Polygon Area Decomposition for Multiple-Robot Workspace Division[*]

Susan Hert[†] Vladimir Lumelsky[‡]

Abstract

We present a new polygon decomposition problem, the *anchored area partition problem*, which has applications to a multiple-robot terrain-covering problem. This problem concerns dividing a given polygon $\mathcal{P}$ into n polygonal pieces, each of a specified area and each containing a certain point (site) on its boundary. We first present the algorithm for the case when $\mathcal{P}$ is convex and contains no holes. Then the generalized version that handles nonconvex and nonsimply connected polygons is presented. The algorithm uses sweep-line and divide-and-conquer techniques to construct the polygon partition. The algorithm assumes the input polygon $\mathcal{P}$ has been divided into a set of p convex pieces ($p = 1$ when $\mathcal{P}$ is convex) and runs in time $O(pn^2 + vn)$, where v is the sum of the number of vertices of the convex pieces.

Keywords – polygon decomposition, area partition, divide and conquer, sweep line, robot workspace partition, robotics, terrain covering

1 Introduction

The *polygon decomposition problem* is the problem of dividing a given polygon into a set of smaller polygons. It has been studied in a variety of forms, and has applications in a variety of fields, including VLSI design, computer graphics, cartography, pattern recognition, and databases. We propose here a new polygon decomposition problem, the *area partition problem*, which has applications in the field of robotics. We present a polynomial-time divide-and-conquer sweep-line algorithm for the solution of one version of this problem for any simple polygon.

Perhaps the most prevalent and widely applicable form of polygon decomposition is triangulation. Algorithms abound for creating triangulations of polygons with various characteristics ([3, 4, 8, 10, 11, 12], *e.g.*). Polynomial-time algorithms have also been presented for decomposing polygons into trapezoids [2], convex polygons [6, 15, 17, 19, 21, 31], star-shaped or monotone polygons [19], and rectangles [21, 23]. In [1], an algorithm is presented for decomposing a polygon into regions based on geodesic distance from a set of points, that is, for constructing the geodesic Voronoi diagram. Many types of polygon decompositions, including triangulations, convex decompositions, quadrilateralizations, and visibility polygon decompositions are considered in [26] and the works referenced there.

[*]This work is supported in part by the National Science Foundation Grant IRI-9220782 and the Sea Grant Program (U.S. Dept. of Commerce, grant NA46RG048).

[†]University of Wisconsin – Madison, Computer Sciences Department, 1210 West Dayton St., Madison, WI, 53706, USA., E-mail: hert@cs.wisc.edu

[‡]University of Wisconsin – Madison, Robotics Laboratory, 1513 University Ave. Madison, WI 53706, USA. E-mail: lumelsky@engr.wisc.edu

Two types of minimal decompositions have also been considered in the literature: (a) decompositions using the minimum number of components and (b) those with minimum total edge length. Polynomial-time algorithms have been developed, for example, for dividing rectilinear polygons into rectangles and polygons into convex polygons by drawing lines of minimal total length [21], for dissection of a rectilinear polygon with arbitrary holes into the minimum number of rectangles [30], and for division of a polygon into the minimum number of convex pieces [7]. Several other minimal decomposition problems have been shown to be NP-hard [22, 27].

Polygon decomposition is also studied in conjunction with the theory of packings, coverings, and tilings ([14, 18, 20], *e.g.*), which concerns arranging a set of polygons, each usually a copy of a single polygon or one of a small set of polygons, in such a way as to occupy as much of a given region as possible with minimal or no overlap of the polygons. Another interesting variation on the polygon decomposition problem comes from the mathematics literature concerned with dissection theory [5, 13]. There it is shown that, given any two polygons of the same area, either can be cut into a set of pieces that can be rearranged to produce the other. Such polygons are said to be *equidecomposible*.

The area partition problem, which we introduce here, is the problem of dividing a given arbitrary polygon into a number of pieces, each with a given area. Previous work on decomposing polygons using area constraints concerns the division of rectangles defined on a gird into equal-area regions. Page and Sastry [28] show that a rectangle defined on a lattice may be divided into two regions of equal area by a polygonal path with vertices on the lattice that are defined by the Fibonacci sequence. They also consider the more general problem of dividing such a rectangle into two parts of not necessarily equal area by a nondecreasing path on the lattice but provide only an approximate solution in this case. Christou and Meyer [9] present a method for partitioning a rectangle defined on a grid into equal-area rectilinear polygons of minimum (or nearly minimum) total perimeter.

The problem we consider is much more general in that it considers arbitrarily shaped simple polygons divided into two or more regions of not necessarily equal areas. As with the general polygon decomposition problem, there are many versions of the area partition problem. The version we consider here is dubbed the *anchored area partition problem*. It is motivated by the following terrain-covering problem in robotics. There are n robots $R_i, i = 1, \ldots, n$, each placed at a distinct starting point S_i on the boundary of a polygonal region $\mathcal{P}$. The robots are given the task of completely covering the given region. That is, each part of the region $\mathcal{P}$ must be visited by one of the robots, so as, for example, to vacuum a floor, mow a lawn, or simply explore a region. To do this most efficiently, the region $\mathcal{P}$ should be divided among the robots so they each do roughly the same amount of work with no overlap of the work regions.

In its assigned region, each robot will execute a terrain-covering algorithm [16, 24, 25, 29]. Two good measures of the amount of work done by a robot performing a given terrain-covering task are the length of the path it generates and the amount of time spent on the task. These values depend on the algorithm the robot executes and the characteristics of the robot and the environment. Given these characteristics, the relative capabilities of the robots can be determined based on an estimate of the area of the region each can cover in a given amount of time.

We assume that, based on the relative capabilities of the robots, it has already been determined what proportion of the area of the region $\mathcal{P}$ each robot should be assigned. These proportions are represented by a set of values $c_i, i = 1, \ldots, n$, with $0 < c_i < 1$ and $\Sigma_{i=1}^{n} c_i = 1.0$. The problem we consider is as follows: Given a polygon $\mathcal{P}$ and n points (*sites*) $S_1, \ldots, S_n$ on the polygon, divide the polygon into n nonoverlapping polygons $\mathcal{P}_1, \ldots, \mathcal{P}_n$ such that $Area(\mathcal{P}_i) = c_i Area(\mathcal{P})$ and S_i is

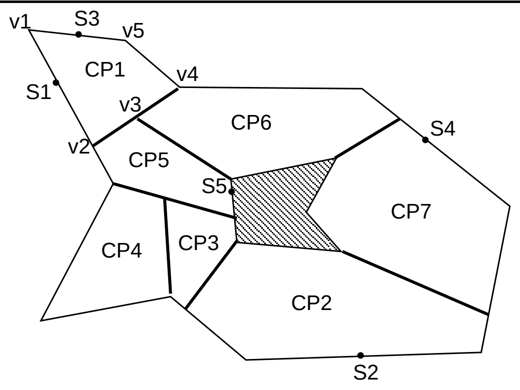

Figure 1: *The polygon $\mathcal{P}$ has been decomposed into 7 convex pieces $\mathcal{CP}_1, \ldots, \mathcal{CP}_7$. The neighbors of $\mathcal{CP}_5$ are $\mathcal{CP}_1$,$\mathcal{CP}_3$,$\mathcal{CP}_4$, and $\mathcal{CP}_6$. The vertices of $\mathcal{CP}_1$ are labeled $v_1, \ldots v_5$; $S(\mathcal{CP}_1) = \{S_1, S_3\}$. The neighbors of $\mathcal{CP}_1$ are $\mathcal{CP}_5$ and $\mathcal{CP}_6$; $NextNeighbor(\mathcal{CP}_1)$ is $\mathcal{CP}_5$. The list $W(\mathcal{CP}_1) = v_3, v_4, v_5, S_3, v_1, S_1, v_2$.*

on $\mathcal{P}_i$.

After introducing some additional notation in Section 2, we present in Section 3 an algorithm to solve the anchored area partition problem when polygon $\mathcal{P}$ is convex and simply connected. Section 4 presents a generalization of this algorithm for the case when $\mathcal{P}$ is nonconvex or nonsimply connected. Section 5 discusses a generalization of the algorithm presented in Section 4 that allows sites in the interior of the polygon $\mathcal{P}$. Examples of the algorithm's performance are given in Section 6, followed by some concluding remarks in Section 7.

2 Problem Description and Notation

An *area partition* of a polygon $\mathcal{P}$ is a set of nonoverlapping polygons $\mathcal{P}_1, \ldots, \mathcal{P}_n$, each of a specified area, such that $\cup_{i=1}^{n} Interior(\mathcal{P}_i) = Interior(\mathcal{P})$. If, in addition, each polygon $\mathcal{P}_i$ in the partition has a particular point (*site*) S_i on it the partition is also known as an *anchored area partition on* $S_1, \ldots, S_n$.

An anchored area partition problem is specified by providing the polygon $\mathcal{P}$, the set of sites $S_1, \ldots, S_n$ on $\mathcal{P}$ and, for each site S_i an *area requirement*, denoted $AreaRequired(S_i)$, which specifies the desired area of each polygon $\mathcal{P}_i$. In the problem we consider, $AreaRequired(S_i) = c_i Area(\mathcal{P})$, where $0 < c_i < 1$ and $\Sigma_{i=1}^{n} c_i = 1$. For any set of sites S, $AreaRequired(S) = \Sigma_{S_i \in S} AreaRequired(S_i)$.

The polygon $\mathcal{P}$ is assumed to have been decomposed into a collection of convex polygons. This can be done in time polynomial in the number of vertices of $\mathcal{P}$ [6, 15, 17, 19, 21, 31]. Let $\mathcal{CP}_j, j = 1, \ldots, p$ denote the set of nonoverlapping convex polygons (*pieces*) such that $Interior(\mathcal{P}) = \cup_{j=1}^{p} Interior(\mathcal{CP}_j)$. (When $\mathcal{P}$ is convex, $\mathcal{P} = \mathcal{CP}_1$.) Each piece $\mathcal{CP}_k$ for which $k < j$ is a *predecessor* of $\mathcal{CP}_j$; when $k > j$, $\mathcal{CP}_k$ is a *successor* of $\mathcal{CP}_j$. Any two pieces $\mathcal{CP}_j$ and $\mathcal{CP}_k$ are *neighbors* if they share an edge (Figure 1).

In general, the polygons $\mathcal{P}_i$ of an area partition will be nonconvex and, when the original polygon $\mathcal{P}$ contains holes, may be nonsimply connected. We construct each polygon $\mathcal{P}_i$ incrementally as a

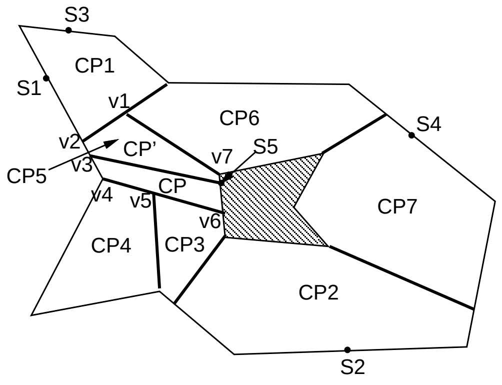

Figure 2: *Piece $\mathcal{CP}_5 = (v_1, v_2, v_4, v_5, v_6, v_7)$ has been divided into two smaller pieces CP and CP'. Assuming the pieces $\mathcal{CP}_j$ are ordered as their numbering indicates, $\mathcal{CP}_1, \ldots, \mathcal{CP}_4$ are predecessors of both CP and CP'. $PredPoly(CP, (v_4, v_5)) = \mathcal{CP}_4 + \mathcal{CP}_3 + \mathcal{CP}_2$; $PredPoly(CP, (v_5, v_6)) = \mathcal{CP}_3 + \mathcal{CP}_2$; $PredPoly(CP) = CP + PredPoly(CP, (v_4, v_5)) + PredPoly(CP, (v_5, v_6))$.*

union of smaller nonoverlapping polygons. Each smaller polygon constructed as part of $\mathcal{P}_i$ is said to be *assigned* to site S_i.

The algorithm we present uses a divide-and-conquer, sweep-line approach to construct an area partition. A given polygon P is divided into two polygons using one or more line segments, and each smaller polygon is recursively divided until the entire partition has been constructed. Each polygon P is represented as an ordered collection of convex pieces. The ordering of these pieces is derived from the ordering of the pieces of the convex decomposition of the original polygon $\mathcal{P}$. Associated with each convex piece CP of a polygon P is the following information (Figure 1):

$V(CP)$ – the vertices of the polygon CP, including all Steiner vertices (vertices of that are not vertices of the original polygon P) that lie on its edges.

$S(CP)$ – the list of sites to be assigned a portion of polygon CP, together with their individual area requirements;

$W(CP)$ – an ordered list of the vertices and sites of CP (The individual elements of $W(CP)$ are represented as $w_1, \ldots, w_m$.);

$Neighbors(CP)$ – A list of the convex pieces of P that share an edge with CP.

Each neighbor CP' of CP is marked as either a predecessor or a successor, depending on the ordering of the pieces of the original convex decomposition of $\mathcal{P}$ that contain CP and CP' (Figure 2). Pieces CP and CP' that are part of the same original piece $\mathcal{CP}_j$ are unordered. The pieces of P that are not neighbors of CP can be similarly designated as either predecessors or successors of CP. For each piece CP that has a successor, let $NextNeighbor(CP)$ denote the neighbor of CP that is its most immediate successor. Unless otherwise indicated, the points $W(CP)$ will be assumed to be ordered counterclockwise (CCW) such that (w_m, w_1) is the edge shared by CP and $NextNeighbor(CP)$. Polygon $PredPoly(CP)$ is defined as CP plus all predecessors of CP that are reachable from CP without entering a successor of CP. Similarly, $PredPoly(CP, e)$ is the collection of predecessors of CP that are reachable by crossing edge e of CP (Figure 2).

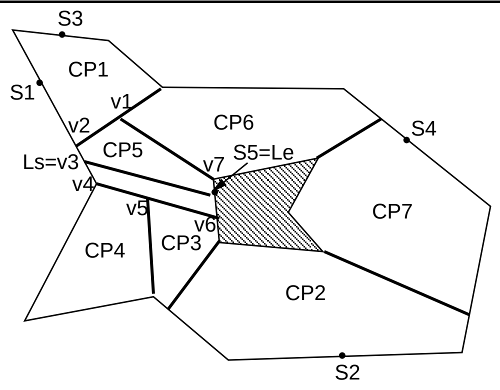

Figure 3: *Given line segment* $L = (L_s, L_e)$, $(\mathcal{CP}_5)^r_L = (L_s, v_4, v_5, v_6, L_e); (\mathcal{CP}_5)^l_L = (L_e, v_7, v_1, v_2, L_s)$; $\mathcal{P}^r_L = (\mathcal{CP}_5)^r_L + \mathcal{CP}_4 + \mathcal{CP}_3 + \mathcal{CP}_2$; $\mathcal{P}^l_L = (\mathcal{CP}_5)^l_L + \mathcal{CP}_1$.

Each polygon P is represented simply as a pointer to one of its convex pieces, from which the other pieces may be reached by following neighbor pointers. Given any convex piece CP, the union of pieces connected to it is referred to as the *polygon rooted at CP* and is denoted $Poly(CP)$. For any polygon P, the set $S(P) = \cup_{CP \in P} S(CP)$.

Let $L = (L_s, L_e)$ denote a line segment oriented from L_s to L_e such that L_s and L_e are both on some convex piece CP of polygon P. Polygon CP^r_L is the portion of CP to the right of L; CP^l_L is the complement $CP - CP^r_L$. Polygon P^r_L is defined as CP^r_L plus $PredPoly(CP, e)$ for each edge e to the right of or containing an endpoint of L; P^l_L is CP^l_L plus $PredPoly(CP, e)$ for each edge e to the left of or containing an endpoint of L (Figure 3).

A polygon P for which $|S(P)| = q$, is called a *q-site polygon*. A polygon P is called

- *area-complete* if $AreaRequired(S(P)) = Area(P)$;
- *area-incomplete* if $AreaRequired(S(P)) > Area(P)$;
- *site-incomplete* if $AreaRequired(S(P)) < Area(P)$.

Using this terminology, the anchored area partition problem on sites $S_1, \ldots, S_n$ can be restated as follows. Given an n-site area-complete polygon $\mathcal{P}$ divided into a set of convex pieces $\mathcal{CP}_j, j = 1, \ldots, p$, and sites $S(\mathcal{P}) = \{S_1, \ldots, S_n\}$ on $\mathcal{P}$, construct n 1-site area-complete polygons $\mathcal{P}_1, \ldots, \mathcal{P}_n$ with $S(\mathcal{P}_i) = \{S_i\}$.

3 Simply Connected, Convex Polygons

When $\mathcal{P}$ is convex, the desired area partition can be achieved using $n-1$ line segments, each of which divides a given q-site, area-complete polygon, $q > 1$, into two smaller convex polygons — a q_1-site area-complete polygon and a q_2-site area-complete polygon with $q_1 + q_2 = q$ and $q_1, q_2 > 0$. In this way it is always possible to achieve an n-site anchored area partition in which each of the polygons $\mathcal{P}_i$ is convex. Section 3.1 describes the algorithm for computing line segments that partition a convex polygon in this way. The complexity of this algorithm is discussed in Section 3.2.

```
Input - convex polygon $P$;
        the list $W(P) = w_k, k = 1, \ldots, m$ of vertices and sites in CCW order;
        the set of sites $S(P) = S_1, \ldots, S_q$ numbered according to their order in the list $W(P)$
Output - two polygons $P_L^l$ and $P_L^r$ and their sites $S(P_L^l)$ and $S(P_L^r)$.

Procedure $ConvexDivide$
  1. Assume $S_1 = w_k$; $L \leftarrow (w_1, w_k)$
  2. $S(P_L^r) \leftarrow \{S_1\}$
  3. while $Area(P_L^r) < AreaRequired(S(P_L^r))$ and $L_e \neq S_n$ do
        if $(k > 1)$ and $(w_{k-1} \in S(P))$ then
           $S(P_L^r) \leftarrow S(P_L^r) \cup w_{k-1}$
        $k \leftarrow k + 1$
        $L_e \leftarrow w_k$
  4. if $L_e = S_1$ and $Area(P_L^r) > AreaRequired(S(P_L^r))$ then
           Move $L_s$ CCW along $P_L^r$ until $Area(P_L^r) = AreaRequired(S(P_L^r))$
     else if $L_e = S_n$ and $Area(P_L^r) < AreaRequired(S(P_L^r))$ then
           Move $L_s$ CW along $P$ until $Area(P_L^r) = AreaRequired(S(P_L^r))$
     else
           Interpolate to find the point $t$ on edge $(w_{k-1}, w_k)$ such that if $L_e = t$ then
                       $Area(P_L^r) = AreaRequired(S(P_L^r))$
           $L_e \leftarrow t$
  5. $P_L^l = P - P_L^r$
  6. $S(P_L^l) = S \setminus S(P_L^r)$
```

Figure 4: *The procedure for dividing a convex area-complete polygon into two smaller area-complete polygons.*

3.1 Algorithm

The procedure for dividing a given convex, area-complete polygon P into two smaller area-complete polygons, which is summarized in Figure 4, works as follows. The list $W(P)$ of vertices and sites of P is provided as input, and the sites $S_1, \ldots, S_q$ are assumed to be numbered according to their appearance in this ordered list. The line segment $L = (L_s, L_e)$ is initialized as the segment (w_1, S_1). Using L_s as a pivot point, this segment is swept counterclockwise around the polygon until one of the following conditions holds:

1. $Area(P_L^r) = AreaRequired(S(P_L^r))$;
2. $S(P_L^r) = \{S_1\}$ and $Area(P_L^r) > AreaRequired(S(P_L^r))$;
3. $Area(P_L^r) < AreaRequired(S(P_L^r))$ and $L_e = S_n$.

In the first case, P_L^r and P_L^l are both area-complete polygons and, since L_e never passes S_n, each contains at least one site, so the desired division has been achieved. In the second case, there are no sites on P to the right of L, so the starting point of the segment L can be moved counterclockwise

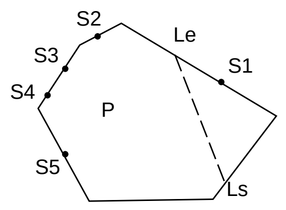

Figure 5: *When the 6-vertex polygon P is cut in two pieces by line segment (L_s, L_e), the result is a 3-vertex polygon and a 7-vertex polygon.*

around P until $Area(P_L^r) = AreaRequired(S(P_L^r))$. In the third case, there are no sites to the left of L, so L_s can be moved clockwise (CW) around P until the correct division of area is achieved.

Lemma 3.1 follows as a straightforward application of the Intermediate Value Theorem of calculus.

Lemma 3.1 *Any convex q-site area-complete polygon P, $q > 1$, can be divided into a convex q_1-site area-complete polygon and a convex q_2-site area-complete polygon with $q_1 + q_2 = q$ and $q_1, q_2 > 0$ using procedure $ConvexDivide$.*

Thus, through repeated applications of this procedure, a convex, n-site area-complete polygon can be partitioned into n convex, 1-site area-complete polygons.

3.2 Complexity Analysis

The procedure $ConvexDivide$ takes time linear in the size of the list $W(P)$. Each vertex and site in $W(P)$ is considered at most twice in this procedure: once to add it to either P_L^r or P_L^l and possibly once to subtract it if the area of P_L^r is initially too large (Step 4). Vertices can be added to or subtracted from polygons in constant time, and, as each vertex is added to P_L^r, the change in area (the area of the triangle added to the polygon) can be computed in constant time. Thus, in total, this procedure requires $O(n + v) = O(m)$ time, where $v = |V(P)|$ and $m = |W(P)|$.

In the worst case, $ConvexDivide$ will partition a q-site polygon, $q > 1$ with v vertices, into a 1-site polygon with 3 vertices and a $(q-1)$-site polygon with $v + 1$ vertices in $O(q + v)$ time (Figure 5). In this case, the next call to $ConvexDivide$ to partition the $(q-1)$-site polygon will also require $O(q+v) = O((q-1) + (v+1))$ time. To compute the complete partition of a convex n-site polygon P with v vertices, $ConvexDivide$ will be called exactly $n - 1$ times. Thus, in the worst case, it requires $O((n-1)(n+v))$ time to compute the complete partition into n 1-site polygons.

4 Nonconvex or Nonsimply Connected Polygons

The algorithm we present for computing an anchored area partition of a nonconvex or nonsimply connected polygon $\mathcal{P}$ is a generalization of the algorithm of Section 3. The polygon $\mathcal{P}$ is assumed to have been divided into a set of convex polygons $\mathcal{CP}_j, j = 1, \ldots, p$, using any one of a number of

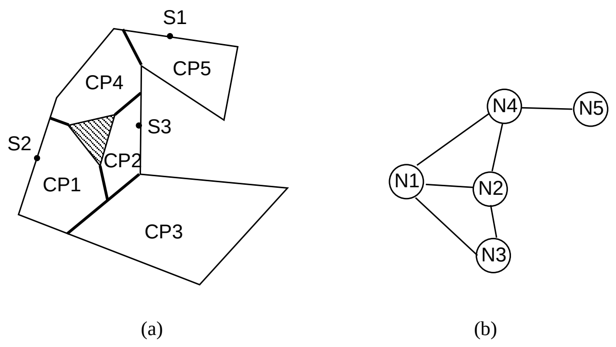

Figure 6: *(a) A given polygon and its convex decomposition. (b) The connectivity graph for the polygon shown in (a). When OrderPieces is called with* $\mathcal{CP}_1$*, the ordering produced is* $\mathcal{CP}_5, \mathcal{CP}_3, \mathcal{CP}_2, \mathcal{CP}_4, \mathcal{CP}_1$.

algorithms [6, 15, 17, 19, 21, 31]. These pieces are ordered and then processed in the specified order in a manner similar to that described for convex polygons in Section 3. The main difference lies, of course, in the fact that when $\mathcal{CP}_j \neq \mathcal{P}$ it is not necessarily the case that $\mathcal{CP}_j$ is area-complete. When $Area(\mathcal{CP}_j)$ is too small for the area requirements of $S(\mathcal{CP}_j)$ (*i.e.*, $\mathcal{CP}_j$ is area-incomplete), the remaining area required by the sites must be acquired from neighboring pieces; when the area of a piece is too large for its sites (*i.e.*, $\mathcal{CP}_j$ is site-incomplete), any part of the piece that is cordoned off for a particular site or set of sites must be done in such a way as to leave the unassigned and unprocessed portions of $\mathcal{P}$ connected. In Section 4.1.1 we describe the procedure for ordering the convex pieces $\mathcal{CP}_j, j = 1, \ldots, p$, of polygon $\mathcal{P}$. The algorithm for processing the pieces is described in Section 4.1.2 and the complexity of both procedures is discussed in Section 4.2.

4.1 Algorithm

4.1.1 Ordering the Pieces

Given the pieces $\mathcal{CP}_j$, a *connectivity graph* $G = (N, E)$ is built. This graph contains one node $N_j \in N$ for each convex piece $\mathcal{CP}_j$ and an edge $e_{jk} \in E$ between any two nodes N_j and N_k corresponding to neighboring pieces of the polygon (Figure 6). From the connectivity graph, an ordering of the pieces is determined. The ordering is such that only the last piece in the ordering has all its neighbors earlier in the ordering. In this way we assure that, as each piece is processed, all but one of its later neighbors can be ignored and it will still be possible to assign each site a connected portion of the polygon $\mathcal{P}$.

To produce an ordering of the pieces, the procedure *OrderPieces* summarized in Figure 7 is used. Given a particular node in the connectivity graph, *OrderPieces* walks through the graph in a depth-first manner, marking each node it visits. When a *leaf node* is found, its corresponding piece is placed next in the ordering. A leaf node is defined as a node N_j with one of the following properties:

1. N_j has only one neighbor;

Input - N_j - a node of the connectivity graph
Output - an ordered list of the pieces

```
Procedure OrderPieces
  if $N_j$ has not been marked then
    if $N_j$ is a leaf node then
      Mark node $N_j$
      output $\mathcal{CP}_j$
      for each neighbor $N_k$ of $N_j$
          $OrderPieces(N_k)$
    else
      Mark node $N_j$
      for each neighbor $N_k$ of $N_j$
          $OrderPieces(N_k)$
      output $\mathcal{CP}_j$
```

Figure 7: *The procedure used to produce an ordering of the convex pieces of a polygon.*

2. All of N_j's neighbors have been marked;

Consider, for example, the polygon shown in Figure 6. If *OrderPieces* is called with N_1 initially, the graph traversal will proceed as follows. Node N_1 is marked and then the neighbors of N_1 are ordered. If N_4 is the first neighbor of N_1 considered, it will be marked and then *OrderPieces* will be called for N_5. Since N_5 has only one neighbor, it is a leaf node, so $\mathcal{CP}_5$ is output first in the ordering. Then N_2, the next neighbor of N_4, is considered. It is not a leaf node, so its unmarked neighbor N_3 is considered. N_3 is a leaf node since both N_1 and N_2 have been marked. Thus $\mathcal{CP}_3$ is second in the ordering. After marking N_3, N_2 has no more unmarked neighbors and thus becomes a leaf node. Piece $\mathcal{CP}_2$ is placed third in the ordering, then $\mathcal{CP}_4$ and finally $\mathcal{CP}_1$.

Assume in what follows that the pieces $\mathcal{CP}_1, \ldots, \mathcal{CP}_p$ have been renumbered in accordance with the ordering produced by *OrderPieces*.

4.1.2 Dividing the Pieces

Each of the pieces $\mathcal{CP}_j, j = 1, \ldots, p$, is processed by dividing $PredPoly(\mathcal{CP}_j)$ into a number of pieces, each of which will either be assigned to a site or remain attached to the rest of $Poly(\mathcal{CP}_j)$, and thus become part of $PredPoly(\mathcal{CP}_k)$, for some $k > j$. The division is accomplished in a recursive fashion. Generally, $PredPoly(\mathcal{CP}_j)$ is divided into two parts using a single line segment that intersects $\mathcal{CP}_j$ at two points. One of these parts is removed from $Poly(\mathcal{CP}_j)$. Then each of the parts is divided until the entire piece has been divided among its sites.

There are two mutually recursive procedures used to accomplish the division of a particular convex piece and its predecessors. One constructs the line segments that divide $PredPoly(CP)$ into its two parts and the other removes polygons created by this division and either assigns them to a particular site or recursively divides them as necessary. We first describe the procedure for constructing the line segment to divide $PredPoly(CP)$. This procedure is summarized in Figure 8.

Input – CP - a convex piece of a polygon

```
Procedure $NonconvexDivide$
    1. Assume $S_i = w_k$, where $S_i \in S(CP)$ is the first site CCW from $w_1$; $L \leftarrow (w_1, w_k)$
    2. $S(P^r_L) \leftarrow \{S_i\}$
    3. while $Area(P^r_L) < AreaRequired(S(CP^r_L))$ and $L_e \neq w_m$ do
          if $(k > 1)$ and $(w_{k-1} \in S(CP))$ then
             $S(CP^r_L) \leftarrow S(CP^r_L) \cup w_{k-1}$
          $k \leftarrow k + 1$; $L_e \leftarrow w_k$
    4. if $Area(P^r_L) > AreaRequired(S(CP^r_L))$ then
             if $L_e = S_i$ then
                $k_1 \leftarrow 1$
                while $Area(P^r_L) > AreaRequired(S(CP^r_L))$ do
                     $k_1 \leftarrow k_1 + 1$; $L_s \leftarrow w_{k_1}$
                $L_1 \leftarrow (w_{k_1}, L_e)$; $T \leftarrow (t_1, t_2, t_3) \leftarrow (w_{k_1}, w_{k_1-1}, L_e)$
             else
                $L_1 \leftarrow (L_s w_{k-1})$; $T \leftarrow (t_1, t_2, t_3) \leftarrow (w_{k-1}, w_k, L_s)$;
             if $Area(P^r_{L_1} + T) > AreaRequired(S(CP^r_L))$ then
                Compute area interpolation point $t$ on $(t_1, t_2)$
                $T' \leftarrow triangle(t_1, t, t_3)$
                $DetachAndAssign(P^r_{L_1} + T' - PredPoly(CP, (t_1, t)))$
                $DetachAndAssign(P^l_{L_1} - T')$
             else if $Area(P^r_{L_1} + PredPoly(CP, (t_1, t_2))) < AreaRequired(S(CP^r_L))$ then
                Compute area interpolation point $t$ on $(t_1, t_2)$
                $T' \leftarrow triangle(t_1, t, t_3)$
                $DetachAndAssign(P^r_{L_1} + T')$
                $DetachAndAssign(P^l_{L_1} - T' - PredPoly(CP, (t_1, t))))$
             else
                $PS \leftarrow$ interior point of $(t_1, t_2)$; $T' \leftarrow triangle(t_1, PS, t_3)$
                $AreaRequired(PS) \leftarrow AreaRequired(S(CP^r_L)) - Area(P^r_{L_1} + T')$
                Order $W(PredPoly(CP, (t_1, t_2)))$ such that
                         $w_1 = PS$ if $L_e \neq S_i$ and $w_m = PS$ if $L_e = S_i$
                $DetachAndAssign(PredPoly(CP, (t_1, t_2)))$
                $DetachAndAssign(P^r_{L_1} + T')$
                $DetachAndAssign(P^l_{L_1} - T')$
       else
             $t \leftarrow$ interior point of $(w_m, w_1)$
             $L_1 \leftarrow (t, S_i)$
             $DetachAndAssign(P^r_{L_1})$
             $DetachAndAssign(P^l_{L_1})$
```

Figure 8: *The procedure for dividing a nonconvex polygon by sweeping a line around one of its convex pieces.*

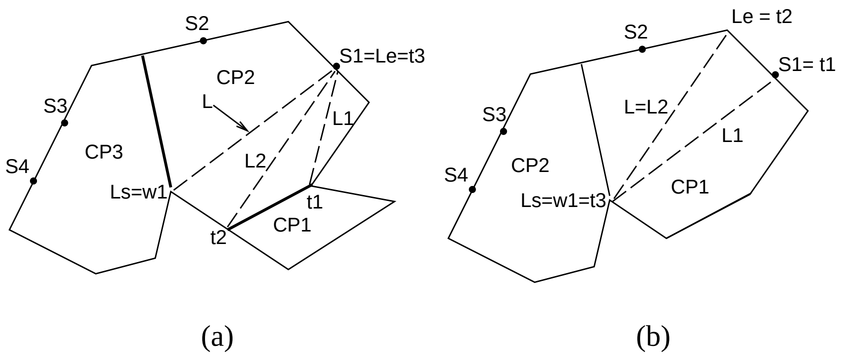

Figure 9: *(a) When* $L = (w_1, S_1)$, $Area(P_L^r) > AreaRequired(S_1)$*; for* L_1 *and* L_2, $Area(P_{L_1}^r) < AreaRequired(S_1) < Area(P_{L_2}^r)$*. The segments* L_1 *and* L_2 *are discovered by moving* L_s *CCW around* CP_2 *toward* S_1*. (b) When* $L = L_2 = (w_1, t_2)$ $Area(P_L^r) > AreaRequired(S_1)$*, but* $Area(P_{L_1}^r) < AreaRequired(S_1)$*.*

The $NonconvexDivide$ **procedure**

As in the $ConvexDivide$ algorithm described in Section 3, a piece CP is divided by sweeping a line segment counterclockwise around CP. However, unlike for the $ConvexDivide$ algorithm, a particular ordering of the points $W(CP)$ is necessary to assure correctness of the more general algorithm described here. In particular, the points $W(CP)$ need to be ordered such that the line constructed to divide $PredPoly(CP)$ makes it possible to detach a portion of $PredPoly(CP)$ and leave the rest attached to a successor of CP. In general, this is accomplished by ordering the points $W(CP)$ such that (w_m, w_1) is the edge to $NextNeighbor(CP)$. When there is no $NextNeighbor(CP)$, the list $W(CP)$ is ordered so $w_m = S_i$ for some $S_i \in S(CP)$.

Recall that CP_L^r is the portion of CP to the right of L and P_L^r is CP_L^r plus all predecessors of CP that are reachable from edges to the right of or containing an endpoint of L. The dividing line L is constructed as follows. The segment L is initialized as (w_1, S_i), where S_i is the first element of $S(CP)$ counterclockwise from w_1. Then, using w_1 as a pivot point, L is swept around CP until either

1. the area of the unassigned portion of the polygon $Poly(CP)$ to the right of L is greater than or equal to the area required by the sites on the portion of CP to the right of L ($Area(P_L^r) \geq AreaRequired(S(CP_L^r))$), or
2. the point w_m is encountered ($L_e = w_m$).

Case 1: If the endpoint of segment L is swept around CP to a point where $Area(P_L^r) > AreaRequired(S(CP_L^r))$, there are two cases to consider. Either $S(CP_L^r) = \{S_i\}$ and $L_e = S_i$, for some i, in which case it is necessary to move the starting point L_s of segment L to decrease $Area(P_L^r)$ and achieve the correct division, or $L_e \neq S_i$ and the ending point of L must be adjusted. In either case, let L_1 and L_2 be two segments that share the endpoint of L that does not change (either L_e in the first case, or L_s in the second case) and be such that $Area(P_{L_1}^r) < AreaRequired(S(CP_{L_1}^r))$ and $Area(P_{L_2}^r) > AreaRequired(S(CP_{L_2}^r))$. Let $T = (t_1, t_2, t_3)$ be the triangle that is the difference

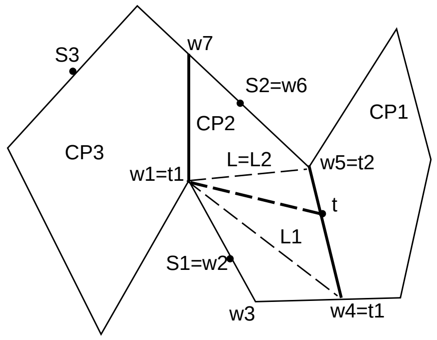

Figure 10: *If the area of polygon* (w_1, w_3, w_4) *is less than* $AreaRequired(S_1)$ *and the area of polygon* $(w_1, w_3, w_4, w_5) > AreaRequired(S_1)$, *there must exist a point* t *in the interior of edge* $(w_4, w_5) = (t_1, t_2)$ *such that* $AreaRequired(S_1) = Area((w_1, w_3, w_4, t))$. *Removing this polygon leaves* $\mathcal{CP}_1$ *still attached to the rest of the polygon.*

between $CP^r_{L_1}$ and $CP^r_{L_2}$ and let (t_1, t_2) be the edge of CP connecting L_1 and L_2 (Figure 9). There are three cases to consider:

1.1 $Area(P^r_{L_1} + T) > AreaRequired(S(CP^r_L))$

1.2 $Area(P^r_{L_1}) + T) \leq AreaRequired(S(CP^r_L))$ and
$Area(P^r_{L_1} + PredPoly(CP, (t_1, t_2))) < AreaRequired(S(CP^r_L))$

1.3 $Area(P^r_{L_1} + T) \leq AreaRequired(S(CP^r_L))$ and
$Area(P^r_{L_1} + PredPoly(CP, (t_1, t_2))) \geq AreaRequired(S(CP^r_L))$

Case 1.1: In the first case, no portion of $PredPoly(CP, (t_1, t_2))$, if it exists, is needed to satisfy the area requirements of the sites $S(CP^r_L)$. A simple linear interpolation will produce the *area interpolation point* $t \in (t_1, t_2)$ such that, if T' is the triangle (t_1, t, t_3), $Area(P^r_{L_1} + T' - PredPoly(CP, (t_1, t))) = AreaRequired(S(CP^r_L))$. Polygons $P^r_{L_1} + T' - PredPoly(CP, (t_1, t))$ and $P^l_{L_1} - T'$ are then divided recursively. Note that since the condition for this case is a strict inequality, $t \neq t_2$ and thus $PredPoly(CP, (t_1, t_2))$ will remain connected to $P^l_{L_1} - T'$ (Figure 10).

Case 1.2: In the second case all of $PredPoly(CP, (t_1, t_2))$ can be used to satisfy the area requirements of $S(CP^r_L)$. Again, linear interpolation will discover an area interpolation point t on the edge (t_1, t_2) that results in a proper division of the area. Defining T' as in the Case 1.1, processing proceeds by dividing polygons $P^r_{L_1} + T'$ and $P^l_{L_1} - T' - PredPoly(CP, (t_1, t))$ recursively. Due to the strict inequality $Area(P^r_{L_1}) + PredPoly(CP, (t_1, t_2))) < AreaRequired(S(CP^r_L))$, $t \neq t_1$ in this case and the polygon $P^r_{L_1} + T'$ will not be degenerate at this point (Figure 11).

Case 1.3: In the last case, only a portion of the polygon $PredPoly(CP, (t_1, t_2))$ is necessary to satisfy the area requirements of sites $S(CP^r_L)$. Therefore $PredPoly(CP, (t_1, t_2))$ must be divided into a portion that will be assigned to $S(CP^r_L)$ and a portion that will remain unassigned. This division is accomplished through the use of a *pseudo-site*. A pseudo-site is a point on an edge between CP and one of its neighbors that is treated as if it were a real site when the neighboring piece is processed. In this case, the location of the pseudo-site PS is computed as any point in

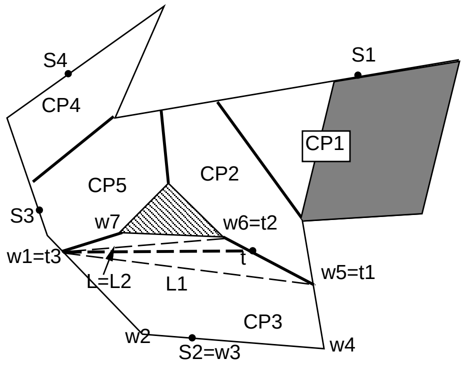

Figure 11: *Piece $\mathcal{CP}_1$ has been divided and the shaded portion assigned to site S_1. $PredPoly(\mathcal{CP}_3,(w_5,w_6))$ = $\mathcal{CP}_2$+ (remainder of $\mathcal{CP}_1$). When $\mathcal{CP}_3$ is divided, L_e is moved around $\mathcal{CP}_3$ to the point w_6 where $Area(P_L^r) > AreaRequired(S_2)$. If it is the case that the area of $PredPoly(\mathcal{CP}_3,(w_5,w_6))$ plus $Area((w_1,w_2,w_4,w_5))$ is less than $AreaRequired(S_2)$, there must exists a point t on the edge (w_5,w_6) such that $Area(PredPoly(\mathcal{CP}_3,(w_5,w_6))) + Area((w_1,w_2,w_4,w_5,t)) = AreaRequired(S_2)$.*

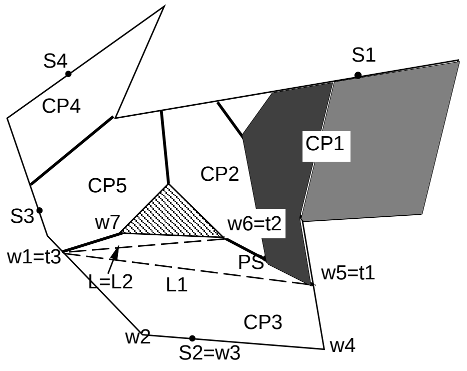

Figure 12: *The lighter shaded area has been assigned to S_1, as in Figure 11. If $Area((w_1,w_2,w_4,w_5,w_6)) < AreaRequired(S_2) \leq Area((w_1,w_2,w_4,w_5))$ $+Area(PredPoly(\mathcal{CP}_3,(w_5,w_6)))$, then $PredPoly(\mathcal{CP}_3,(w_5,w_6))$ must be divided. The point PS is a pseudo-site used to acquire area from $PredPoly(\mathcal{CP}_3,(w_5,w_6))$. The darker shaded area, encompassing parts of $\mathcal{CP}_1$ and $\mathcal{CP}_2$, is the polygon that gets assigned to PS by recursive calls to the divide procedure. This polygonal region will then be assigned to S_2.*

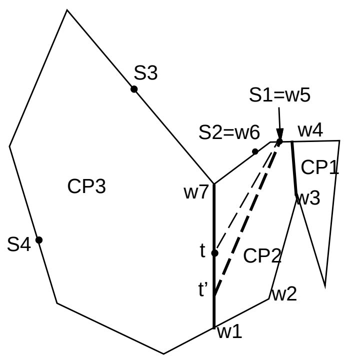

Figure 13: *$Area(\mathcal{CP}_1+\mathcal{CP}_2) < AreaRequired(\{S_1, S_2\})$, so these polygons must be divided among the sites S_1 and S_2. The point t on edge (w_7, w_1) is computed and the line $L = (t, S_1)$ constructed. If $Area((t, w_1, w_2, w_3, w_4, w_5)) + Area(\mathcal{CP}_1) > AreaRequired(S_1)$, a recursive call to $NonconvexDivide$ will determine the point t' that results in a polygon of the correct area for S_1. Otherwise, a pseudo-site for S_1 will be created on the segment (t, w_1).*

the interor of edge (t_1, t_2). Let T' be the triangle (t_1, PS, t_3). The area requirement for PS is the difference between $AreaRequired(S(CP^r_L))$ and $Area(P^r_{L_1} + T')$. Once the pseudo-site has been computed, $PredPoly(CP, (t_1, PS))$ is divided and pieces of $PredPoly(CP, (t_1, PS))$ are assigned to PS. The pieces assigned to PS in this division are then added to $P^r_{L_1} + T'$ and this new area-complete polygon is divided recursively, as is the polygon $P^l_{L_1} - T'$ (Figure 12).

Case 2: If the line segment L is swept all the way around the polygon and it is never the case that $Area(P^r_L) > AreaRequired(S(CP^r_L))$, then it is necessary to create at least one pseudo-site for at least one of the sites in $S(CP^r_L)$ in order to acquire area from a successor of CP. This is accomplished as follows. A point t in the interior of the edge (w_m, w_1) connecting CP to $NextNeighbor(CP)$ is chosen. Let $L = (t, S_i)$ where S_i is the first site in $S(CP)$ counterclockwise from w_1. The elements of $W(P^r_L)$ are ordered such that $t = w_1$ then polygon P^r_L is divided. This division results either in the creation of a pseudo-site on edge (w_1, w_2) of P^r_L if $Area(P^r_L)$ is too small to satisfy the area requirement of S_i, or in the assignment to S_i of a portion of P^r_L that satisfies its area requirement. In either case, after P^r_L has been divided and a portion of it removed, the remaining portion is divided. This could result in the creation of more pseudo-sites on the edge (w_m, w_1) of CP. These pseudo-sites will be encountered when $NextNeighbor(CP)$ is processed. Pieces of the polygon assigned to the pseudo-sites will actually be assigned to their corresponding original sites (Figure 13). The assignment of polygonal pieces to sites and creation of pseudo-sites is accomplished by the $DetachAndAssign$ procedure, described below.

Note that, given a q-site area-compete polygon, the $NonconvexDivide$ procedure always results in the creation of either a q_1-site area-complete polygon and a q_2-site area-complete polygon, $q_1, q_2 > 0, q_1 + q_2 = q$ (Cases 1.1, 1.2, 1.3, and 2), or a 1-site area-incomplete polygon that is assigned to a particular site and another q-site area-complete polygon of smaller area (Case 2). Note also that given a convex piece CP with $|S(CP)| = q$, the dividing line L will divide CP into a q_1-site convex polygon and a q_2-site polygon, $q_1 + q_2 = q$ but such that one of q_1 and q_2 may be zero. A 0-site polygon will be created by the division when $PredPoly(CP)$ is site-incomplete. Due

```
Input – $Poly(CP)$ - a polygon rooted at convex piece $CP$

Procedure $DetachAndAssign$
  if $|S(CP)| = 0$ then return

  if $PredPoly(CP)$ is area-complete then
    if $S(CP) = \{S_i\}$ for some $i$ then
      Assign $PredPoly(CP)$ to $S_i$
      Detach $PredPoly(CP)$ from $Poly(CP)$
    else
      Detach $PredPoly(CP)$ from $Poly(CP)$
      Order $W(CP)$ such that $w_m = S_i$ for some $S_i \in S(CP)$
      $NonconvexDivide(CP)$
  else if $PredPoly(CP)$ is area-incomplete then
    if $S(CP) = \{S_i\}$ for some $i$ then
      Assign $PredPoly(CP)$ to $S_i$
      Detach $PredPoly(CP)$ from $Poly(CP)$
      Make pseudo-site for $S_i$ on edge to $NextNeighbor(CP)$
    else
      Order $W(CP)$ such that edge $(w_m, w_1)$ is the edge to $NextNeighbor(CP)$.
      $NonconvexDivide(CP)$
  else
    Order $W(CP)$ such that $(w_m, w_1)$ is the edge to $NextNeighbor(CP)$
    $NonconvexDivide(CP)$
```

Figure 14: *The procedure for removing portions of a polygon and assigning them to sites or recursively dividing them as necessary.*

to the prescribed ordering of the elements of $W(CP)$, this 0-site polygon will remain attached to $NextNeighbor(CP)$. Note also, that after all recursive divisions of a particular piece CP are done, only 0-site portions of $PredPoly(CP)$ remain; all other portions will have been removed and assigned to sites. Thus we have the condition that for any piece CP, $S(PredPoly(CP)) = S(CP)$.

The *DetachAndAssign* procedure

Next we describe the procedure, summarized in Figure 14, that detaches pieces of the polygon carved out by the dividing lines constructed in *NonconvexDivide* and either assigns them to particular sites or divides them by calling *NonconvexDivide* again.

As pointed out above, for any piece CP of a polygon P, $S(PredPoly(CP)) = S(CP)$. The area requirements of the sites $S(CP)$ therefore dictate how $PredPoly(CP)$ is divided. Once $PredPoly(CP)$ is divided, the remaining portion of $Poly(CP)$ will be divided by subsequent recursive calls to *DetachAndAssign* and *NonconvexDivide*. Given any convex piece CP, there are three cases to consider:

1. $PredPoly(CP)$ is area-complete;

2. $PredPoly(CP)$ is area-incomplete;

3. $PredPoly(CP)$ is site-incomplete.

Case 1: In the first case, if there is only one site $S_i \in S(CP)$, it is not necessary to divide $PredPoly(CP)$ further; it is assigned to site S_i and removed from polygon $Poly(CP)$. If $PredPoly(CP)$ is area-complete and there is more than one site in $S(CP)$ then $PredPoly(CP)$ is simply removed from $Poly(CP)$ and processed as a smaller version of the original problem, for which an ordering of the convex pieces has already been established.

Case 2: In the second case, since $Area(PredPoly(CP)) < AreaRequired(S(CP))$, not only must the polygon $PredPoly(CP)$ be divided among its sites, but area must also be acquired from neighboring pieces for the sites in $S(CP)$. As indicated above, this is accomplished through the use of pseudo-sites. If $S(CP) = \{S_i\}$ for some i, then $PredPoly(CP)$ is assigned to S_i and removed from $Poly(CP)$. Then the location of the pseudo-site PS_i is computed as an interior point of the edge connecting CP to $NextNeighbor(CP)$. The area requirement for PS_i is $AreaRequired(S_i)$ less the area of the pieces already assigned to S_i. When $NextNeighbor(CP)$ is processed, pieces of the polygon will be assigned to S_i through its pseudo-site PS_i. If $|S(CP)| > 1$, a pseudo-site is not created yet since it is not clear which of the sites will require the extra area. The division procedure is called to divide $PredPoly(CP)$. The division procedure will recursively divide $PredPoly(CP)$, eventually producing at least one piece CP' such that $|S(CP')| = 1$ and $Area(PredPoly(CP')) < AreaRequired(S(CP'))$, at which point the pseudo-site will be created.

Case 3: In the third case, the area of $PredPoly(CP)$ is too large for the area requirements of the sites $S(CP)$. Therefore, part of $PredPoly(CP)$ must be assigned to sites on pieces that have yet to be processed. That is, a 0-site portion of $PredPoly(CP)$ must be created. This is accomplished by simply calling the division procedure for $PredPoly(CP)$.

Lemma 4.1 illustrates the correctness of the above procedures and follows from repeated application of the Intermediate Value Theorem.

Lemma 4.1 *Any q-site area-complete polygon P, $q > 1$, will be divided by the procedures NonconvexDivide and DetachAndAssign into either a q_1-site area-complete polygon and a q_2-site area-complete polygon with $q_1 + q_2 = q$ and $q_1, q_2 > 0$ or a 1-site area-incomplete polygon, which is removed from P, and a q-site area-complete polygon P' with $Area(P') < Area(P)$. In the latter case, one of the q sites of P' is a pseudo-site located on the boundary between P' and the part of P that was removed.*

It follows from this lemma that all the pieces assigned to a particular site, either through pseudo-sites or not, must be connected. Thus, given any q-site area-complete polygon, repeated applications of these two mutually recursive procedures must produce a set of q 1-site area-complete polygons. The area partition of $\mathcal{P}$ can be constructed by calling *DetachAndAssign* with each piece $\mathcal{CP}_j, j = 1, \ldots, p$, in turn.

4.2 Complexity Analysis

In this section we show that for a polygon $\mathcal{P}$ that has been divided into p pieces $\mathcal{CP}_1, \ldots, \mathcal{CP}_p$ such that $v_j = |V(\mathcal{CP}_j)|$ and $v = \Sigma_{i=1}^{p} v_j$, the procedures *OrderPieces*, *NonconvexDivide*, and *DetachAndAssign* require $O(pn^2 + vn)$ time to construct an anchored area partition on n sites in the worst case.

To order the convex pieces $\mathcal{CP}_1, \ldots, \mathcal{CP}_p$ of the polygon $\mathcal{P}$, *OrderPieces* visits each node of the corresponding connectivity graph at most twice. Thus the ordering can be produced in time linear in the number of pieces p.

The *NonconvexDivide* procedure requires $O(pn^2 + nv)$ time to divide all p pieces among the n sites. In the worst case, each of the p pieces will be divided into n polygons, each of which will be assigned to one of the sites. For each of these n polygons carved out of a piece $\mathcal{CP}_j$, the division requires $O(n + v_j)$ time, where $v_j = |V(\mathcal{CP}_j)|$. Thus in total it requires $O(pn^2 + vn)$ time to divide all the pieces. Note that this is true even though it is not always the case that when a piece CP is divided, the change in the area of CP_L^r as new vertices are added to it can be computed in constant time. When an edge e is added to CP_L^r and $PredPoly(CP, e)$ exists, it is sometimes necessary to visit each piece of $PredPoly(CP, e)$ to compute the change in the area of CP_L^r. This requires time linear in the number of pieces of $PredPoly(CP, e)$. However, for each site that is assigned some portion of a piece $\mathcal{CP}_j$, the number of times that piece, or a portion of it, must be visited to discover its area can be limited to a constant number using the proper data structure. Thus the computation of $Area(PredPoly(CP, e))$ requires at most $O(pn)$ time in total, and the overall running time of $O(pn^2 + nv)$ for the procedure *NonconvexDivide* is not affected.

To detach a piece of the polygon constructed in *NonconvexDivide* it is necessary to update the neighbor pointers of the piece and its current neighbors. For a convex polygon CP_j with v_j vertices, this requires $O(v_j)$ time. In the worst case, the division of a convex piece with v_j vertices will result in one convex piece with three vertices and one with $v_j + 1$ vertices. If each piece is divided among all the sites and each division results in an additional vertex the total time required to detach each of these pieces is no more than the sum of the number of vertices in the worst case, which is $O(v_j + n^2)$. So for all p pieces, the total time required to detach the pieces is $O(v + pn^2)$.

Assigning a polygon to a particular site can be done in constant time if with each pseudo-site PS_i created for site S_i information about where to attach the pseudo-site's polygon to the remainder of site S_i's polygon is stored. Thus, as it is constructed each site's polygon will be a collection of convex pieces. Merging these into a single polygon requires time linear in the sum of the number of vertices of the pieces, which is at most $O(pn + v)$. Thus, to merge all n site polygons requires $O(pn^2 + vn)$ time.

Thus we see that *DetachAndAssign* and *NonconvexDivide* both require $O(pn^2 + vn)$ time and *OrderPieces* requires $O(p)$ time. The total time required to compute the anchored area partition on n sites using these procedures is $O(pn^2 + vn)$.

5 Interior Sites

Using the algorithm described in Section 4, sites in the interior of the polygon $\mathcal{P}$ can be easily handled with a minor modification to the initial convex decomposition of $\mathcal{P}$. We need only assure that each site S_i lies on some piece of the convex decomposition. If the original decomposition does not accomplish this, then for each site S_i that lies in the interior of a piece $\mathcal{CP}_j$, we simply divide that piece in two using a line segment that passes through the site S_i (Figure 15). The orientation of this line segment may be chosen arbitrarily without affecting the correctness of the algorithm. Using this modified convex decomposition, the algorithm then proceeds exactly as described in Section 4.

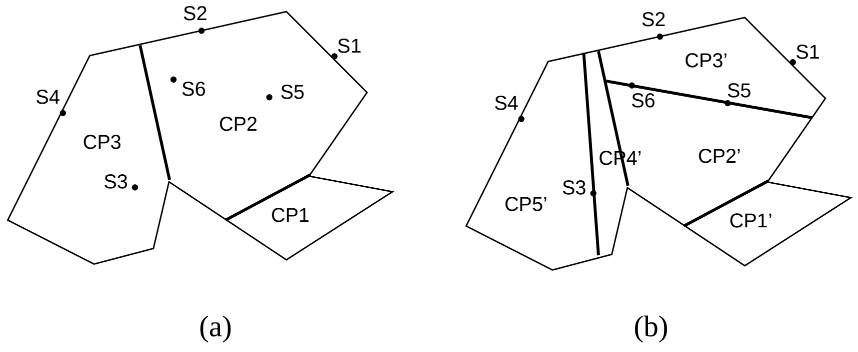

Figure 15: *(a) The original convex decomposition of $\mathcal{P}$ results in three pieces and three sites in the interior of some piece (b) A line is drawn through each interior site creating a new convex decomposition with five pieces. Note that the orientation of line that divides any piece may be chosen arbitrarily. The line dividing $\mathcal{CP}_2$ into $\mathcal{CP}'_2$ and $\mathcal{CP}'_3$ is chosen to pass through both sites S_5 and S_6.*

6 Examples

In this section, we present examples of our algorithm's performance for convex, nonconvex, and nonsimply connected workspaces. The first example, Figure 16, shows a nonregular convex polygon with $v = 7$ vertices and $n = 7$ sites. In this example, the area requirements for the sites, shown as a percent of the total area in parentheses next to each site, are unequal. Figure 17 shows a nonconvex polygon with 12 vertices that is to be divided into 7 equal-area pieces. The polygons labeled $CP_1, \ldots, CP_5$ in Figure 17(a) constitute the initial decomposition of the polygon $\mathcal{P}$ into convex pieces, numbered according to the order in which they will be processed. Figures 17(b) through (e) show the progression of the algorithm through the division of each convex piece. Figure 17(f) shows the resulting equal-area partition for the given sites. Figure 18 presents an example of an environment with two polygonal holes that is to be divided into 5 equal-area polygons. The initial convex decomposition results in the 11 convex pieces $CP_1, \ldots, CP_{11}$, again numbered in the order in which they will be processed. From these pieces the partition shown in Figure 18(b) is produced. Figure 19 shows an environment with one polygonal hole and four sites, two of which line in the interior of the polygon. The polygon is to be divided into four equal-area pieces. A convex decomposition of the original polygon that results in each site lying on some convex piece is shown in Figure 19(a); the resulting anchored area partition is presented in Figure 19(b).

In each of these examples, the partition shown is not the only one that could be produced by our algorithm. For convex polygons, different partitions can be achieved with different orderings of the vertices (Figure 20(a)). For nonconvex and nonsimply connected polygons, different partitions can be achieved with different orderings of the convex pieces (Figure 21) or with different convex decompositions. Similarly, different partitions can be achieved when one or more sites lie in the interior of the polygon by choosing different orientations for the segments that pass through each interior site and divide the containing convex pieces.

Different partitions of a given polygon can also be achieved by changing the distribution of the

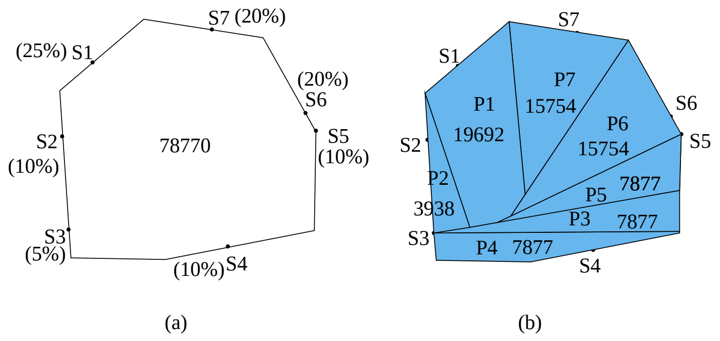

Figure 16: *An anchored area partition for 7 sites with unequal area requirements. Each polygon is labeled with its area. (a) The original polygon, with each site labeled with the percent of the total area required. (b) The resulting partition.*

sites on the polygon (Figure 20(b)). Generally, it is desirable to produce more simply shaped regions (*e.g.*, because robot terrain-covering algorithms can be implemented more easily and efficiently in more simply shaped regions). One measure of the simplicity of a polygon P is its *compactness*, computed, for example, as $Area(P)/Perimeter(P)$. Figure 22 shows a plot of average compactness of the partition polygons relative to the original polygon when sites are distributed uniformly and randomly around the perimeter of a convex polygon. Each data point represents an average over 400 partitions created from 20 random convex polygons, each with 20 distributions of sites. The relative compactness does not vary significantly with the number of vertices of the original polygon. However, as the number of sites increases, the relative compactness decreases. More significantly, the difference between the relative compactness for uniformly distributed sites and randomly distributed sites increases, with the values for uniformly distributed sites consistently higher. This data suggests that distributing the sites uniformly around the initial polygon generally results in more compact partition polygons.

7 Conclusion

We have introduced a new type of polygon decomposition for arbitrary, simple polygons — the area partition. Each polygon of this decomposition is constructed to have a particular area. We have considered here only one version of the area partition problem, the anchored area partition problem, which is motivated by a multiple-robot terrain-covering problem in robotics. In the terrain-covering problem, each part of a given polygonal region must be visited by one of a collection of n robots. Thus the original region should be divided into a set of n smaller, nonoverlapping regions, one for each robot; the desired areas of these regions are determined by the relative terrain-covering capabilities of the individual robots. The polynomial-time algorithm we present will partition any simple polygon – convex or nonconvex and with or without polygonal holes. When the original polygon $\mathcal{P}$ is convex and without holes, each polygon of the area partition will also be convex. No such nice properties are guaranteed when $\mathcal{P}$ is not convex or contains holes. In fact, in this case, the polygons of the partition could contain degeneracies.

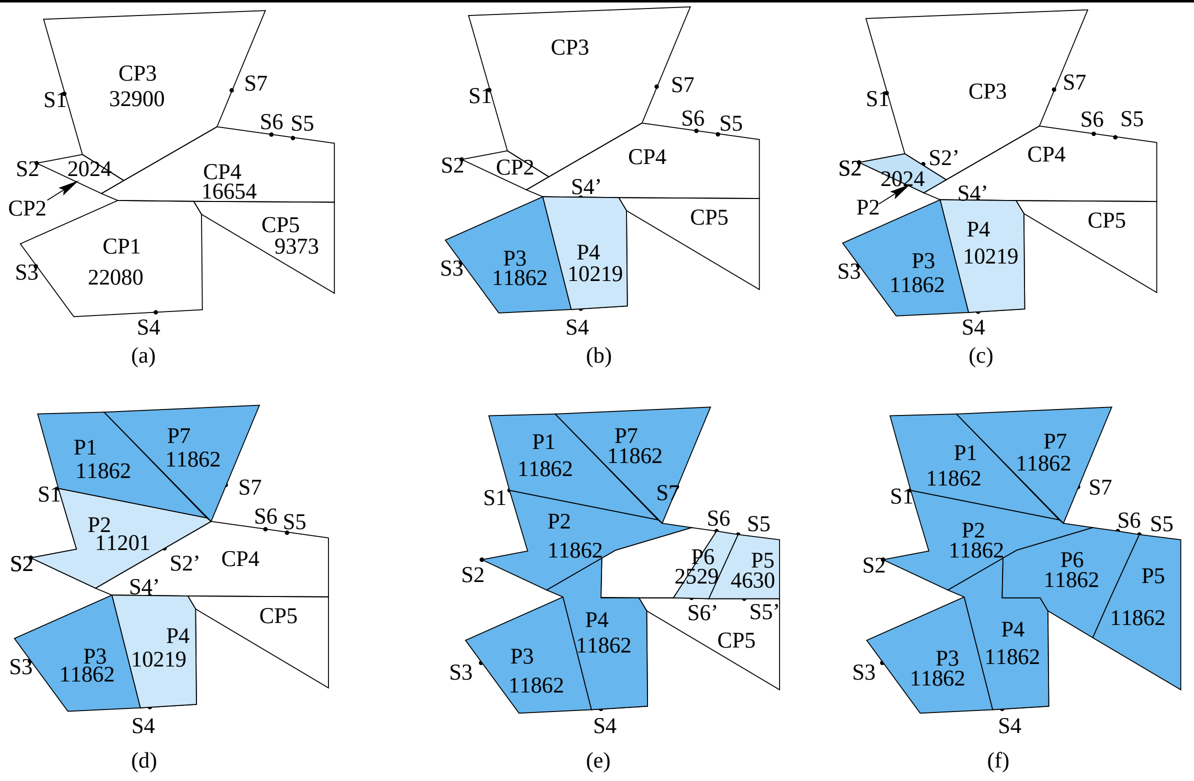

Figure 17: *An anchored area partition for a nonconvex polygon with equal area requirements for 7 sites. The total area of the polygon is 83031. Dark grey polygons are area-complete; light grey ones are area-incomplete. Polygons are labeled with their areas. (a) The initial convex pieces* $CP_1, \ldots, CP_5$*. (b)* CP_1 *is divided between sites* S_3 *and* S_4*. A pseudo-site* S_4' *is created on the edge between* CP_1 *and its next neighbor* CP_4*. (c)* CP_2 *is assigned to* S_2 *and pseudo-site* S_2' *is created. (d)* CP_3 *is divided into two area-complete polygons* P_1 *and* P_7 *and an area-incomplete polygon is assigned to* S_2' *and merged with the previous* P_2*. A new pseudo-site* S_2' *is created on the edge between* CP_3 *and* CP_4*. (e) With the endpoint of the sweep line at the right endpoint of the edge between* CP_4 *and* CP_5*, it is never the case that* $Area(P_L^r) > AreaRequired(S(P_L^r))$*, so a pseudo-site* S_5' *is created on the edge between* CP_4 *and* CP_5 *and the area-incomplete polygon* P_5 *is assigned to* S_5*. The process is repeated after removing* P_5 *from* CP_4 *and the area-incomplete polygon* P_6 *and pseudo-site* S_6' *are created. Then area-complete polygons are created for pseudo-sites* S_2' *and* S_4' *and merged with polygons* P_2 *and* P_4*, respectively. (f)* CP_5 *and the remainder of* CP_4 *are divided between pseudo-sites* S_5' *and* S_6'*.*

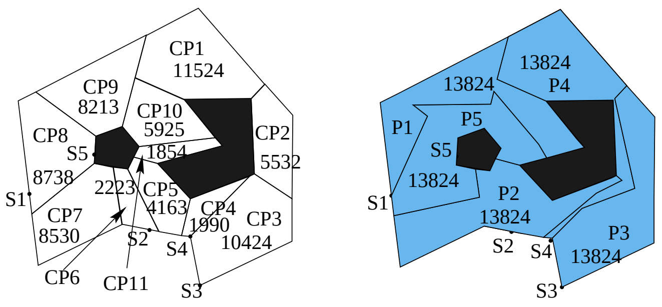

Figure 18: *An anchored area partition for a nonsimply connected polygon. The black areas represent the polygonal holes in the workspace. (a) The original polygon divided into 11 convex pieces, each labeled with its area. The total area of the polygon is 69118. (b) The resulting partition. Note that polygon P_3, while quite narrow in some places, is not disconnected.*

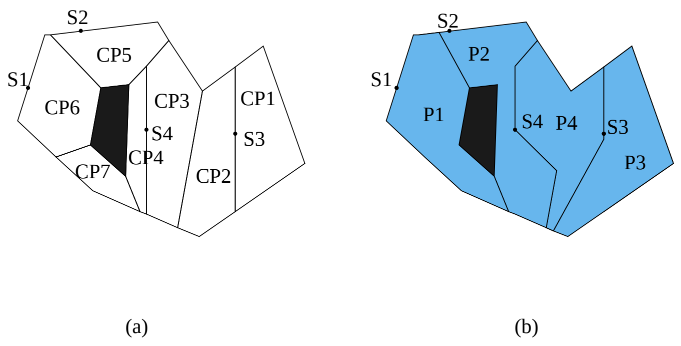

Figure 19: *An anchored area partition for a nonsimply connected polygon with sites in the interior of the polygon (a) The convex decomposition with each site located on some convex piece. (b) The resulting partition.*

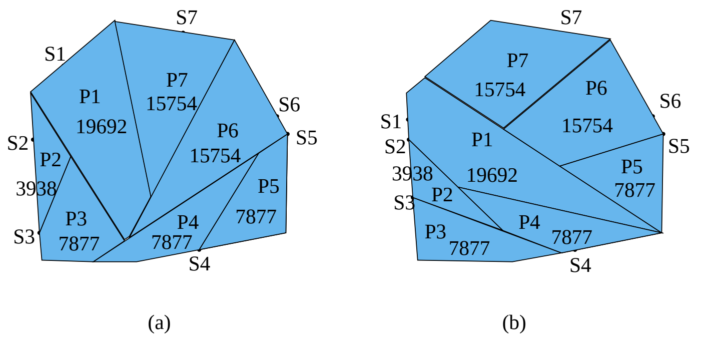

Figure 20: *(a) The partition that results for the initial polygon shown in Figure 16(a) when the vertices and sites in $W(\mathcal{P})$ are rotated in the ordering. (b) Using the same ordering of the vertices as in Figure 16(a) but a different distribution of the sites a much different partition results.*

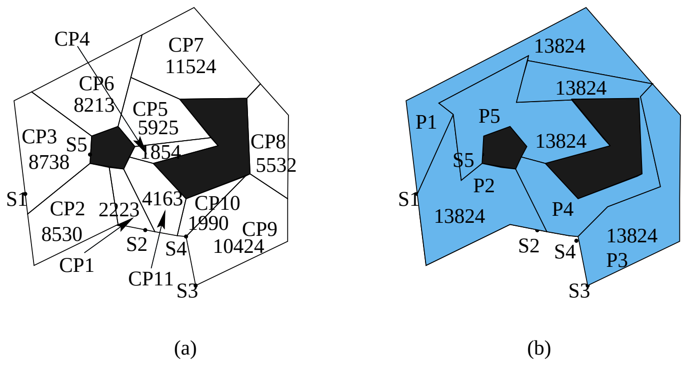

Figure 21: *(a) The convex pieces shown in Figure 18 are reordered (b) The resulting partition. Note that P_4 is again quite narrow but not disconnected.*

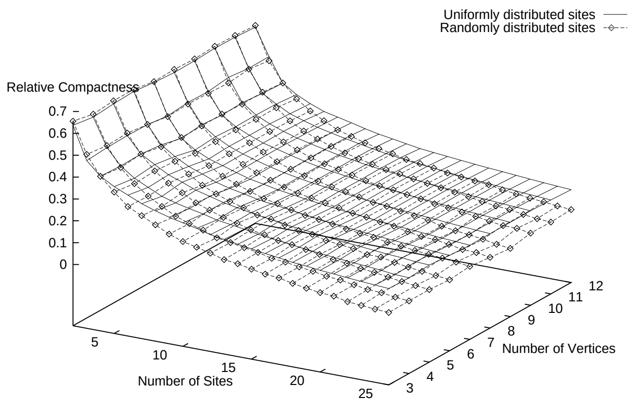

Figure 22: *Average relative compactness of the partition polygons over a set of random convex initial polygons. The solid lines show relative compactness for sites distributed uniformly around the perimeter; dashed lines show the same data for sites distributed randomly.*

In general, the area partition of the polygon $\mathcal{P}$ constructed by our algorithm is only one of many possible area partitions. Among other things, it is dependent upon the shape and ordering of the pieces of the convex decomposition $\mathcal{CP}_1, \ldots, \mathcal{CP}_p$. It may be possible, with a different initial decomposition, a different ordering, or even a different algorithm, to construct an area partition that results in fewer degeneracies, fewer reflex vertices, or more compact partition polygons.

There are many other types of area partitions one may consider. For example, constructing an unanchored area partition that is not constrained by a set of sites is easily done by a plane-sweeping method when $\mathcal{P}$ is convex and simply connected. Such a simple method does not generally work for a nonconvex or nonsimply connected polygon. A partition unconstrained by sites can be constructed using the algorithm presented here by simply choosing a set of points on the polygon with the desired area requirements to act as sites. The distribution of sites greatly affects the characteristics of the partition polygons created by our algorithm. Though empirical evidence suggests that a uniform distribution of the sites around the perimeter of a convex polygon generally results in more compact partition polygons, it is unclear how best to choose the site locations for an arbitrary polygon. This is an important consideration for the robotics setting we consider, as robots generally operate more efficiently in more simply shaped regions.

Acknowledgements

Thanks to Greg Sharp and Andrew Prock for helpful discussions and input on earlier drafts of this paper.

References

[1] B. Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4:109–140, 1989.

[2] Ta. Asano and Te. Asano. Minimum partition of polygonal regions into trapezoids. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 233–241, 1983.

[3] B. S. Baker, E. Grosse, and C. S. Rafferty. Nonobtuse triangulation of polygons. *Discrete Comput. Geom.*, 3:147–168, 1988.

[4] M. Bern, S. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 221–230, 1994.

[5] V. G. Boltianskii. *Equivalent and Equidecomposible Figures.* Topics in Mathematics. D. C. Heath and Company, Boston, 1963.

[6] B. Chazelle. *Computational Geometry and Convexity.* PhD thesis, Yale University, 1980.

[7] B. Chazelle and D. Dobkin. Decomposing a polygon into its convex parts. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 38–48, 1979.

[8] Chiuyuan Chen and Ruei-Chuan Chang. On the minimality of polygon triangulation. *BIT*, 30:570–582, 1990.

[9] I. T. Christou and R. R. Meyer. Optimal equi-partition of rectangular domains for parallel computation. Report MP-TR-95-04, University of Wisconsin, Madison, 1995.

[10] L. De Floriani and E. Puppo. An on-line algorithm for constrained Delaunay triangulation. *Comput. Vision Graph. Image Process.*, 54:290–300, 1992.

[11] P. A. Devijver and S. Maybank. On the computation of the Delaunay triangulation of a convex polygon. In *Proc. 6th IEEE Internat. Conf. Pattern Recogn.*, pages 420–422, 1982.

[12] H. Edelsbrunner, T. S. Tan, and R. Waupotitsch. An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 44–52, 1990.

[13] H. Eves. *A Survey of Geometry Volume 1*, volume 1. Allyn and Bacon, Inc., Boston, 1963.

[14] G. Fejes Tóth and W. Kuperberg. A survey of recent results in the theory of packing and covering. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 251–279. Springer-Verlag, 1993.

[15] D. H. Greene. The decomposition of polygons into convex parts. In F. P. Preparata, editor, *Computational Geometry*, volume 1 of *Advances in Computing Research*, pages 235–259. JAI Press, London, England, 1983.

[16] S. Hert, S. Tiwari, and V. Lumelsky. A terrain-covering algorithm for an AUV. *Journal of Autonomous Robots*, 3:91 – 119, 1996.

[17] S. Hertel and K. Mehlhorn. Fast triangulation of simple polygons. In *Proceedings of Conference on Foundations of Computational Theory*, pages 207–218, New York, 1983. Springer-Verlag.

[18] S. Kannan and D. Soroker. Tiling polygons with parallelograms. *Discrete Comput. Geom.*, 7:175–188, 1992.

[19] J. M. Keil. Decomposing a polygon into simpler components. *SIAM J. Comput.*, 14:799–817, 1985.

[20] C. Kenyon and R. Kenyon. Tiling a polygon with rectangles. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 610–619, 1992.

[21] C. Levcopoulos and A. Lingas. Bounds on the length of convex partitions of polygons. In *Proc. 4th Conf. Found. Softw. Tech. Theoret. Comput. Sci.*, volume 181 of *Lecture Notes in Computer Science*, pages 279–295. Springer-Verlag, 1984.

[22] A. Lingas. The power of non-rectilinear holes. In *Proc. 9th Internat. Colloq. Automata Lang. Program.*, volume 140 of *Lecture Notes in Computer Science*, pages 369–383. Springer-Verlag, 1982.

[23] W. Lipski. Finding a Manhattan path and related problems. *Networks*, 13:399–409, 1983.

[24] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4):462 – 472, 1990.

[25] B. J. Oommen, S. S. Iyengar, N. S. V. Rao, and R.L Kashyap. Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case. *IEEE Journal of Robotics and Automation*, RA-3(6):672 – 681, 1987.

[26] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.

[27] J. O'Rourke and K. J. Supowit. Some NP-hard polygon decomposition problems. *IEEE Trans. Inform. Theory*, IT-30:181–190, 1983.

[28] W. Page and K. R. S. Sastry. Area-bisecting polygonal paths. *Fibonacci Quarterly*, 30(3):263–373, 1992.

[29] N. S. V. Rao and S. S. Iyengar. Autonomous robot navigation in unknown terrains: Incidental learning and environmental exploration. *IEEE Transactions on System, Man and Cybernetics*, 20(6):1443 – 1449, 1990.

[30] V. Soltan and A. Gorpinevich. Minimum dissection of rectilinear polygon with arbitrary holes into rectangles. *Discrete Comput. Geom.*, 9:57–79, 1993.

[31] S. B. Tor and A. E. Middleditch. Convex decomposition of simple polygons. *ACM Trans. Graph.*, 3:244–265, 1984.