregarding the technical details of maintaining the active intervals of each polygon in the segment tree, consider a query. A query edge can intersect an active interval in one of four ways, see Figure 3.5.

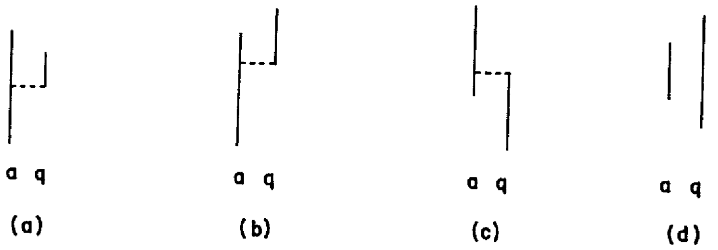


Figure 3.5

Intersections of types (a) and (b) can be found by searching the segment tree for the external node corresponding to the y-value of the bottom endpoint of the query edge. An intersection of type (c) can be found by a similar query with the top endpoint y-value. In both cases all cover sets on the search path intersect the query edge. Now intersections of type (d) are not found by either of these queries. For this purpose we specify an additional field $downcover(u)$ for each internal node u in the segment tree. This new field contains all intervals inserted in u's subtree but not at u, that is $downcover(u) = \bigcup cover(v)$, for all proper descendants v of u. Note that $downcover(u) = downcover(\Lambda u) \cup downcover(\rho u) \cup cover(\lambda u) \cup cover(\rho u)$ for all internal nodes u, where $downcover(u)$ for u an external node is defined to be $\emptyset$. Maintaining $downcover$ sets is no more difficult than maintaining cover sets, since an interval is inserted in at most $O(\log n)$ $downcover$ sets. Hence insertion and deletion complexity is not affected. But an intersection of type (d) is now easily found. Perform a dummy insertion of the query edge. Now for each $cover(u)$ to which it would be added all intervals in the corresponding $downcover(u)$ intersect the query edge. Conversely every intersection of type (d) must be found in this way. Hence all pairwise intersections will be found and moreover each update requires $O(\log n)$ time and each query $O(\log n + k')$ time, where k' is the number of pairwise intersections found. Using the 1-fold interval tree rather than the segment tree gives an optimal space bound, thus we have:

Theorem 3.3 *Given p polygons with n vertices all k pairwise intersections can be found in $O(n \log n + k)$ time and $O(n)$ space, thus solving Problem 5.*

We now consider Problems 6 and 7. Again we first consider the case for p rectangles. Now for the p-intersection case we have:

Theorem 3.4 *Given p rectangles, there is a p-intersection, that is they all have a point in common, if and only if every pair of rectangles have a point in common.*

Proof: Clearly if there is a point common to all rectangles then every pair have a common point. Conversely, assume every pair have a common point. We argue by contradiction. If the p rectangles do not have a p-intersection, then there is a maximal k such that there is a k-intersection, where $2 \leq k < p$. Let $R_1, \ldots, R_k$ be the k rectangles that have a k-intersection and R be one of the remaining rectangles. Consider a common point L of $R_1, \ldots, R_k$ which is leftmost. Now L must lie on the leftmost edge of one of the $R_1, \ldots, R_k$, let it be without loss of generality R_1, otherwise there would be another common point to its left. If R is to the left of L then R does not intersect R_1 - a contradiction. But a similar contradiction is obtained for each of the other three possibilities. Thus $R_1, \ldots, R_p$ have a p-intersection. □

This rectangle version of Helly's Theorem leads to the following stratagem for determining the existence of a p-intersection:

Determine if there is a pairwise non-intersection.

But this can be done by, essentially, the same algorithm which finds all pairwise intersections - the query at a sweeping point becomes: Is there an active segment which does not intersect the incoming segment?

An alternative approach is simply to count the number of pairwise intersections and if it is equal to $\binom{p}{2}$ then there is a p-intersection. Otherwise there is at least one pairwise non-intersection. Either way we obtain:

Theorem 3.5 *Given p rectangles the existence of a p-intersection can be determined in $O(p\log p)$ time and $O(p)$ space.*

Unfortunately Theorem 3.4 does not hold for polygons, see Figure 3.6. Indeed Figure 3.6 displays p *convex* polygons with $\binom{p}{2}$ intersections but no p-intersection, for $p \geq 3$. However for convex polygons Helly's Theorem holds directly, that is:

Theorem 3.6 *Given p convex polygons there is a p-intersection if and only if every three polygons have a common point.*

Although such a characterization does not hold in general we are still able to prove:

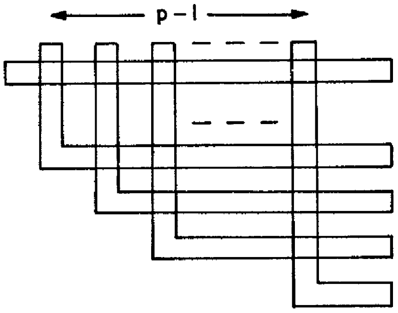


Figure 3.6

Theorem 3.7 *Given p polygons and an integer k then the existence of a k-intersection can be determined in* $O(n\log n)$ *time and* $O(n)$ *space, where the p polygons have n vertices.*

Proof Sketch: We use the sweep-line and segment tree approach described above to solve Problem 5. In other words we sweep the polygons from left to right so that at each position of the sweep line we have for each active polygon the disjoint intervals it forms with respect to the sweep line, that is the active intervals. Now there is a k-intersection of polygons if and only if at some position of the sweep line there is a k-intersection of active intervals. We represent the active intervals in a *counting segment tree*, that is $cover(u)$ for each node u is replaced by its size, $\#cover(u)$. Moreover for each node u in the tree define a new field $thickness(u)$ to be the maximal k-intersection in the subtree rooted at u. Observe that $thickness(u) = max(thickness(\lambda u),\ thickness(\rho u)) + \#cover(u)$, hence the thickness fields are easily updated after insertion and deletion. But this means that there is a k-intersection of active intervals exactly when $thickness(root) \geq k$. Moreover only insertions increase this value, so after each insertion $thickness(root)$ is checked. □

Note that the above proof sketch also yields a solution to Problem 7, therefore we have:

Theorem 3.8 *Given p polygons with n vertices their thickness can be determined in* $O(n\log n)$ *time and* $O(n)$ *space.*

4. CONVEXITY PROBLEMS

We have already defined the generally accepted notion of convexity for isothetic polygons. In this section we consider convexity testing, that is Problem 2, how the convex hull of a polygon is computed, that is Problem 3, how we might define the convex hull of a collection of polygons, and the convex skull problem.

Solving Problem 2 reduces to testing whether or not a given polygon can be decomposed into at least 2 and at most 4 staircases, which fit together appropriately. Ignoring the degenerate cases when they are only 2 or 3 staircases, observe that four staircases form a convex polygon if every staircase has the same orientation when they are rotated to place them in the north-west corner. This observation also holds, essentially, for the 2 and 3 case too. Thus testing a polygon for convexity reduces to testing if its north-west curve is a staircase of the form displayed in Figure 2.2(a). Since the polygon is specified by the sequence of its vertices in clockwise order, this reduces to five linear scans of them. The first determines the, at most, four curves, the second decides if the north-west curve is a staircase, the third tests the north-east curve, and so on, yielding:

Theorem 4.1 *Let P be a polygon with n vertices, then P can be tested for convexity in $O(n)$ time and space. Moreover both these bounds are optimal.*

Now solving Problem 3, that is computing the convex hull of a polygon, is similar to testing it for convexity. The only difference is that a smallest enclosing staircase for each curve is computed. This is straightforwardly computed, see [MF], [NLLW] and [OSW1], yielding:

Theorem 4.2 *Let P be a polygon with n vertices, then the convex hull of P can be computed in $O(n)$ time and space, which is optimal.*

The convex hull of one polygon is easily defined, but when given a collection of polygons it is not even clear what definition should be used.

Thus we have:

Problem 8 *Given a collection of polygons define their convex hull.*

To begin to understand what is meant by the convex hull of a collection of polygons, we first make precise the notion of a *convex collection*. A collection C of polygons is convex if its intersection with every line is either empty or a line segment, in other words it is the same definition as is used for individual polygons. We now discuss three reasonable definitions of the convex hull of a collection of polygons.

Definition 4.1 Minimal Collection
Given a collection C of polygons its convex hull is the *minimal area convex collection* of disjoint polygons containing the given collection.

By the *area* of a collection we mean the area of its union, and by *containment* we mean a collection C_1 contains a collection C_2 if every polygon in C_2 is contained in the union of the polygons in C_1.

Definition 4.2 Minimal Polygon
Given a collection C of polygons its convex hull is a *minimal area convex polygon* P containing the given collection.

Definition 4.3 Half Plane
Given a collection C of polygons its convex hull is the intersection of all *closed isothetic half-planes* containing the collection.

There are four isothetic closed half-planes as displayed in Figure 4.1.

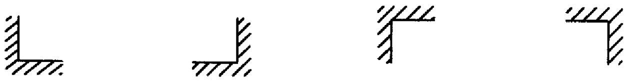


Figure 4.1

Definition 4.1 is the isothetic version of the classical definition of the convex hull and is used in [OSW2]. Definition 4.2 is the one used in [MF] and [NLLW], while Definition 4.3 corresponds to finding the maximal vectors in a set of vectors [KuLP]. We call these the *mc*-, the *mp*-, and the *hp-convex* hulls, respectively, and for a collection C we denote them by $MC(C)$, $MP(C)$ and $HP(C)$, respectively. It is not difficult to show that:

$$MC(C) \subseteq HP(C) \subseteq MP(C)$$

and, moreover, in many cases these containments are proper. See Figure 4.2, for example:

Figure 4.2(a) displays a collection of three rectangles and its *mc*-convex hull! Figure 4.2(b) and (c) show its *hp*-convex hull and an *mp*-convex hull, respectively.

For the usual non-isothetic convex hull of a point set there are a number of equivalent formulations, viz:

(1) The convex hull of a set is the intersection of all convex sets containing the given set.

(2) The convex hull of a set is the intersection of all half-planes containing the given set.

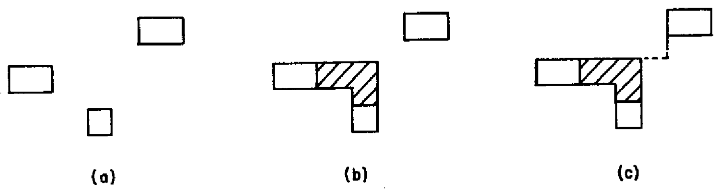


Figure 4.2

(3) A point is in the convex hull of a set if and only if it is in the convex hull of some three or fewer points from the given set.

(4) A point is in the convex hull of a set if and only if every line through the point divides the set into two non-empty sets.

Now (1) only holds for *mc*-convex hulls, (2) only holds for *hp*-convex hulls, when isothetic half-planes are meant, (3) only holds for *hp*- and *mc*-convex hulls, and (4) only holds for none of them, when isothetic lines are used.

For the usual non-isothetic convex hull we also require:

(5) The convex hull of a set to be *unique*.

(6) The convex hull of a set to be *stable* with respect to insertion and deletion of points.

Now (5) holds for both *hp*- and *mc*-convex hulls, but not for *mp*-convex hulls, see Figure 4.3. It is possible that there are infinitely many *mp*-convex hulls of a given collection, indeed Figure 4.3 only displays 3 possible *mp*-convex hulls for a collection of two rectangles. However from the point of view of stability *hp*- and *mp*-convex hulls are both stable, but *mc*-convex hulls are not, see Figure 4.4. In Figure 4.4(a) a collection and its *mc*-convex hull are displayed, while in Figure 4.4(b) a rectangle is added causing the *mc*-convex hull to grow tremendously.

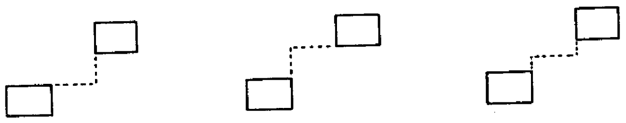


Figure 4.3

Each of these convex hulls can be computed in O($n$log$n$) time and

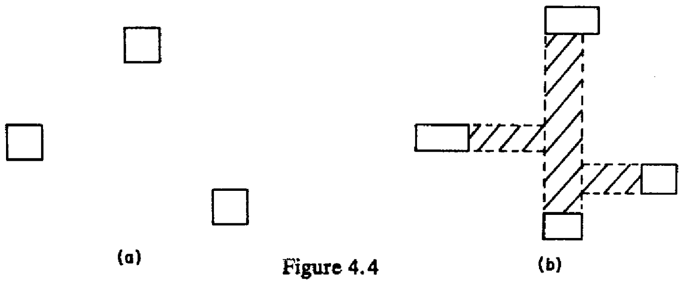


Figure 4.4

O(n) space, where the collection contains p polygons with a total of n vertices. Moreover for the case of the *hp*-convex hull (and probably for the others) this can be improved to O($n$log$h$) time, where h is the number of vertices in the *hp*-convex hull. To summarize:

Theorem 4.3 [OSW1], [KS]
Let C be a collection of p polygons with a total of n vertices. Then $MC(C)$ and $MP(C)$ can be computed in O($n$log$n$) *time and* O(n) *space, while $HP(C)$ can be computed in* O($n$log$h$) *time and* O(n) *space, where h is the number of vertices on the hp-convex hull.*

One interesting question for the *mp*-convex hull of a collection C is: Is $MP(C)$ unique? We have:

Theorem 4.4 [OSW1]
Let C be a collection of p polygons with n vertices. Then whether or not $MP(C)$ is unique can be decided in O($n$log$n$) *time and* O(n) *space.*

Finally, the stability of the *hp*-convex hull means that *hp*-convex hulls can be maintained, efficiently, under both deletions and insertions.

Theorem 4.5 [OvL]
Let C be a collection of p polygons with n vertices. Then $HP(C)$ can be maintained in O(nmlog^2n) *time for each deletion or insertion of a polygon with m vertices.*

We now turn to the *convex skull*, which is in one sense a close relative of the convex hull. We have:

Problem 9 The Convex Skull Problem
Given a polygon P find a largest area convex polygon Q with $Q \subseteq P$. Q is called a convex skull of P.

This problem is also known as the *potato peeling* problem, namely peel the smallest area from P to leave a convex polygon [Go]. See Figure 4.5 for an example.

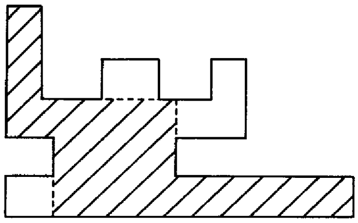


Figure 4.5

In [WoY] it is proved that:

Theorem 4.6 *A convex skull of a polygon with n vertices can be found in* $O(n^3 \log n)$ *time and* $O(n)$ *space.*

A simpler problem is:

Problem 10 *Given a polygon P find a largest area rectangle R with $R \subsetneq P$.*

In this case we find:

Theorem 4.7 *A largest-area rectangle contained in a polygon with n vertices can be found in* $O(n^2)$ *time and* $O(n \log n)$ *space.*

Proof sketch: Each side of a largest-area rectangle with the given polygon must touch one edge, at least, of the polygon. For otherwise it could be enlarged. See Figure 4.6.

Hence, using the sweep line paradigm, sweep from left to right keeping track of possible largest-area rectangles. At each position of the sweep line there are at most n candidate rectangles; these can be represented in a segment tree. If the sweep point corresponds to a left vertical edge, then it starts a new candidate rectangle. If it corresponds to a right vertical edge then it terminates some candidates and restricts others, see Figure 4.7. Since there may be as many as n restrictions at each sweep point, see Figure 4.7, we obtain the result. □

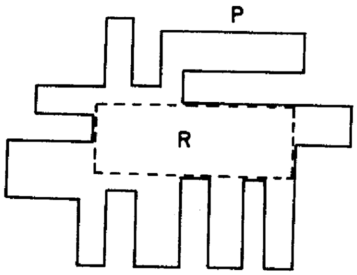


Figure 4.6

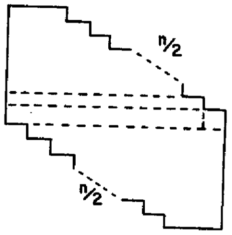


Figure 4.7

5. COMBINATIONAL PROBLEMS

By combinational we mean that the polygons are combined in some manner and, possibly, some properties of their combination are computed. For example the measure problem, that is finding the area of the plane covered by a collection, is one of the earliest of these problems, see [B] and [vLW]. We consider three combinational problems for a collection of polygons: the contour problem, the hidden line problem, and the boolean masking problem.

Problem 11 *Given p polygons with a total of n vertices, report the disjoint polygons obtained as their union; that is their* contour. See Figure 5.1.

Clearly the resulting polygon may have $O(n^2)$ edges, see Figure 3.1, for example. [LipP] gave the first algorithm for this problem, when the polygons are, in fact, rectangles. Their general approach is found in the subsequent work of [Gü1], [Gü2], and [Wo]. [LipP] observed that it is only necessary to compute the horizontal edges of the resulting polygons, since the

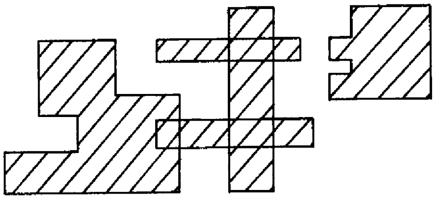


Figure 5.1

vertical edges may then be deduced from them. This gives a two-phase algorithm:

Phase 1: Compute the horizontal edges of the contour.

Phase 2: Compute the complete contour.

Phase 2 requires the horizontal edges to be sorted so for this reason it has a worst-case behavior of $O(n^2 \log n)$ or, more precisely, $O(r \log r)$ if there are r edges resulting from Phase 1. However Phase 1 can be solved in $O(n \log n + r)$ time and $O(n)$ space, as demonstrated in [Gü1], [Gü2], and [Wo]. In [Wo] a solution is given, based on the segment tree, which solves Problem 11 in full generality, not just for rectangles. The results proved in [Wo] are:

Theorem 5.1 *Given p polygons with a total of n vertices. Then the r horizontal edges of their contour can be computed in $O(n \log n + r)$ time and $O(n)$ space. Moreover these bounds are optimal.*

We now turn to our second problem:

Problem 12 The Boolean Masking Problem
Given p_r disjoint red polygons with n_r vertices and p_g disjoint green polygons, with n_g vertices, report the polygons formed by their:

(a) OR - *the union of the red and green polygons,*
(b) AND - *the intersection of the red polygons with the green polygons,*
(c) XOR - *their OR minus their AND,*
(d) NAND - *the intersection of the non-red portion of the plane with the green polygons.*

These operations are fundamental in mask design checking, see [La] and [MeC], where the red and green polygons correspond to polygons on two different layers.

Clearly Problem 12 is a special case of the contour problem, so Theorem 5.1 provides upper bounds for its solution where $p = p_r + p_g$ and $n = n_r + n_g$. Indeed this is the result in [Pr], which is also implied in [OWW2], namely:

Theorem 5.2 *Given a collection of $P = p_r + p_g$ polygons with $n = n_r + n_g$ vertices, as above, their OR, AND, XOR, and NAND can be computed in* $O(n\log n + r)$ *time and* $O(n+r)$ *space, where r is the number of vertices of the resulting collection.*

Finally we turn to:

Problem 13 The Hidden Line Problem
Given p opaque polyhedra in 3-space with n vertices and a given isothetic viewing direction, report the polygonal faces, and portions thereof, that can be seen from the viewing direction.

In [OWW1] a hidden line elimination algorithm is presented which deals with collections of arbitrary polyhedra viewed from an arbitrary direction. The restrictions to isothetic polyhedra and isothetic views leads immediately to:

Theorem 5.3 *Given a collection of opaque isothetic polyhedra with n vertices and given an isothetic direction, then the two-dimensional view from this direction can be computed in* $O(n\log^2 n + r)$ *time and* $O(n\log n)$ *space, where r is the number of vertices in the resulting view.*

In [S2] the hidden line problem for a single polygon of n vertices is solved in linear time.

6. CLUSTERING PROBLEMS

We discuss three techniques for partitioning a collection of polygons into clusters. The first, and most obvious, is based on intersections and connectivity, that is connected components. The second is based on minimal area convex hulls, while the third is based on diagonal closure. The latter clustering technique occurs when viewing, from a geometrical point of view, deadlock and safety problems for locked transaction systems [Pa] and [LipP]. The second technique has been proposed as a method of partitioning components in a chip layout into macro-components, which are then more easily manipulated.

Problem 14 Connected Components Problem
Given a collection of polygons with n vertices determine their connnected components.

Given a collection of polygons, a subcollection is connected if its union is a connected set of points. A *connected component* of the collection is a subcollection which is connected and is also disjoint from the remaining polygons.

The connected components are the equivalence classes formed by taking the transitive closure of the pairwise intersection relation. Thus they can be found as a spin-off from Theorem 3.2. However this implies that a total of $O(n \log n + k \log k)$ time is required using the standard UNION-FIND method [AHU], where k is the number of pairwise intersections. Since $k = O(n^2)$ this method produces an inefficient algorithm. Fortunately a better approach is possible, as shown in [EvLOW] and [GuS].

Use the sweep line approach as we did in solving Problem 5, see Section 2. This gives rise to the notion of a *left-connected component*, with respect to the sweep line, that is a connected subcollection of the collection, where each polygon in the component is either dead or active. Clearly the connected components are the left components associated with the rightmost position of the sweep line. A crucial observation is that not only does the intersection of the sweep line with each left-connected component give at least one line segment, but also for two or more components these can be, at most, nested, they cannot overlap. In Figure 6.1(a) a valid configuration is displayed showing the nesting of two components A and B, while C is completely disjoint from both B and C.

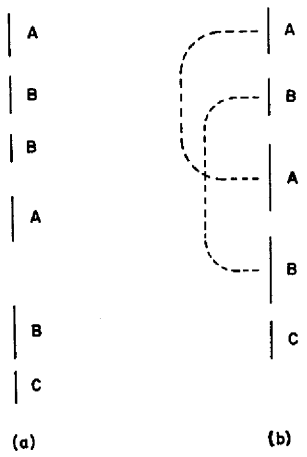


Figure 6.1

Figure 6.1(b) displays an impossible situation since the two appearances of B must be left connected as are the two appearances of A, hence A and B must intersect; a contradiction.

Since the one-dimensional projection of the left-connected components on the sweep line is so highly structured, it leads to an efficient algorithm to find connected components.

A high level algorithm can now be given:

For each sweep point x:

(1)x corresponds to a left edge.
Determine the left-connected components it joins, belongs to, or initiates. Modify left-connected components appropriately, forming unions where necessary.

(2)x corresponds to a right edge.
Determine the left connected component it belongs to and modify it appropriately.

Since we are forming equivalence classes, underlying this algorithm we have a classical UNION-FIND problem, since we need in both Steps (1) and (2) to perform FINDs, while in Step (1) we also need to perform UNION. It is sufficient to use any one of the methods of solving UNION-FIND problems which requires $O(n\log n)$ time for $O(n)$ operations. Beyond this we need to represent each left-connected component as disjoint line segments, be able to insert and delete edges from them, and given an edge query determine which left-connected components it intersects. It turns out to be important to carry out the query in at most two phases. In the first phase it only reports the left-connected components it possibly intersects. For this purpose only the current topmost and bottommost points of each left-connected component are used. The result of the query is either it overlaps some of these end-points or it is enclosed by some pair of endpoints. In the former case this immediately causes UNIONs to be carried out, while in the latter case the smallest enclosing pair should be investigated further to determine if one UNION should be carried out.

Data structures can be designed which yield the following result, generalizing the result for rectangles in [EvLOW] and [GuS]:

Theorem 6.1 *Given a collection of polygons with* n *vertices, its connected components can be found in* $O(n\log n)$ *time and* $O(n)$ *space.*

We now consider:

Problem 15 *Determine the minimal area convex hull of a collection* C *of polygons, that is* $MC(C)$.

As pointed out in Section 4 $MC(C)$ may consist of two or more disjoint convex polygons, each of which contains at least one of the original polygons. Now $MC(C) \supseteq \{MC(P) : P \text{ in } C\}$, and this collection can be found in time linear in the number of vertices. Therefore, without any loss of generality, we may assume that C is, in fact, a collection of convex polygons.

In [OSW2] the following theorem is proved:

Theorem 6.2 *Given a collection* C *of polygons with a total of* n *vertices,* $MC(C)$ *can be computed in* $O(n\log n)$ *time and* $O(n)$ *space.*

The algorithm to compute $MC(C)$ is very similar to that for computing connected components, so we omit any further details.

Finally we turn to the notion of diagonal closure. Given a collection C of polygons we say it is *north-east closed* if for every pair of points $p_1 = (x_1,y_1)$ and $p_2 = (x_2,y_2)$ satisfying (1) and (2) below, the point (x_2,y_1) is in C, where:

(1) p_1 and p_2 are in the same connected component, and

(2) $x_1 < x_2$ and $y_1 > y_2$.

See Figure 6.2.

Similarly we say C is *south-west closed* if the point (x_1,y_2) is in C, and C is *diagonally-closed* if it is north-east and south-west closed.

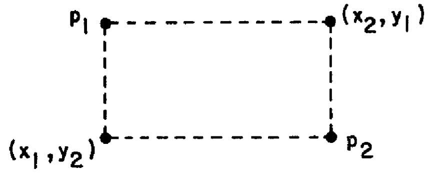


Figure 6.2

Given a collection C of polygons we define its *north-east closure* as the minimal area collection C' of polygons, with $C' \supseteq C$ and C' is north-east closed. Again we obtain *south-west closure* and *diagonal closure*. For a collection C we denote these closures by $NE(C)$, $SW(C)$, and $D(C)$ respectively.

Problem 16 *Given a collection C of polygons determine NE(C), SW(C), and D(C).*

In [SSW] the following theorem is proved:

Theorem 6.3 *Let C be a collection of polygons. Then* $D(C) = NE(SW(C)) = SW(NE(C))$.

Thus determining $D(C)$ reduces to determining the north-east and south-west closure of a collection. Let $rotate(C, \theta)$ denote the rotation of a collection C by θ degrees about the origin, hence $C = rotate(C, 2\pi)$. Then we obtain:

Theorem 6.4 *Let C be a collection of polygons. Then* $NE(C) = rotate(SW(rotate(C,\pi)),\pi)$ *and* $SW(C) = rotate(NE(rotate(C,\pi)),\pi)$.

Thus we only need to determine north-east closure in order to be able to determine diagonal closure. In [SSW] it is shown that $NE(C)$ can be computed in $O(n \log n)$ time and space for a collection C of polygons having n vertices. The technique use is based once more on the sweep line paradigm. It is also similar to the connected components algorithm, needing some extra work, since regions need to be added to the original polygons. Thus:

Theorem 6.5 *Given a collection C of polygons with n vertices* $NE(C)$, $SW(C)$, *and* $D(C)$ *can be computed in* $O(n \log n)$ *time and space.*

The importance of diagonal closure stems from the work of [Pa] on locked transaction systems. Given two transactions, each of which can be considered to consist of a sequence of *lock* and *unlock* operations on at most

n variables, we wish to determine whether they are deadlock-free and safe, when executed in a concurrent environment. Associating the two transaction sequences with the sequence of points 1, ... on the x- and y-axes, respectively, then they define rectangular forbidden regions. For example letting T_1 = *lock y*, *lock x*, *unlock x*, *unlock y*, and T_2 = *lock x*, *lock y*, *unlock y*, *unlock x* we obtain Figure 6.3.

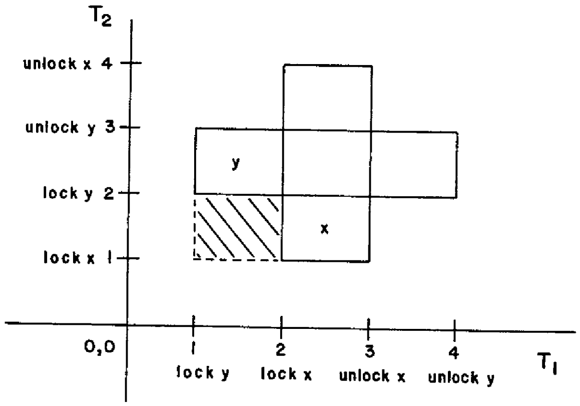


Figure 6.3

The region labelled x corresponds exactly to the restriction that variable x may only be locked by one transaction at a time. A staircase from 0,0 to 4,4 which does not pass through any rectangle corresponds to a possible interleaving of T_1 and T_2, and vice versa, that is a *computation history*.

Consider the point (2,2). A staircase which approaches close to this point (it cannot reach it since this would imply both T_1 and T_2 are locking variable y) cannot reach point (4, 4) at all, since it must go up or go right. The former is not possible since this requires T_2 to lock y also, and the latter is not possible since this requires T_1 to lock x also. In other words T_1 and T_2 are *deadlocked*. The rectangular area determined by (1, 1) and (2, 2) is a region of *deadlock*. Note that the diagonal closure of the two rectangles fills in this region, the shaded region in Figure 6.3. To characterize deadlock we refer to the left edges of a collection as the left edges of its union, and to the lower edges as the lower edges of its union. Basically these edges, in the diagonal closure, must be in the original collection for deadlock freedom to occur.

Similarly, the notion of the *safety* of T_1 and T_2 can be characterized as the absence of any staircase which separates the forbidding rectangles into two non-empty sub-collections. Indeed [Pa] proves:

Theorem 6.6 *Let C be the collection of forbidding rectangles corresponding to two transactions T_1 and T_2 of n variables. Then T_1 and T_2 are* safe *if and only if $D(C)$ consists of a single polygon and T_1 and T_2 are* deadlock-free *if and only if the lower and left edges of $D(C)$ are (parts of) lower and left edges of C.*

7. VISIBILITY PROBLEMS

There are a number of visibility issues for polygons which either arise from some practical problem or arise from considerations of the complexity of polygons.

Examples of the latter situation are the various art gallery theorems.

Problem 17 *Given a polygon place a number of guards or watchmen on vertices, on edges, in the interior, or even in the exterior, so that each point on an edge, in the interior, in the exterior, etc., can be seen by at least one watchman.*

An elegant proof of one result of this type is:

Theorem 7.1 **[O'R1]**
$\lfloor n/4 \rfloor$ guards, stationed on vertices, are necessary and sufficient to see every point in the interior of a polygon of n edges.

One can also assume that the guards have some mobility, for example along a line segment. This has been investigated by [O'R2]. Other variations, which have not been investigated to the best of my knowledge, are only allowing guards isothetic mobility and isothetic sight. Assuming the walls are reflective is another possible assumption.

However we prefer to consider visibility problems arising from more practical situations. To this end we define:

Problem 18 *Given a collection of polygons we define a symmetric relation* vis, *for "is visible to," by: For two polygons A and D, A* vis *D if and only if there is a horizontal or vertical line segment joining A to D which intersects no other polygon, see Figure 7.1.*

If we wish to connect components A and D with a wire, then when $A\, vis\, D$ we have a straight connection, whereas if *not* $A\, vis\, D$ then a connecting wire needs to be routed around the blocking obstacles.

It is convenient to split the *vis* relation into its horizontal and vertical components. Hence we say $A\, hvis\, D$ if $A\, vis\, D$ via a horizontal line segment and $A\, vvis\, D$ if $A\, vis\, D$ via a vertical line segment. In general we might have both $A\, hvis\, D$ and $A\, vvis\, D$, since the given polygons are not necessarily

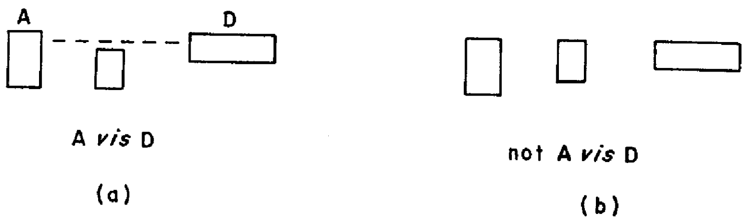


Figure 7.1

convex. Note that *vis* may be the empty relation, see Figure 7.2.

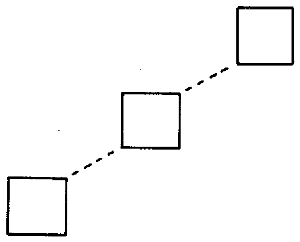


Figure 7.2

We first prove a combinatorial result concerning the size of *vis*, namely the maximum number of visibility pairs among n polygons.

Theorem 7.2 *Given a collection of* p *disjoint convex polygons, then* hvis *(and* vvis*) has size at most* $3p$ *and, therefore,* vis *has size at most* $6p$.

Proof: Consider *hvis* only. Each polygon has a bottommost and topmost edge. For every polygon, extend each of these edges to the right and left until they meet another polygon, otherwise they become rays or lines. In all three cases a partition of the plane is obtained, see Figure 7.3. Intuitively this process gives the horizontal *visibility regions* for each polygon. Now observe that each bottommost (topmost) edge of a polygon with its extension abuts three visibility regions as in Figure 7.4. This follows from the disjointness assumption and the construction of the regions. Each polygon together with its extension determines one horizontal edge for each of 6 regions. But each region, apart from the extremal ones, has two horizontal edges. Therefore there are, at most, 6p/2 regions, that is 3p regions. □

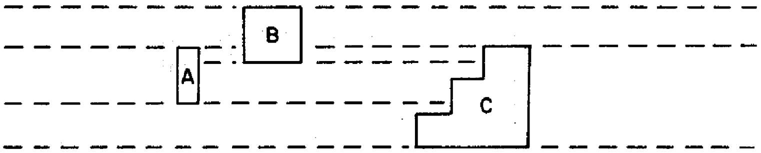


Figure 7.3

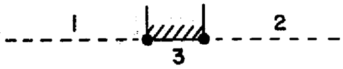


Figure 7.4

This notion was used in [GJS] for printed circuit testing. In their application they have curves rather than polygons. Their problem is to determine a clustering of curves into disjoint groups, so that it is sufficient to test groups against groups for fabrication faults leading to short circuits. They show that there are at most 12 groups needed, in general, independent of the number of curves. In this setting a curve corresponds to a wiring and a visibility corresponds to the possibility of a short circuit. This is one application of the local relation *vis* to a global problem. There are bound to be others. For example a solution to the contour problem, see Section 5, depends on local visibility.

8. DISCUSSION

It is to be hoped that the survey presented here, although a biased one, will convince the reader that the isothetic viewpoint is a rich source of applications and problems. There are of course topics which have been omitted. Two of them, especially deserving of mention are: *motion* problems and *decomposition* problems. Elsewhere in this volume Sue Whitesides and Godfried Toussaint discuss motion problems in the context of robotics, see also [ToS]. Given a collection of polygons, some of which are fixed and are called *obstacles*, plan a sequence of movements to move the *movable* polygons to some given new positions, so that no two polygons collide and no two polygons touch. In [COSW] all polygons are movable and the aim is to separate them all, giving the game SEPARATION^{TM}.

Two examples of decomposition problems are to decompose the union of a collection of polygons into a minimal number of disjoint rectangles and into disjoint rectangles with minimal contour length. The first decomposition method occurs as a problem in databases [LipLLMP], [LLMP], the second as a problem in wire routing on VLSI chips [LiPRS].

Research into isothetic problems has: yielded new data structures, for example the segment tree, the 1-fold interval tree, and the priority search tree; resulted in new paradigms, for example the sweep line, dynamization, and paradigmatic transformations; and re-affirmed the utility of established paradigms, for example divide-and-conquer and dynamic programming.

Acknowledgements

Many of the results quoted in this survey stem from collaborative research in which I have been involved. The following are no longer just collaborators, but are also friends, without whom this survey could not have been written. I thank Jon Bentley, Bernard Chazelle, Herbert Edelsbrunner, Ralf Hartmut Güting, Thomas Ottmann, Eljas Soisalon-Soininen, Jan van Leeuwen, Peter Widmayer, and Chee Yap. Also Karel Culik II is responsible for the present form of Theorem 7.2, improving my own bound by p.

References

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

[B] Bentley, J.L., Algorithms for Klee's Rectangle Problems. Unpublished (1977).

[BKST] Bentley, J.L., Kung, H.T., Schkolnick, M., and Thompson, C.D., On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of the ACM 25* (1978), 536-543.

[BO1] Bentley, J.L., and Ottmann, Th., Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on Computers C-28* (1979), 643-647.

[BO2] Bentley, J.L., and Ottmann, Th., The Complexity of Manipulating Hierarchically Defined Sets of Rectangles. *Proceedings of the 10th International Symposium on Mathematical Foundations of Computer Science; Lecture Notes in Computer Science 118* (1981), Springer Verlag, 1-15.

[BW] Bentley, J.L., and Wood, D., An Optimal Worst Case Algorithm for Reporting Intersections on Rectangles. *IEEE Transactions on Computers C-29* (1980), 563-580.

[COSW] Chazelle, B., Ottmann, Th., Soisalon-Soininen, E., and Wood, D., The Complexity and Decidability of SEPARATION^{TM}. University of Waterloo, Computer Science Technical Report CS-

83-34 (1983).

[Br] Brown, K.O. Comments on 'Algorithms for Reporting and Counting Geometric Intersections.' *IEEE Transactions on Computers C-30* (1981), 147-148.

[E1] Edelsbrunner, H., A Time- and Space-Optimal Solution for the Planar All Intersecting Rectangles Problem. Technical University of Graz, Graz, Austria, Institute fuer Informatinsverarbeitung, Report F50 (1980).

[EO] Edelsbrunner, H., and Overmars, M.H., Batched Dynamic Solutions to Decomposable Searching Problems. TU Graz IIG Technical Report No. F118 (1983).

[EvLOW] Edelsbrunner, H., van Leeuwen, J., Ottmann, T.A., and Wood, D., Connected Components of Orthogonal Geometric Objects. *RAIRO* (1983), to appear.

[FSS] Ferrari, L., Sankar, P.V., and Sklansky, J., Minimal Rectangular Partitions of Digitized Blobs. *Proceedings of the 5th International Conference on Pattern Recognition,* Miami Beach (1980), 1040-1043.

[FPT] Fowler, R.J., Paterson, M.S., and Tanimoto, S.L., Optimal Packing and Covering in the Plane are NP-Complete. *Information Processing Letters 12* (1981), 133-137.

[GBT] Gabow, H., Bentley, J.L., and Tarjan, R.A., Geometric Scaling. In preparation (1983).

[GJS] Garey, M.R., Johnson, D.S., and So, H.C., An Application of Graph Coloring to Printed Circuit Testing. *IEEE Transactions on Circuits and Systems CAS-23* (1976), 591-599.

[Go] Goodman, J.E., On the Largest Convex Polygon Contained in a Non-Convex n-gon or How to Peel a Potato. *Geometriae Dedicata 11* (1981), 99-106.

[GVDK] Groen, F.C.A., Verbeek, P.W., DeJong, N., and Klumper, J.W., The Smallest Box Around a Package. *Pattern Recognition 14* (1981), 173-178.

[GuS] Guibas, L., and Saxe, J., Problem 80-15. *Journal of Algorithms 4* (1983), 177-181.

[Gü1] Güting, R.H., An Optimal Contour Algorithm for Iso-Oriented Rectangles. *Journal of Algorithms* (1983), to appear.

[Gü2] Güting, R.H., Stabbing c-Oriented Polygons. *Information Processing Letters* (1983), to appear.

[Gü3] Güting, R.H., Docterol dissertation, Dortmund, 1983.

[GüW] Güting, R.H., and Wood, D., Finding Rectangle Intersections by Divide-and-Conquer. *IEEE Transactions on Computers* (1983), to appear.

[HS] Haralick, R.M., and Shapiro, L.G., Decomposition of Polygonal Shapes by Clustering. *Proceedings of the IEEE Computer Society*

Conference on Pattern Recognition and Image Processing (1977), 183-190.

[Hw] Hwang, F.K., An O($n\log n$) Algorithm for Rectilinear Minimal Spanning Tree. *Journal of the ACM 26* (1979), 177-182.

[IA] Imai, H., and Asabo, T., Finding the Connected Components and a Maximum Clique of an Intersection Graph of Rectangles in the Plane. University of Tokyo, Tokyo, Japan, Department of Mathematical Engineering and Instrumentation Physics; manuscript (1981).

[KS] Kirkpatrick, D.G., and Seidel, R., The Ultimate Planar Convex Hull Algorithm? *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing* (1982), 35-42.

[KuLP] Kung, H.T., Luccio, F., and Preparata, F.P., On Finding the Maxima of a Set of Vectors. *Journal of the ACM 22* (1975), 469-476.

[La] Lauther, U., An O($N\log N$) Algorithm for Boolean Mask Operations. *Proceedings of the 18th Design Automation Conference*, Nashville (1981), 555-562.

[LeP] Lee, D.T., and Preparata, F.P., An Improved Algorithm for the Rectangle Enclosure Problem. Submitted for publication.

[LeW] Lee, D.T., and Wong, C.K., Finding Intersections of Rectangles by Range Search. *Journal of Algorithms 2* (1981), 337-347.

[vLW] van Leeuwen, J., and Wood, D., The Measure Problem for Rectangular Ranges in d-Space. *Journal of Algorithms 2* (1981), 282-300.

[Lh] Lehert, P., Clustering by Connected Components in O(n) Expected Time. *RAIRO Informatique/Computer Science 15* (1981), 207-218.

[Li] Lingas, A., The Power of Non-Rectilinear Holes. *Proceedings of the 9th Colloquium on Automata, Languages and Programming; Lecture Notes in Computer Science 140* (1982), Springer Verlag, 369-383.

[LiPRS] Lingas, A., Pinter, R.Y., Rivest, R.L., and Shamir, A., Minimum Edge Length Partitioning of Rectilinear Polygons. *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing* (1982), 53-63.

[Lip] Lipski, W. Jr., Finding a Manhattan Path and Related Problems. University of Illinois, Urbana, Illinois, Coordinated Science Laboratory; Report T-85 (1979).

[LipLLMP] Lipski, W. Jr., Lodi, E., Luccio, F., Mugnai, C., and Pagli, L., On Two-Dimensional Data Organization II. *Fundamenta Informaticae II* (1979), 245-260.

[LipP] Lipski, W. Jr., and Papadimitriou, C.H., A Fast Algorithm for Testing for Safety and Detecting Deadlock in Locked Transaction

Systems. *Journal of Algorithms 2* (1981), 211-226.

[LipP1] Lipski, W. Jr., and Preparata, F.P., Finding the Contour of a Union of Iso-Oriented Rectangles. *Journal of Algorithms 1* (1980), 235-246.

[LipP2] Lipski, W. Jr., and Preparata, F.P., Segments, Rectangles, Contours. *Journal of Algorithms 2* (1981), 63-76.

[LLMP] Lodi, E., Luccio, F., Mugnai, C., and Pagli, L., On Two-Dimensional Data Organization I. *Fundamenta Informaticae II (1979), 211-226.*

[Mc1] McCreight, E.M., Efficient Algorithms for Enumerating Intersecting Intervals and Rectangles. XEROX Palo Alto Research Center, Palo Alto, California; Report CSL-80-9 (1980).

[Mc2] McCreight, E.M., Priority Search Trees. XEROX Palo Alto Research Center, Palo Alto, California; Report CSL-81-5 (1981).

[MeC] Mead, C., and Conway, L., *Introduction to VLSI Systems.* Addison-Wesley Publishing Co., Reading, Mass., 1980.

[MF] Montuno, D.Y., and Fournier, A., Finding the $x-y$ Convex Hull of a Set of $x-y$ Polygons. University of Toronto CSRG Technical Report 148 (1982).

[NLLW] Nicholl, T.M., Lee, D.T., Liao, Y.Z., and Wong, C.K., Constructing the X-Y Convex Hull of a Set of X-Y Polygons. *BIT* (1983), to appear.

[NP] Nievergelt, J., and Preparata, F.P., Plane-Sweep Algorithms for Intersecting Geometric Figures. *Communications of the ACM 25* (1982), 739-747.

[O'R1] O'Rourke, J., Galleries Need Fewer Mobile Guards: A Variation on Chvátal's Theorem. *Geometriae Dedicata 14* (1983), 273-283.

[O'R2] O'Rourke, J., An Alternative Proof of the Rectilinear Art Gallery Theorem. Johns Hopkins University Technical Report 82-15 (1982).

[OSW1] Ottmann, Th., Soisalon-Soininen, E., and Wood, D., On the Definition and Computation of Rectilinear Convex Hulls. *Information Sciences* (1983), to appear.

[OSW2] Ottmann, Th., Soisalon-Soininen, E., and Wood, D., Rectilinear Convex Hull Partioning of Sets of Rectilinear Polygons. University of Waterloo, Computer Science Technical Report CS-83-19 (1983).

[OWW1] Ottmann, Th., Widmayer, P., and Wood, D., A Worst-Case Efficient Algorithm for Hidden Line Elimination. University of Waterloo, Computer Science Technical Report CS-82-33 (1982).

[OWW2] Ottmann, Th., Widmayer, P., and Wood, D., A Fast Algorithm for Boolean Mask Operations. University of Waterloo, Computer Science Technical Report CS-82-37 (1982).

[Ov1] Overmars, M.H., The Design of Dynamic Data Structures. To appear as *Lecture Notes in Computer Science* (1983), Springer Verlag.

[OvL] Overmars, M.H., and van Leeuwen, J., Maintenance of Configurations in the Plane. *Journal of Computer and System Sciences 23* (1981), 166-204.

[Pa] Papadimitriou, C.H., Concurrency Control by Locking. *SIAM Journal on Computing 12* (1983), 215-226.

[Pr] Pracchi, M., Boolean Expressions of Rectilinear Polygons with VLSI Applications. University of Illinois-Urbana, Coordinated Science Laboratory Technical Report R-978 (1982).

[R] Rappaport, D., Eliminating Hidden-Lines from Monotone Slabs. *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing* (1982), 43-52.

[S1] Sack, J.R., An $O(n\log n)$ Algorithm for Decomposing Simple Rectilinear Polygons into Convex Quadrilaterals. *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing* (1982), 64-74.

[S2] Sack, J.R., A Simple Hidden-Line Algorithm for Rectilinear Polygons. *Proceedings of the 21st Allerton Conference on Communication, Control, and Computing* (1983).

[Sh] Shamos, M.I., Computatinal Geometry. Doctoral Dissertation, Yale University (1978).

[ShH] Shamos, M.I., and Hoey, D., Geometric Intersection Problems. *Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science* (1976), 208-215.

[Sk] Sklansky, J., Measuring Concavity on Rectangular Mosaic. *IEEE Transactions on Computers C-21* (1972), 1355-1364.

[SkCH] Sklansky, J., Chazin, R.L., and Hansen, B.J., Minimum Perimeter Polygons of Digitized Silhouettes. *IEEE Transactions on Computers C-21* (1972), 260-268.

[SSW] Soisalon-Soininen, E., and Wood, D., An Optimal Algorithm for Testing for Safety and Detecting Deadlocks in Locked Transaction Systems. To appear in *ACM SIGACT-SIGMOD Conference on Principles of Database Systems* (1982).

[Sr] Srihari, S.N., Representation of Three-Dimensional Digital Images. *Computer Surveys 13* (1981), 399-424.

[T] Tompa, M., An Optimal Solution to a Wire-Routing Problem. *Journal of Computer and System Sciences 23* (1981), 127-150.

[ToS] Tousaint, G., and Sack, J.R., Some New Results on Moving Polygons in the Plane. *Proceedings of the Robotic Intelligence and Productivity Conference* (1983).

[vLW] van Leeuwen, J., and Wood, D., The Measure Problem for Rectangular Ranges in d-Space. *Journal of Algorithms 2* (1981), 282-300.

[WM] Wesley, M.A., and Markowski, G., Fleshing Out Projections. Thomas J. Watson Research Center, Yorktown Heights, New York, Computer Science Department; Report RC 8884 (1981).

[Wo] Wood, D., The Contour Problem for Rectilinear Polygons. University of Waterloo, Computer Science Technical Report CS-83-20 (1983).

[WoY] Wood, D., and Yap, C., Computing the Convex Skull of an Isothetic Polygon. In preparation (1983).

[YPK] Yannakakis, M., Papadimitriou, C.H., and Kung, H.T., Locking Policies: Safety and Freedom from Deadlock. *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science* (1979), 286-297.