To appear in *Advanced Robotics*, Special Issue on Mechanical Assembly

Assembly Partitioning along Simple Paths: the Case of Multiple Translations[*]

Dan Halperin
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

Randall H. Wilson
Intelligent Systems and Robotics Center
Sandia National Laboratories
Albuquerque, NM 87185-1008

August 28, 1996
Revised December 2, 1996

Abstract

We consider the following problem that arises in mechanical assembly planning: given an assembly, identify a subassembly that can be removed as a rigid object without disturbing the rest of the assembly. This is the *assembly partitioning* problem. Specifically, we consider planar assemblies of simple polygons and subassembly removal paths consisting of a single finite translation followed by a translation to infinity. Such paths are typical of the capabilities of simple actuators in fixed automation and other high-volume assembly machines. We present a polynomial-time algorithm to identify such a subassembly and removal path or report that none exists. In addition, we extend this algorithm and analysis to removal paths consisting of a small number $k > 2$ of translations. We discuss extending the algorithm to 3D and to other types of motions typical in non-robotic automated assembly.

1 Introduction

Fixed automation or special-purpose assembly machines can achieve very high throughput, often down to cycle times of one product per second for synchronous assembly machines and

[*]Work on this paper by the first author has been supported by a grant from the Stanford Integrated Manufacturing Association (SIMA), by NSF/ARPA Grant IRI-9306544, and by NSF Grant CCR-9215219. This research was performed while the second author was at Stanford University, and has been supported by the Stanford Integrated Manufacturing Association (SIMA).

similar systems, and even faster for fixed automation. For this reason, they are often chosen over general-purpose robots for assembly of high-volume products. However, designing such an assembly system for a given product is a complex process, often requiring eight months or more from the time prototype parts are available [5]. Reducing this lead time would allow faster time-to-market with lower cost for many high-volume products.

This paper generalizes assembly planning techniques originally developed for robotic and general-purpose assembly to apply to motions consisting of two translations, which are typical of high-volume assembly systems. Such systems use simple actuators and mechanical drive systems to produce motions having a few degrees of freedom. For instance, a standard module for the Bodine Model 64 synchronous assembly machine produces motions that acquire a part, translate it to an intermediate point, then insert it. Pick-and-place machines composed of two linear actuators also produce two-translation motions.

The assembly operations required by industrial products are almost exclusively *monotone two-handed* [15, 23], meaning that each operation places a part or rigid subassembly into its final position relative to another subassembly. A sequence of assembly operations that builds a product from its individual parts is an *assembly sequence*. If such a sequence of monotone two-handed operations exists for a product, then we say the product is monotone two-handed. In the rest of this paper, we only consider monotone two-handed assembly sequences for assemblies of rigid parts.

For example, the assembly in Figure 1(a) can be assembled by a monotone two-handed assembly sequence involving only translations: the two small parts are placed together in final relative position, then this subassembly is moved down and to the right to insert into the largest part. On the other hand, the assembly in Figure 1(b) is not monotone two-handed. It can be assembled by placing the smallest part inside the largest but off to the left, then placing the medium-sized part, and then moving the smallest part into its final position. Because the small part moves twice, this assembly sequence is not monotone. The assembly could also be constructed by a single three-handed operation moving the two smaller parts independently (the table holding the stationary parts is considered a "hand"). Two-handed and monotone assembly sequences are much preferred, especially for high-volume products as considered here.

Since the most constraints on assembly are present in the assembled state, assembly sequences are often generated in reverse. Then assembly sequencing is reduced to the *partitioning problem*: given an assembly, determine a proper subset of the parts that can be removed (as a single rigid object) without disturbing the other parts. Recursing on the resulting two subassemblies generates an assembly sequence.

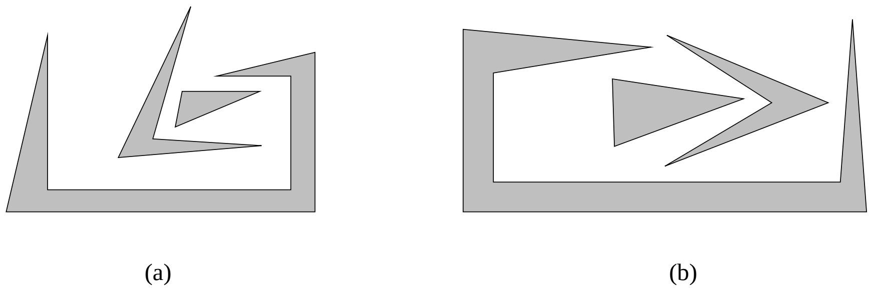

Figure 1: Assemblies requiring non-straight-line motions for disassembly. (a) is a monotone two-handed assembly while (b) is not.

Past work has shown that when a disassembly motion may consist of any number of translations, the partitioning problem (and thus sequencing itself) is NP-complete [12]. On the other hand, polynomial-time partitioning is possible when the motions are limited to single translations to infinity [2, 22].

This paper generalizes the *non-directional blocking graph* (or *NDBG*) of [22] to an intermediate case: motions consisting of a small number of translations. We consider the following problem: given a planar assembly of simple polygons, identify a subassembly that can be removed as a rigid object by a motion consisting of a finite translation followed by a translation to infinity. We present an algorithm that solves this problem in $O(n^{1.38}N^6)$ time, where n is the number of parts in the assembly and N is the total number of vertices of the polygons. We then extend this algorithm to partitioning assemblies with a small number $k > 2$ of translations.

The rest of the paper is organized as follows. Section 2 surveys related work. In Section 3 we present the basic necessary terminology together with an initial analysis of our solution approach. Section 4 gives the algorithm to partition an assembly of polygons with two translations, and analyzes its computational complexity. We extend the algorithm and analysis to the case of a bounded number $k > 2$ of translations in Section 5. Finally, Section 6 discusses extending the algorithm to three dimensions, and considers other simple motions typical of high-volume assembly that would yield to a similar approach.

2 Related Work

For an overview of geometric separation problems see Toussaint [20]. Pollack, Sharir, and Sifrony [17] give an efficient algorithm to separate two simple polygons in the plane with a sequence of translations.

Polynomial-time solutions to the assembly partitioning problem exist for certain types of restricted disassembly motions. Arkin, Connelly, and Mitchell [2] give a polynomial-time algorithm for partitioning an assembly of polygons in the plane with a single infinite translation. Wilson and Schweikard [18] give a corresponding result for polyhedral assemblies. When a small number of translation directions is given, Agarwal *et al.* [1] give a more efficient algorithm to determine a linear ordering of part removals to disassemble a collection of polyhedra.

Wilson and Latombe [22] present a framework for solving partitioning problems with restricted motions, considering in particular disassembly motions consisting of either (a) infinitesimal translations and rotations or (b) infinite translations. In the infinitesimal case, a subassembly is identified that can move a very small distance, which is only a necessary condition on a removal path. A more efficient and more practical algorithm for partitioning with infinitesimal motions is given in [8].

Wilson *et al.* [21] extend the framework of [22] to arbitrary removal motions. However, Kavraki and Kolountzakis [12] show that for assemblies of polygons in the plane the problem of partitioning for arbitrary removal motions is NP-complete.

This paper addresses an intermediate case for assemblies of simple polygons in the plane. In this intermediate case the number of translations allowed to remove a subassembly is greater than one, but bounded by some constant k. We first consider the case where $k = 2$, and then consider the case $k > 2$.

3 Preliminaries: Terminology and Initial Analysis

In this section we present some of the terminology that we will be using throughout the paper, together with an initial analysis of the problem we are considering.

3.1 Arrangements of Curves and of Surfaces

An *arrangement* of curves in the plane is the subdivision of the plane induced by these curves. Consider, for example, the arrangement induced by a collection of m lines in the plane. An arrangement of lines partitions the plane into vertices, edges and faces. A vertex is an intersection point of two lines, an edge is a maximal connected portion of a line that does not meet any vertex, and a face is a maximal connected region of the plane not meeting any edge or vertex. If we assume that the lines are in general position, namely, no two are parallel and no more than two meet at a single point, then it can be shown that the number of vertices (as well as the number of edges or faces) in the arrangement is $\Theta(m^2)$.[1] Similarly, in three dimensions, we may consider an arrangement of m planes, which partitions space into vertices, edges, faces and cells.

Arrangements are defined for non-linear objects as well, and the concept extends to higher dimensional Euclidean spaces. We will discuss arrangements of (hyper)surfaces or surface patches in d-dimensional Euclidean space, where the surfaces are assumed to be algebraic of bounded maximum degree, and bounded by algebraic surfaces of maximum constant degree. See [6] for detailed discussions on arrangements of lines and of hyperplanes in higher dimensional spaces. For discussion of arrangements of curves and surfaces (not necessarily linear) see, e.g., [7, 10]; in particular, the thesis [10] focuses on arrangements of surfaces in 3-space that are induced by motion planning problems and are closely related to the arrangements discussed in Section 4 of this paper.

We will often refer to two quantities regarding arrangements of m surfaces in d-space: The complexity of the entire arrangement, and the time complexity of computing the entire arrangement. The complexity of an arrangement of m surface patches in d-space is defined as the overall number of cells of dimension k, $0 \leq k \leq d$, in the arrangement. It is well known that the maximum complexity $K_d(m)$ of an arrangement of m surface patches in d-space as defined above is $\Theta(m^d)$ (see, e.g., [16]).

As for computing an arrangement of m surfaces in d-space, we should first be more explicit as to what we mean by "computing" an arrangement. For our purposes, we need a way to visit all cells in the arrangement, at each step moving from a cell to one of its neighbors. We will denote the worst-case running time of the best known algorithm for solving this problem by $T_d(m)$, where d is the dimension of the space, and the maximum is taken over all arrangements of m surfaces as defined above. For arrangements of lines in the plane, the running time of such an algorithm is $\Theta(m^2)$ (see, e.g., [6]). In fact, $T_2(m)$ is roughly quadratic, i.e., there is an algorithm with near-quadratic running time to compute the arrangement induced by any collection of m algebraic curves of bounded degree. In higher dimensions, there are also algorithms to solve this problem with running time that is close to the worst case combinatorial complexity of the entire arrangement. Basu *et al.* [3] give an algorithm to find a point in every cell in a d-dimensional arrangement, in time $O(m^d)$, where d is fixed, and where we assume that the algebraic degree of the surfaces is bounded (their result explicitly handles the dependence on this degree). It may be

[1]The notation $\Theta(f(n))$ states that the quantity $Q(n)$ in question is $O(f(n))$, and that the bound is tight. To be exact, there exist constants $n_0 > 0, C_1 > 0$ and $C_2 > 0$ such that the quantity $Q(n)$ is bounded above by $C_1 f(n)$ and bounded below by $C_2 f(n)$ for $n > n_0$.

more convenient for our purpose, though, to have an algorithm that produces these sample points in a systematic way, namely, in a way that for most pairs of successive points that the algorithm produces they lie in adjacent cells. This can be done with roughly the same running time, using a simple spatial sweep algorithm [4]. In any case, we will use the notation $T_d(m)$ as defined above rather than be specific about this bound.

The computation of subdivisions and their traversal are well studied topics in computational geometry. Hence we will not go into the full details of such algorithms. We refer the interested reader to the book [19] and the papers [4, 9] for more details.

3.2 The Minkowski Sum of Two Polygons, Envelopes and Shadows

Let P and Q be two sets in R^2. The *Minkowski sum* (or vector sum) of P and Q, denoted $P \oplus Q$, is the set $\{p+q \mid p \in P,\ q \in Q\}$. We will also use the notation $P \ominus Q = P \oplus (-Q) = \{p - q \mid p \in P,\ q \in Q\}$. Choose a reference point r rigidly attached to Q, and suppose that Q is placed such that the reference point coincides with the origin. Then $P \ominus Q$ is the loci of placements of the reference point where $P \cap Q \neq \emptyset$. The Minkowski sum is a useful concept in robot motion planning and related areas [14], often called the *configuration space obstacle*, or *C-obstacle*: the set Q will collide with P under a rigid translational motion along a path t exactly when the reference point r, moved along t, intersects $P \ominus Q$.

For the assembly partitioning problem that we are considering, we will confine ourselves to the Minkowski sum of polygonal sets, which is itself a polygonal set. Let P and Q be two simple polygons, and let u and v denote the number of vertices in P and Q respectively. It is well known that the combinatorial complexity of the boundary of $P \ominus Q$, namely the overall number of edges and vertices on the boundary of $P \ominus Q$, is $O(u^2v^2)$, and this bound is tight in the worst case (see, e.g., [11]). However, for our purposes, we only need to know the *outer boundary* of $P \ominus Q$. Since it is defined by at most $O(uv)$ segments, its complexity is at most $O(uv\alpha(uv))$ [17], where $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function. Figure 2(a) shows two polygons P and Q, and Figure 2(b) their Minkowski sum $P \ominus Q$.

In our analysis, we will consider portions of the Minkowski sum of two polygons, which we will refer to as *envelopes*. Let M denote the outer boundary of $P \ominus Q$. The boundary M is a simple polygon. We define the *central envelope* of M, $E_C(M)$, to be the collection of points of M first hit by rays from the origin. More precisely, if ρ is a ray from the origin, we let $F(\rho, M)$ be the intersection point of ρ and M which is nearest to the origin; $F(\rho, M)$ is undefined if $\rho \cap M = \emptyset$. Then, $E_C(M)$ is the union of $F(\rho, M)$ over all rays through the

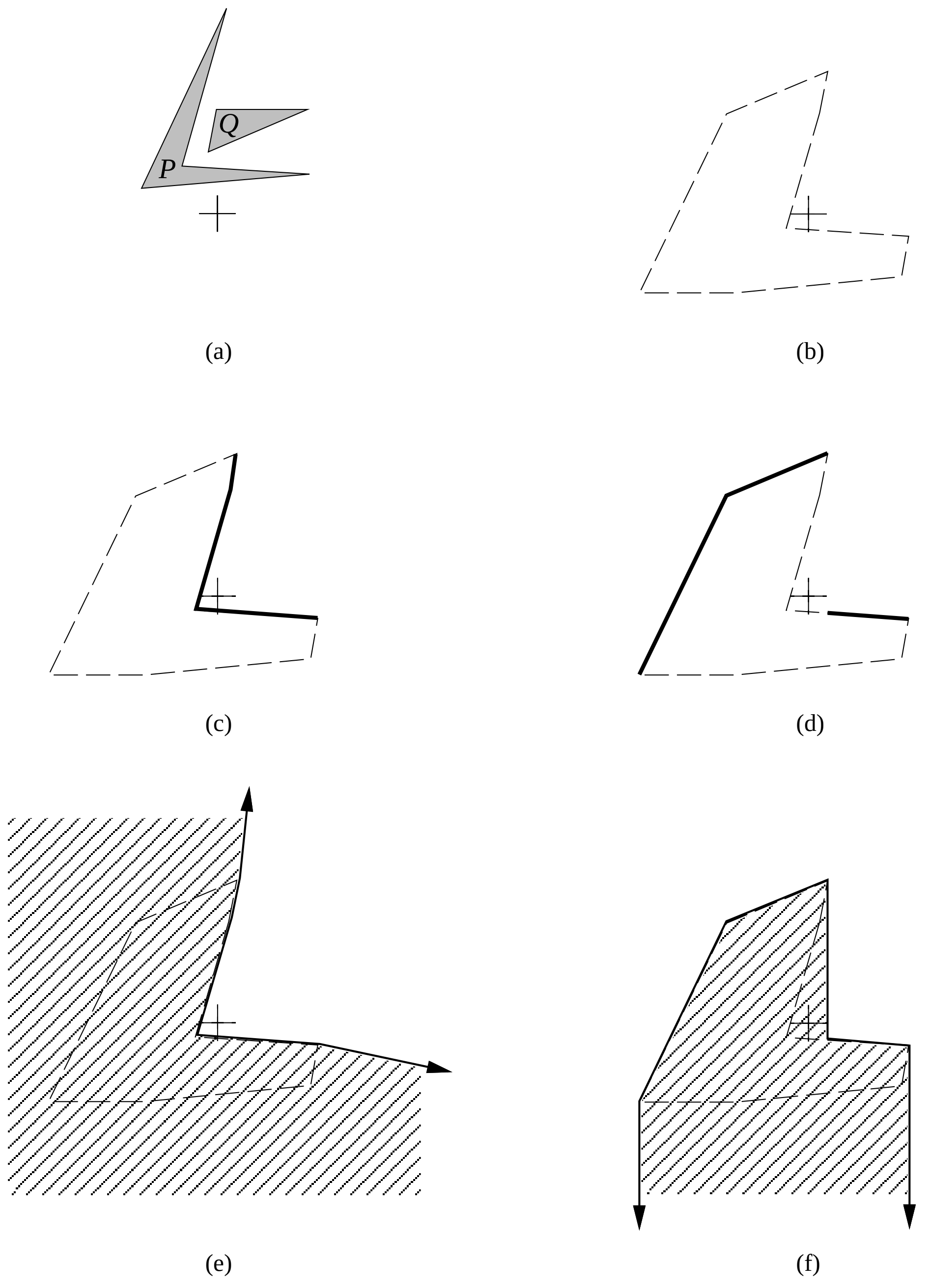

Figure 2: Some geometric preliminaries: (a) Two polygons P and Q, (b) the boundary of the Minkowski sum of P and $-Q$, $M = bd(P \ominus Q)$, (c) the central envelope $E_C(M)$, (d) the ϕ-upper envelope $E_{90^\circ}(M)$, (e) the central shadow $S_C(M)$, and (f) the ϕ-shadow $S_{90^\circ}(M)$.

origin where $F(\rho, M)$ is defined. Figure 2(c) shows the central envelope for the polygon in Figure 2(b).

Next we define the *ϕ-upper envelope* of M. Let λ be a directed line, whose angle with the positive x-direction is ϕ. Let $G(\lambda, M)$ be the farthest point of intersection of λ and M in the direction of λ, if they indeed intersect. The ϕ-upper envelope of M, $E_\phi(M)$ is defined to be the union of $G(\lambda, M)$ over all lines λ whose angle with the positive x-direction is ϕ, whenever $G(\lambda, M)$ is defined. If ϕ is $90°$, then our definition of $E_\phi(M)$ is similar to the standard definition of upper envelopes (see, e.g., [7]). Figure 2(d) shows the ϕ-upper envelope of the polygon in Figure 2(b) for $\phi = 90°$.

Corresponding to each of the two envelopes, we now define a *shadow*. The *central shadow* of M, $S_C(M)$, is the collection of points in the plane such that the line connecting them to the origin intersects the central envelope $E_C(M)$. Similarly, the *ϕ-shadow* of M, $S_\phi(M)$, is the union of rays whose angle with the positive x-axis is $\phi + 180°$ and whose termini lie on the ϕ-upper envelope $E_\phi(M)$. Figures 2(e) and 2(f) show the central shadow and $90°$-shadow, respectively, of the polygon in Figure 2(b).

We will use the central shadow and ϕ-shadow of $P \ominus Q$ to reason about relative translations of Q with respect to P. For an initial translation (x, y) of Q, the point (x, y) is inside the central shadow $S_C(P \ominus Q)$ exactly when Q collides with P along the translation (x, y). When Q is already at position (x, y) and translated infinitely in direction ϕ, it collides with P if and only if (x, y) is inside the ϕ-shadow $S_\phi(P \ominus Q)$.

Let P_i and P_j be two parts in an assembly A, with n_i and n_j vertices respectively. As mentioned above the complexity of the outer boundary of $P_i \ominus P_j$ is $O(n_i n_j \alpha(n_i n_j))$. The same bound also holds for each of the envelopes we have defined, and for the boundary of the corresponding shadows of $P_i \ominus P_j$. Now, the outer boundary of $P_i \ominus P_j$, as well as the two envelopes lie each in the union of a collection S of $O(n_i n_j)$ line segments—the collection of segments underlying the Minkowski sum. Each of these segments is either the Minkowski sum of an edge of P_i and a vertex of $-P_j$ or the Minkowski sum of a vertex of P_i and an edge of $-P_j$. The boundary of each of the shadows also lies in a collection of $O(n_i n_j)$ segments, which is the collection S augmented with additional rays extended from endpoints of some of the segments in S.

For the subsequent analysis, which aims at worst-case asymptotic time bound for the algorithm that we present, it is sufficient to deal with the appropriate raw collection of segments rather than with the boundary of $P_i \ominus P_j$. Hence, we state the following Lemma.

Lemma 3.1 *Let P_i and P_j be two polygonal parts with n_i and n_j vertices respectively. The* *outer boundary M of $P_i \ominus P_j$, the central envelope of M, the ϕ-envelope of M for any fixed ϕ, the boundary of the central shadow and the boundary of the ϕ-shadow for any fixed ϕ, each lies in the union of $O(n_i n_j)$ segments.*

3.3 Partitioning along a Single Path

Consider an assembly A of non-overlapping simple polygons in the plane. We will call the polygons *parts*, and will use the term *subassembly* loosely, to refer to a set of parts as well as their union. In general, we wish to identify a proper subassembly $S \subset A$ that can be completely separated from $A \setminus S$ [2] by a collision-free rigid motion along a continuous path t. In this paper, we will confine ourselves to paths that are the concatenation of a small number k of straight line segments.

Now, consider which subassemblies of A could follow a given rigid motion t. Since a subassembly occupies space equal to the union of its parts, the motion t causes a collision between a subassembly S and $A \setminus S$ if and only if t causes a collision between some part in S and some part in $A \setminus S$.

Let S be a subassembly removable along t. If a part Q moved along t collides with another part P (left stationary), then we say that P *blocks* Q along t. If P blocks Q, then either P must be in the moved subassembly S or Q must not be in S. These constraints on membership in S can be easily represented with a *blocking graph* [22]. The blocking graph of A for motion t, written $G_A(t)$, is a directed graph with a node for each part of A and an arc from node Q to node P exactly when Q is blocked by P along t. A subassembly S can be removed with rigid motion t if and only if no arcs in $G_A(t)$ connect nodes in S to nodes in $A \setminus S$. Such a subassembly exists exactly when $G_A(t)$ is not strongly connected.[3]

The set of positions in which a part Q collides with part P is given by $P \ominus Q$, i.e., the Minkowski sum of P and $-Q$ (see Subsection 3.2). Part Q will collide with P under a rigid motion t exactly when the reference point of Q, moved along t, intersects $P \ominus Q$. To find the collisions between all pairs of parts for a single motion, all pairwise differences can be computed, taking the origin of the coordinate system to be the common reference point for all the parts. When the pairwise differences are superimposed, they partition the plane into regions, within which the set of pairs of parts colliding is fixed. A rigid motion t causes the reference point to move through a sequence of regions, collecting constraints for the

[2]Here "\" denotes the set subtraction operator.

[3]A *strongly connected component* (or *strong component*) of a directed graph is a maximal subset of nodes such that for any pair of nodes (n_1, n_2) in this subset, a path connects n_1 to n_2. A graph is strongly connected if it consists of one strong component.

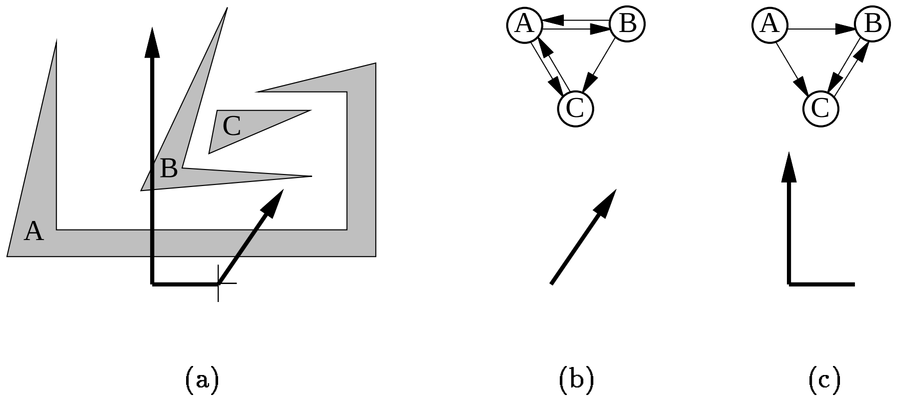

Figure 3: An assembly and its blocking graphs for two translational motions

symbol	*description*	*defined in*
n	number of parts in the assembly A	
N	overall number of features of parts in the assembly A	
$K_d(m)$	complexity of a d-dimensional arrangement of m surfaces	Subsection 3.1
$T_d(m)$	time to compute a d-dimensional arrangement of m surfaces	Subsection 3.1
$P \ominus Q$	the Minkowski sum of P and $-Q$	Subsection 3.2
$S_C(M)$	the central shadow of a polygon M	Subsection 3.2
$S_\phi(M)$	the ϕ-shadow of a polygon M	Subsection 3.2
$G_A(t)$	the blocking graph for assembly A along path t	Subsection 3.3

Table 1: Summary of notation

blocking graph of t. For further discussion see [21].

Figure 3 shows an assembly and the blocking graphs for two motions, one a translation up to the right, and the other a translation to the left then upward. For instance, part B collides with part C when translated up right, so the constraint $B \rightarrow C$ is present in the corresponding blocking graph. The blocking graph of the up-right translation is strongly connected, and in fact no subassembly can be removed along that path. For the two-step motion, the blocking graph is not strongly connected. Instead, there are no outgoing arcs from $\{B, C\}$ to $\{A\}$, so B and C may be removed rigidly in a motion to the left then upward.

We summarize our notation in Table 1.

4 Partitioning with Two Translations

Given an assembly A of simple polygons, we wish to determine whether any subassembly S of A can be completely separated from the subassembly $A \setminus S$ by a finite translation of S followed by an infinite translation of S. We describe a rigid motion consisting of two translations by a triple $t = (x, y, \phi)$, where (x, y) is the displacement caused by the first translation, and ϕ is the direction of the second (infinite) translation. Since any motion (x, y, ϕ) will move the subassembly to infinity in direction ϕ, we need only ensure that the motion is collision-free with the complement of the moved subassembly.

To solve the problem, we partition the (x, y, ϕ)-space into cells such that the set of subassemblies that can follow a motion is fixed for all motions in a cell. First we derive the constraints arising from the first translation (x, y), and then the constraints arising from a second translation given by ϕ. Then we combine these sets of constraints to obtain the final three-dimensional arrangement, i.e., the subdivision of the (x, y, ϕ)-space.

4.1 The First Translation

Consider now the constraints on subassemblies that can follow the first translation (x, y) without collision, as x and y vary. The origin of the plane represents the null translation $(0, 0)$, and we wish to calculate the critical curves in the plane at which the blocking graph for the first translation changes.

For any two parts P and Q, the set of placements of Q for which Q intersects P is the polygonal C-obstacle $C = P \ominus Q$. However, not all edges of C are critical curves; once a translation enters C, the constraint $Q \rightarrow P$ will be present in the blocking graph for all translations further in the same direction. This leads us to compute the central shadow of C, $S_C(C)$. The edges of the central shadow $S_C(P \ominus Q)$ are (potential) critical curves for the first translation: the arc $Q \rightarrow P$ is present in blocking graphs for just those motions $(x, y) \in \text{int}(S_C(P \ominus Q))$ ending inside the shadow.[4]

We now superimpose the central shadows $S_C(P_i \ominus P_j)$ for all pairs of parts (P_i, P_j). The boundary edges of the shadows determine a subdivision of the plane into regions (an arrangement of segments), such that for all points (x, y) inside each region, $G_A((x, y))$ is fixed.

[4]We say *potential* critical curves because we only care about edges where the transitive closure of the blocking graph changes, which is possibly a subset of the above edges. It might be the case, that considering only actual critical curves, that is, only edges where the transitive closure of the blocking graph changes, could lead to better bounds.

Figure 4 shows the C-obstacles (dotted lines) and the boundaries of the corresponding central shadows (solid) for each pair of parts of the assembly in Figure 3. B/C is the obstacle for moving part B and stationary part C; the obstacle C/B is identical to B/C, rotated 180 degrees around the origin. The full arrangement for the first translation is shown at the bottom of Figure 4.

To interpret this arrangement, consider the path in Figure 3c, whose first displacement is a finite translation to the left. This displacement (call it d_1) is marked as $*$ in Figure 4. The line segment from the origin to $*$ crosses the boundaries of central shadows A/B, A/C, and C/B. This means that if A follows d_1, then B and C must move with it, and likewise if C follows d_1, B must move with it. Hence two subassemblies can follow d_1 without collision: B alone or $\{B, C\}$. A similar analysis for any point in the arrangement yields the subassemblies that can follow the displacement corresponding to that point.

How many segments are there in this arrangement? Let the assembly A have n parts, and assume first that each part has at most some fixed number of vertices. (At this point we wish to emphasize the effect of the *number* of parts on the complexity. Below, in Subsection 4.4 we will give a refined analysis.) Thus the total complexity of the assembly is $O(n)$. By our assumption, the C-obstacle for two parts can have an outer boundary with at most some fixed (constant) number of vertices, and the same holds for the boundary of the central shadow of such a polygon. For the consequent analysis, we need to know the maximum overall number of segments in the arrangement, which is $O(n^2)$ (Lemma 3.1). We denote this set of $O(n^2)$ segments by S_1.

4.2 The Second Translation

We now concentrate on the second, infinite translation. Such a translation is in fact a translation along a ray and it can be specified by three parameters (x, y, ϕ), where (x, y) is the starting point of the ray and ϕ is its direction. We wish to partition the (x, y, ϕ)-space into cells such that the set of subassemblies that can follow a motion along a ray is fixed for all the rays represented by the points in a cell. Note that, for the moment, we ignore the effect of the first translation on this subdivision.

We will define a collection of critical surfaces that induce the desired subdivision. We start by fixing a direction ϕ_0 and considering a two-dimensional cross-section of the three-dimensional space (x, y, ϕ), at ϕ_0. At the fixed ϕ_0, the critical curves are defined similar to the subdivision of the first translation, only this time we consider the ϕ_0-shadows of the C-obstacles rather than the central shadows. It is evident that inside a ϕ_0-shadow of a C-

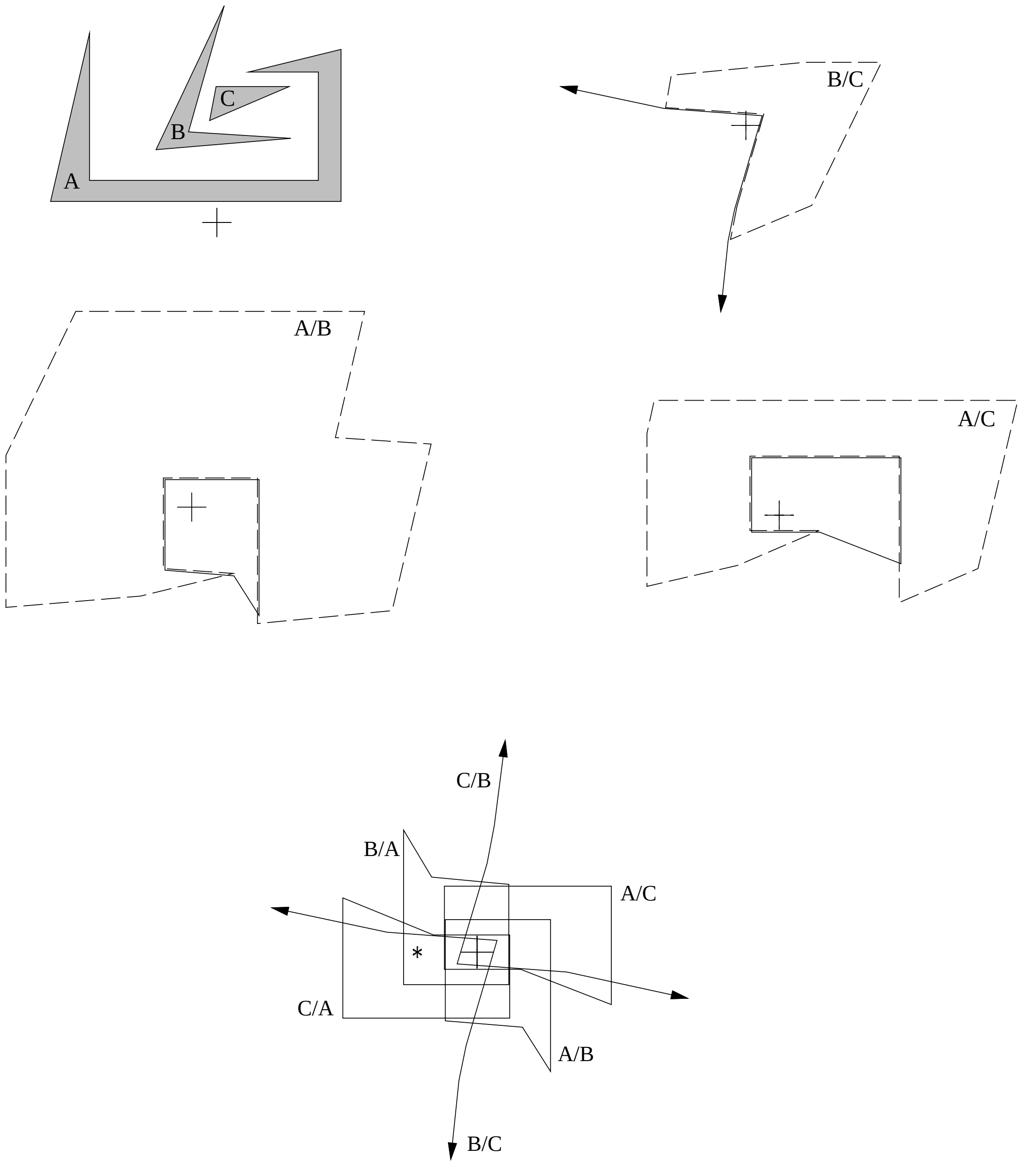

Figure 4: An assembly, central shadows for pairs of parts, and the arrangement for the first translation. The point marked by $*$ corresponds to a translation to the left (in this case the first displacement of the path in Figure 3c)

obstacle, a new constraint is added, that does not exist out of that shadow. The collection of critical curves that determine the subdivision of the ϕ_0-cross-section are the boundaries of the ϕ_0-shadows of the C-obstacles.

To get the three-dimensional critical surfaces, we let ϕ vary, and let the boundaries of the ϕ-shadows vary accordingly. The collection of critical surfaces is defined to be the union of the ϕ-shadow boundaries for all $\phi \in [0, 2\pi)$. However, to simplify the analysis, we will consider a superset of the above collection. Namely, for every C-obstacle D, we will consider the surfaces traced by the set $S(D)$ of all the segments underlying the C-obstacle D (see Subsection 3.2) as ϕ varies, together with surfaces traced by rays in the $\phi + 180^\circ$ direction extended from endpoints of segments in $S(D)$ (that is, at a fixed ϕ we extend a ray from an endpoint of a segment in $S(D)$ in the $\phi + 180^\circ$ direction, and let this ray change as ϕ changes; we consider the surface swept by this ray to be an additional critical surface and repeat this for every endpoint of any segment in $S(D)$). The resulting collection of surfaces is clearly a superset of the critical surfaces traced by the ϕ-shadow boundaries. We denote this superset of critical surfaces by $\mathcal{S}_2$.

By the assumptions in Subsection 4.1, the number of surfaces induced by each C-obstacle is bounded by a constant. Since there are $O(n^2)$ C-obstacles (one for every pair of original parts), we conclude that $\mathcal{S}_2$ consists of $O(n^2)$ critical surfaces, that partition the space (x, y, ϕ) into non-critical cells such that for all infinite translations (x, y, ϕ) inside each cell, the blocking graph $G_A((x, y, \phi))$ is fixed.

Figure 5 shows the ϕ-shadows for the assembly of Figure 3, and the resulting arrangement, for the upward translation $\phi = 90^\circ$. To understand this arrangement, consider the point marked by the $*$. This point represents an infinite translation upward starting from the x, y coordinates of the $*$. This corresponds to the second displacement d_2 of the path in Figure 3c. The $*$ is contained in the shadows A/B, A/C, B/C, and C/B. This means that if A follows d_2, then it will collide with stationary B or C, and likewise if B follows d_2 then it will collide with C, and vice versa. Importantly, the $*$ is not contained in shadows B/A or C/A, so that the subassembly $\{B, C\}$ can follow d_2 without colliding with A.

4.3 Combining the Two Translations

As stated in the beginning of this section, the triple (x, y, ϕ) can represent not only the second infinite translation, but in fact the two translations. A point (x_0, y_0, ϕ_0) represents a path of a subassembly that starts at the origin, moves to the point (x_0, y_0) along a straight line segment, and then moves to infinity along a ray in the ϕ_0 direction. We already

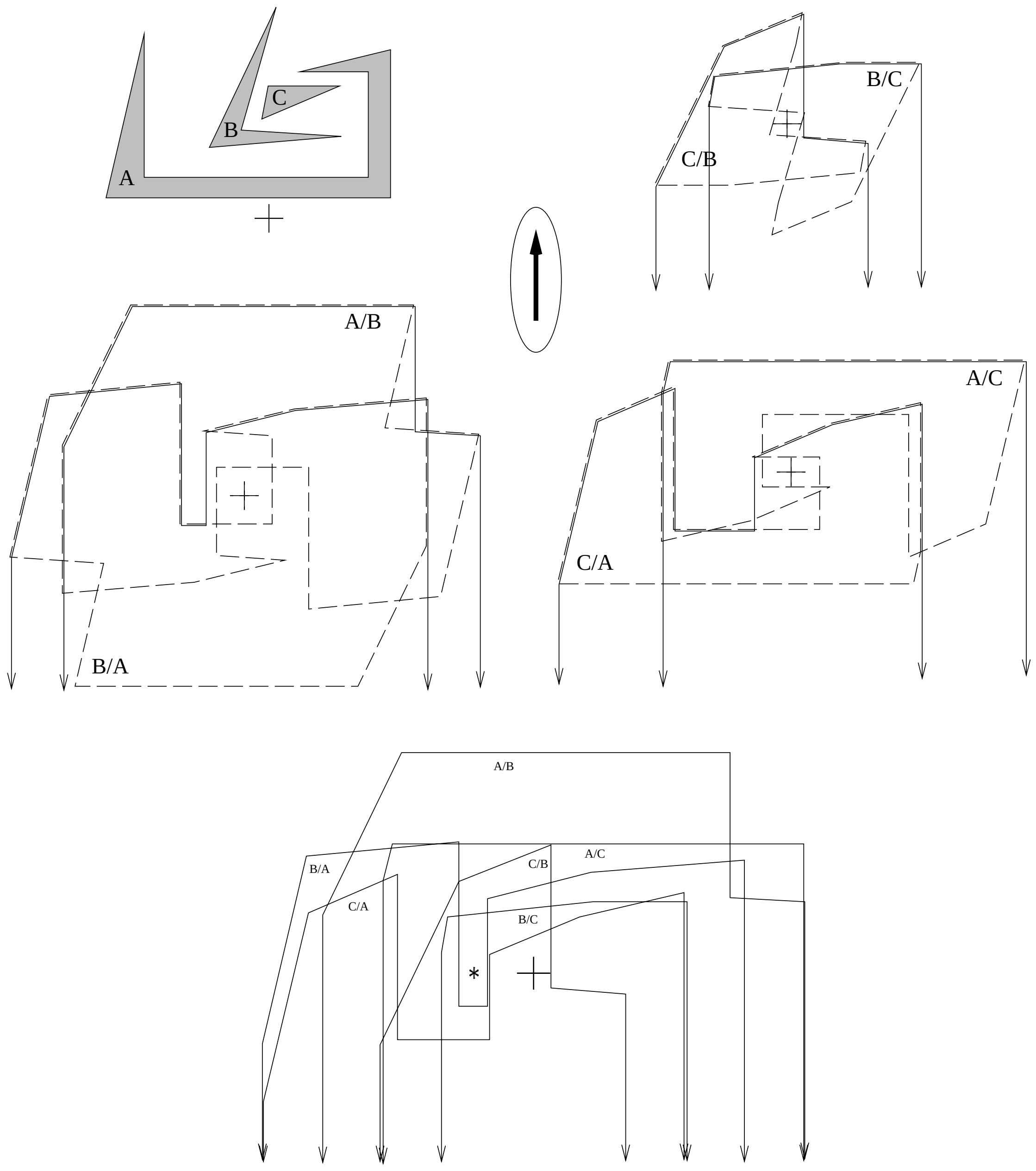

Figure 5: An assembly, ϕ-shadows for pairs of parts when $\phi = 90°$, and a 2D slice of the arrangement for the second translation at $\phi = 90°$. The point marked by $*$ corresponds to an infinite translation upward starting from the (x, y) coordinates of the $*$ (ie. the second displacement of the path in Figure 3c)

have the set $\mathcal{S}_2$ of constraint surfaces that subdivides the space (x, y, ϕ) into non-critical cells, and we now refine this subdivision according to the constraints induced by the first translation. Consider a cross-section of the (x, y, ϕ)-space at ϕ_0. If we wish to add the constraints induced by the first translation, we should simply add the segments in the set S_1 to the arrangement. But this statement is valid for any ϕ value. Therefore, we extend each segment in S_1 into a vertical "wall" (in the ϕ direction) in the (x, y, ϕ)-space. We will denote this collection of vertical walls, extended from S_1, by $\mathcal{S}_1$. Thus we have completed the subdivision of the space (x, y, ϕ) into non-critical cells, such that for any two-translation path $t = (x, y, \phi)$ in each cell the set of blocking constraints $G_A(t)$ is fixed.

The number of elements in $\mathcal{S}_1$ as well as in $\mathcal{S}_2$ is $O(n^2)$. These surfaces (or more precisely, surface patches) are clearly algebraic of bounded degree. By the discussion in Subsection 3.1 the maximum number of cells in a 3D arrangement induced by m such surfaces, $K_3(m)$, is $O(m^3)$. Hence the maximum number of cells in the above subdivision is $O(n^6)$.

Finally, to solve the original problem, i.e., to find a subassembly that can be partitioned with two translations (if one exists), we proceed as follows. We compute the subdivision of the (x, y, ϕ)-space by the surfaces in $\mathcal{S}_1 \cup \mathcal{S}_2$, using an algorithm with running time $T_3(n^2)$ (see Subsection 3.1). For each cell produced by the algorithm, we compute the blocking graph $G_A(t)$ corresponding to a representative path t for that cell. The blocking graphs for all cells in the arrangement can be computed incrementally in the following way. We begin by choosing a point $t = (x_0, y_0, \phi_0)$ in some cell c_0 of the arrangement; the blocking graph $G_A(t)$ can be easily computed by checking for inclusion of (x_0, y_0) in the central shadows and ϕ_0-shadows of the C-obstacles, in time proportional to the number of shadows, that is $O(n^2)$. (This step will be dominated by other steps of the algorithm, also in the refined analysis that we give below.) We then perform a systematic traversal of the arrangement, at each step moving from a cell to one of its neighbors. When we step through a critical surface, we either add or remove the constraint corresponding to that surface, depending on whether we are entering a shadow or leaving it.

For example, consider the path in Figure 3c, consisting of a finite translation to the left followed by an infinite upward translation. This path corresponds to the position of the $*$ in the arrangements in Figures 4 and 5 (we do not show the combined arrangement because it is too complex to be interpreted manually). Combining the shadows in which the $*$ is contained from both arrangements gives A/B, A/C, B/C, and C/B, which is represented by the blocking graph in Figure 3c. This blocking graph is not strongly connected, and therefore one of the strong components (in this case $\{B, C\}$) has no outgoing arcs and can be removed along the path.

Since at most a constant number of constraints is added or removed at each step,[5] all the blocking graphs can be computed in time proportional to the size of the arrangement, i.e., in time $O(n^6)$. Strong connectedness can be checked in time linear in the size of the graph, which in this case is bounded by n^2. Thus in an assembly of n polygons, each having at most some constant number of vertices, the total time to find a subassembly removable by two translations is

$$T_3(m) + K_3(m)O(n^2), \quad \text{where} \quad m = O(n^2).$$

We summarize the discussion above in the following theorem.

Theorem 4.1 *Given a planar assembly consisting of n disjoint simple polygons, each having at most some constant number of vertices, it can be determined in $O(n^8)$ time whether there is a subassembly that can be removed along a path consisting of a single finite translation followed by a translation to infinity. The algorithm outputs both the labels of the parts in the removable subassembly and the specifications of the path.*

4.4 Refined Analysis

In this subsection we introduce another parameter into the analysis of the running time of our algorithm, and also indicate several points where the algorithm may be improved.

Let the assembly A we wish to partition consist of n parts (as before) and let N be the total number of vertices of all the parts together. Denote the number of vertices of part $P_i \in A$, by n_i. As discussed in Subsection 3.2 (see also Lemma 3.1), the crucial parameter[6] for each C-obstacle defined by two parts P_i, P_j, is the number of segments underlying the Minkowski sum $P_i \ominus P_j$, which is $O(n_i n_j)$. Hence in both sets $\mathcal{S}_1$ and $\mathcal{S}_2$ of the analysis in this section, the overall number of surfaces is

$$\sum_{i \neq j} O(n_i n_j) = O(N^2) \; .$$

Therefore we have

Theorem 4.2 *Given a planar assembly consisting of n disjoint simple polygons, having a total of N vertices altogether, it can be determined in $O(n^2 N^6)$ time whether there is*

[5]We are assuming here *general position* of the parts in the arrangement. Certain technical modifications to the algorithm will be necessary to handle "degenerate" assemblies.

[6]For practical use, one may gain a lot from computing only the boundary rather than using all the segments. This may affect the constant factor in our analysis.

a subassembly that can be removed along a path consisting of a single finite translation followed by a translation to infinity. The algorithm outputs both the labels of the parts in the removable subassembly and the specifications of the path.

The bound above assumes a worst case combinatorial bound $O(m^3)$ for an arrangement of m surfaces in 3-space. It might be that the underlying arrangement does not achieve this bound. To exploit the potential "sparseness" of the arrangement one can employ an output-sensitive algorithm for computing the arrangement, like the one devised by de Berg *et al.*[7] [4].

It may appear that checking each blocking graph independently for strong connectedness is rather wasteful, since each graph differs from the previous one by at most a single constraint. In related work, Khanna, Motwani, and Wilson have shown the following: given a directed graph with n nodes, and a "long" (compared to n) sequence of edge insertions and deletions to that graph, the strong connectedness of all resulting graphs can be determined in amortized time $O(n^{1.38})$ per graph [13]. The method groups the sequence of graphs into phases and pre-processes the common sub-graph for each phase. The method applies directly to our problem, resulting in the following theorem.

Theorem 4.3 *Given a planar assembly consisting of n disjoint simple polygons, having a total of N vertices altogether, it can be determined in $O(n^{1.38}N^6)$ time whether there is a subassembly that can be removed along a path consisting of a single finite translation followed by a translation to infinity. The algorithm outputs both the labels of the parts in the removable subassembly and the specifications of the path.*

5 Partitioning with a Bounded Number of Translations

We now consider partitioning an assembly along a path consisting of a small number k of translations $m_1, m_2, \ldots, m_k$. There are $2k - 1$ degrees of freedom in specifying the path t. Two parameters (x_i, y_i) specify the endpoint of each of the first $k - 1$ moves, and one parameter ϕ specifies the direction of the last (infinite) move. Hence we will examine the k-translation problem in a $2k - 1$-dimensional space with coordinates $(x^{(1)}, y^{(1)}, \ldots, x^{(k-1)}, y^{(k-1)}, \phi)$.

We distinguish three different types of moves along the path t: the *first* move m_1 starting at $(0,0)$, the *last* (infinite) move m_k, and the *intermediate* moves $m_2, \ldots, m_{k-1}$.

[7]One does not need all the machinery of [4] in this case. All that is needed here is the so-called "first step decomposition" of the arrangement. Although [4] deals with triangles in space, the algorithm for computing the required decomposition can be easily extended to the arrangements studied in this paper.

As we will show below, the analysis of the first and the last moves is similar to the analysis in Subsections 4.1 and 4.2 respectively. Thus we start by analyzing the contribution of constraint (hyper)surfaces by an intermediate move m_i.

Consider the four-dimensional space with coordinates $(x^{(i-1)}, y^{(i-1)}, x^{(i)}, y^{(i)})$, where every point represents an intermediate translational move in the plane from $(x^{(i-1)}, y^{(i-1)})$ to $(x^{(i)}, y^{(i)})$. We wish to subdivide the space $(x^{(i-1)}, y^{(i-1)}, x^{(i)}, y^{(i)})$ into non-critical cells, such that the blocking graph corresponding to the moves represented by all the points inside the cell is the same. It can be easily shown that every feature of a C-obstacle contributes at most a fixed (constant) number of critical three dimensional hypersurfaces, all algebraic of bounded degree.

The analysis of constraint curves induced by the first motion in the 2D space with coordinates $(x^{(1)}, y^{(1)})$ is the same as the analysis in Subsection 4.1. Similarly, the analysis of constraint surfaces induced by the last move in the 3D space with coordinates $(x^{(k-1)}, y^{(k-1)}, \phi)$ is the same as the analysis in Subsection 4.2. Our goal now is to put all the constraints together in the $2k-1$-dimensional space with coordinates $(x^{(1)}, y^{(1)}, \ldots, x^{(k-1)}, y^{(k-1)}, \phi)$. We use the guidelines that we have used in Subsection 4.3. For example, consider a 3D constraint hypersurface defined above in the space $(x^{(i-1)}, y^{(i-1)}, x^{(i)}, y^{(i)})$. This 4D space can be viewed as a 4D cross-section of the underlying $2k-1$ dimensional space, where all but the four coordinates $(x^{(i-1)}, y^{(i-1)}, x^{(i)}, y^{(i)})$ are fixed. Moreover, the same 3D constraint prevails for every other 4D cross-section as above of the $2k-1$-dimensional space. Hence, we extend every critical hypersurface defined in the k subspaces $(x^{(1)}, y^{(1)}), (x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}), \ldots, (x^{(k-1)}, y^{(k-1)}, \phi)$ into a $2k-2$-dimensional surface in the space $(x^{(1)}, y^{(1)}, \ldots, x^{(k-1)}, y^{(k-1)}, \phi)$, in the obvious manner.

Let μ denote the overall number of features (vertices and edges) in all the $O(n^2)$ original C-obstacles defined earlier. Recall that, by the analysis in Subsection 4.4, $\mu = O(N^2)$. Note that each of the k moves contributes $O(\mu)$ critical $2k-2$ dimensional surfaces. Therefore, the number of non-critical cells in the subdivision of the space $(x^{(1)}, y^{(1)}, \ldots, x^{(k-1)}, y^{(k-1)}, \phi)$, is $O((k\mu)^{2k-1})$.

To compute a subassembly that is separable by k translations we employ an algorithm to produce all the cells in the above $2k-1$-dimensional arrangement, and for each cell we check the corresponding blocking graph for strong connectedness.

Theorem 5.1 *Given a planar assembly consisting of n disjoint simple polygons, having a total of N vertices altogether, it can be determined in $O((kN^2)^{2k-1}n^2)$ time whether there is a subassembly that can be removed along a path consisting of at most k translations where* *the last translation extends to infinity. The algorithm outputs both the labels of the parts in the removable subassembly and the specifications of the path.*

Proof. The actual computation of the arrangement can be carried out in time roughly $O((kN^2)^{2k-1})$ (see Subsection 3.1) and hence it is negligible in comparison with the strong connectivity check for all the cells. The $2k-1$ dimensional arrangement is determined by $O(kN^2)$ hyper surfaces, and therefore the maximum number of cells in it is $O((kN^2)^{2k-1})$. In each cell we spend time $O(n^2)$ to check for strong connectivity, and the bound $O((kN^2)^{2k-1}n^2)$ follows. □

6 Discussion

The above result builds on two existing concepts: the NDBG [22] and the "interference diagram" [21]. Previously, NDBGs were studied only for simple types of motions, and thus yielded either a subset or superset of the possible assembly operations, while it was not clear how to use the interference diagram efficiently in order to solve the partitioning problem. Notably, the algorithm above encodes exactly the type of motions executed by 2-axis linear actuators and some other common types of assembly automation. Hence this paper is a step in showing the full generality of the NDBG approach.

The major open problem that this paper raises is to improve the running time of the algorithms presented in it. Some possible directions for improvement are suggested in Subsection 4.4. A related question, which applies to other instances of the NDBG framework as well, is the following: We compute a collection of $n(n-1)$ C-obstacles. However, this collection of C-obstacles is induced by only n parts. Can we exploit this fact to improve the running time of our algorithm, or of other algorithms that deal with the NDBG? We are currently investigating this question.

Of practical importance is the extension of our algorithm to polyhedral assemblies and motions in three dimensions. In 3D, the first finite translation (x, y, z) would be followed by an infinite translation in a direction given by two angles (ϕ, ψ). The C-obstacle $P_i \ominus P_j$ is a polyhedron with $O(n_i n_j)$ planes supporting its faces, and central shadows and (ϕ, ψ)-shadows can be defined as in the 2D case. Thus the 5D space of motions will be divided into cells by a number of surfaces generated by these C-obstacles as ϕ and ψ vary, with a blocking graph for each cell, and so on. While polynomial, the running time of such an algorithm may be rather high in the worst case.

A similar methodology might be applied to other simple motions performed by highspeed assembly machines. For instance, many such machines only perform horizontal and vertical motions. In this case (in 3D, with the orientation of the assembly given), a horizontal staging motion at angle ϕ is followed by a vertical insertion of length z, resulting in a 2D space of motions.

Another common assembly mechanism rotates a part from its feeding position to an intermediate point, then inserts the part vertically. The important parameters defining such a motion are the length z of the vertical insertion and the center (x, y) of planar rotation. One additional point must be addressed, however: only rotation centers (x, y) that completely remove the subassembly are valid, and the valid rotation centers might vary depending on the subassembly. This deserves further investigation.

References

[1] P. Agarwal, M. de Berg, D. Halperin, and M. Sharir. Efficient generation of k-directional assembly sequences. In *Proc. 7th ACM-SIAM Symp. on Discrete Algorithms*, pages 122–131, 1996.

[2] E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles, with applications to planning assemblies. In *Proc. 5th ACM Symp. on Computational Geometry*, pages 334–343, 1989.

[3] S. Basu, R. Pollack, and M.-F. Roy. A new algorithm to find a point in every cell defined by a family of polynomials. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition.* Springer-Verlag. To appear.

[4] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. *Discrete Comput. Geom.*, 15:35–61, 1996.

[5] E. Dunn, Sales Engineer, Bodine Assembly and Test Systems. Personal communication, April 1994.

[6] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[7] L. Guibas and M. Sharir. Combinatorics and algorithms of arrangements. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, pages 9–36. Springer, 1993.

[8] L. J. Guibas, D. Halperin, H. Hirukawa, J.-C. Latombe, and R. H. Wilson. A simple and efficient procedure for polyhedral assembly partitioning under infinitesimal motions. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2553–2560, 1995.

[9] D. Halperin. Arrangements. In E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry.* CRC Press. to appear.

[10] D. Halperin. *Algorithmic motion planning via arrangements of curves and of surfaces.* PhD thesis, Dept. of Computer Science, Tel-Aviv Univ., July 1992.

[11] A. Kaul, M. A. O'Connor, and V. Srinivasan. Computing Minkowski sums of regular polygons. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 74–77, 1991.

[12] L. Kavraki and M. N. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*, 55(3):159–165, 1995.

[13] S. Khanna, R. Motwani, and R. H. Wilson. On certificates and lookahead in dynamic graph problems. In *Proc. 7th ACM-SIAM Symp. on Discrete Algorithms*, pages 222–231, 1996.

[14] J.-C. Latombe. *Robot Motion Planning.* Kluwer, 1991.

[15] B. K. Natarajan. On planning assemblies. In *Proc. 4th ACM Symp. on Computational Geometry*, pages 299–308, 1988.

[16] R. Pollack and M. Roy. On the number of cells defined by a set of polynomials. *C.R. Acad. Sci. Paris*, 316(I):573–577, 1993.

[17] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete and Computational Geometry*, 3:123–136, 1988.

[18] A. Schweikard and R. H. Wilson. Assembly sequences for polyhedra. *Algorithmica*, 13(6):539–552, 1995.

[19] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.

[20] G. T. Toussaint. Movable separability of sets. In G. T. Toussaint, editor, *Computational Geometry.* Elsevier, 1985.

[21] R. H. Wilson, L. Kavraki, T. Lozano-Perez, and J.-C. Latombe. Two-handed assembly sequencing. *Intl. J. of Robotics Research*, 14(4):335–350, 1995.

[22] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1994.

[23] J. D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly.* PhD thesis, Univ. of Michigan, 1988.