

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

Abstract

We consider the following problem that arises in assembly planning: given an assembly, identify a sub-assembly 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 sub-assembly 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. We discuss extending the algorithm to 3D, other types of motions typical in non-robotic automated assembly, and motions consisting of more than two translations.

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 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 [3]. Reducing this lead time would allow faster time-to-market with lower cost for many high-volume products.

*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 IRL-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).

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*, 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. 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 then inserted into the larger), while the assembly in Figure 1(b) cannot. In the rest of this paper, we only consider monotone two-handed assembly sequences. We also restrict our attention to assemblies of rigid parts.

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.

Past work has shown that when a disassembly motion may consist of any number of translations, the

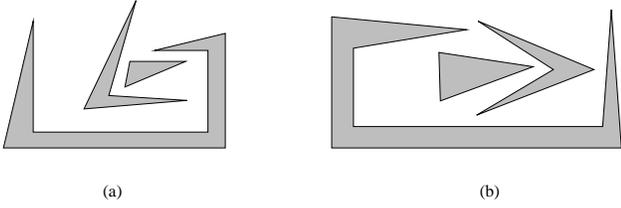


Figure 1: Assemblies requiring non-straight-line motions for disassembly. (a) is a monotone two-handed assembly while (b) is not.

partitioning problem (and thus sequencing itself) is NP-complete [7]. On the other hand, polynomial-time partitioning is possible when the motions are limited to single translations to infinity: Arkin *et al.* [1] present an algorithm for planar assemblies of polygons, and [12, 13] consider assemblies of polyhedra.

This paper generalizes the *non-directional blocking graph* (or *NDBG*) of [13] to motions consisting of multiple 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 $O(n^2N^6)$ time, where n is the number of parts in the assembly and N is the total number of vertices of the polygons. We have extended this algorithm to partitioning assemblies with a small number $k > 2$ of translations. For lack of space, we do not present the latter algorithm here; see [6] for more detail.

The rest of the paper is organized as follows. In Section 2 we describe the NDBG and give other background. Section 3 gives the algorithm to partition an assembly of polygons with two translations, and Section 4 analyzes its computational complexity. Section 5 briefly considers paths of more than two translations. 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 Background

Consider an assembly A of non-overlapping objects. The objects are the *parts* of the assembly, and any subset of them is a *subassembly*. In general, we wish to identify a proper subassembly $S \subset A$ that can be completely separated from $A \setminus S$ (the rest of the assembly) by a collision-free rigid motion along a continuous

path t .

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 . The set of constraints on membership in S can be represented with a *blocking graph* [13]. 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.¹ When $G_A(t)$ is not strongly connected, one of its strong components is a subassembly that can follow t without collision.

Figure 2 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.² 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.

For translational paths in two dimensions, a path is a continuous mapping $t : [0, \infty) \rightarrow (x, y)$, where $t(0) = (0, 0)$ and each point (x, y) on the path is considered a relative offset from a part's initial position in the assembly. The set of offsets of a part Q that collide with part P is given by the *configuration-obstacle* (or *C-obstacle*) $P \ominus Q$, i.e., the Minkowski sum of P and $-Q$ [9]. Hence part P blocks Q along a path t exactly when t intersects $P \ominus Q$. To find the collisions between all pairs of parts for a single motion, all pairwise C-obstacles $P \ominus Q$ can be computed. When the

¹A *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.

²The techniques of this paper apply equally well to more “normal” assemblies, with parts in contact. To simplify the presentation, however, we use examples such as those in figure 2.

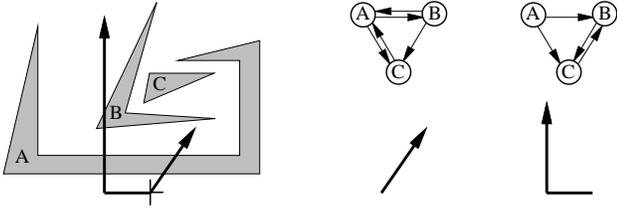


Figure 2: An assembly and its blocking graphs for two translational motions

pairwise C-obstacles are superimposed, they partition the plane into regions, within which the set of pairs of parts colliding is fixed; this is called the *interference diagram* for the assembly. A path t causes the subassembly to move through a sequence of regions, collecting constraints on the blocking graph of t . For further discussion see [10].

3 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.

To solve the problem, we partition the (x, y, ϕ) -space of possible motions into cells such that the blocking graph $G_A(t)$ is fixed for all motions t in a cell. Then by checking the blocking graph for each cell for strong connectedness, we can determine whether a subassembly can be removed along motions in that cell. If all blocking graphs are strongly connected, no subassembly can be removed by a two-translation path.

The next subsection derives the constraints arising from the first translation (x, y) , and subsection 3.2 derives the constraints arising from a second translation given by ϕ . Then in subsection 3.3 we combine these sets of constraints to obtain the final three-dimensional arrangement, i.e., the subdivision of the (x, y, ϕ) -space.

3.1 The First Translation

Consider first 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 space of motions (x, y) at which the blocking graph for the first translation changes.

As mentioned above, for any two parts P and Q , the set of placements of Q for which Q intersects P is the polygonal C-obstacle $C = P \ominus Q$. Part P blocks Q for translations (x, y) that intersect C . Let the *central shadow* of a region R , written $S_C(R)$, be the set of points (x, y) such that the line segment from $(0, 0)$ to (x, y) intersects R . Each edge on the boundary of a central shadow $S_C(R)$ is either a portion of an edge of R or a ray extended from a vertex of R . The edges of the central shadow $S_C(P \ominus Q)$ are critical curves for the first translation: the arc $Q \rightarrow P$ is present in blocking graphs for exactly those motions $(x, y) \in \text{int}(S_C(P \ominus Q))$ ending inside the shadow.

Figure 3b shows the C-obstacle and resulting central shadow of the two polygons in figure 3a.

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, the blocking graph $G_A((x, y))$ is fixed. We denote the set of line segments making up this arrangement by S_1 .

Figure 4 shows the C-obstacles (dashed lines) and the boundaries of the corresponding central shadows (solid) for each pair of parts of the assembly in Figure 2. 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.

3.2 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 blocking graph for 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. Let the ϕ -shadow of a region R , written $S_\phi(R)$, be the set of points (x, y) such that the ray (x, y, ϕ) intersects R . As

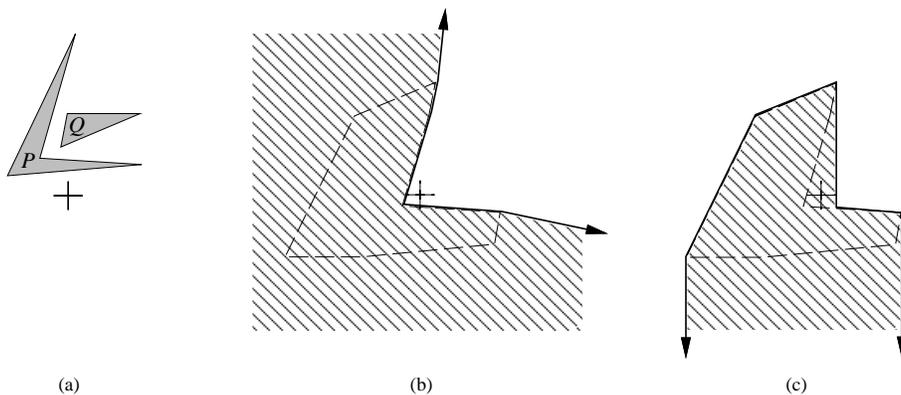


Figure 3: (a) Two polygons P and Q , (b) the boundary of $P \ominus Q$ (dashed) and the central shadow $S_C(P \ominus Q)$, and (c) the ϕ -shadow $S_{90^\circ}(P \ominus Q)$

with central shadows, each edge on the boundary of a ϕ -shadow $S_\phi(R)$ is either a portion of an edge of R or a ray extended from a vertex of R . Then the blocking graph for a ray (x, y, ϕ_0) contains the edge $Q \rightarrow P$ exactly when $(x, y) \in \text{int}(S_\phi(P \ominus Q))$. 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.

Figure 3c shows the C-obstacle and resulting ϕ -shadow of the two polygons in figure 3a for the direction of motion $\phi_0 = 90^\circ$.

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 $\phi \in [0, 2\pi)$. We denote this set of critical surfaces by \mathcal{S}_2 . The surfaces in \mathcal{S}_2 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 boundaries of the ϕ -shadows for the assembly of Figure 2, and the resulting arrangement, for the upward translation $\phi = 90^\circ$.

3.3 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 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. Since the first translation is unaffected by the value of ϕ , we extend each segment in the first arrangement \mathcal{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 \mathcal{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.

Finally, 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 $\mathcal{S}_1 \cup \mathcal{S}_2$. For each cell produced by the algorithm, we compute the blocking graph $G_A(t)$ corresponding to a representative path t for that cell. We check each blocking graph for strong connectedness: if $G_A(t)$ is not strongly connected, then we output one of its strong components and the motion t as a solution. If all blocking graphs are strongly connected, then the assembly cannot be partitioned with two translations.

The blocking graphs for all cells in the arrangement can be computed incrementally in the following way. We begin by choosing a point $t_0 = (x_0, y_0, \phi_0)$ in some cell c_0 of the arrangement; the blocking graph $G_A(t_0)$ can be easily computed by checking for inclusion of (x_0, y_0) in the central shadows and ϕ_0 -shadows of the pairwise C-obstacles. 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.

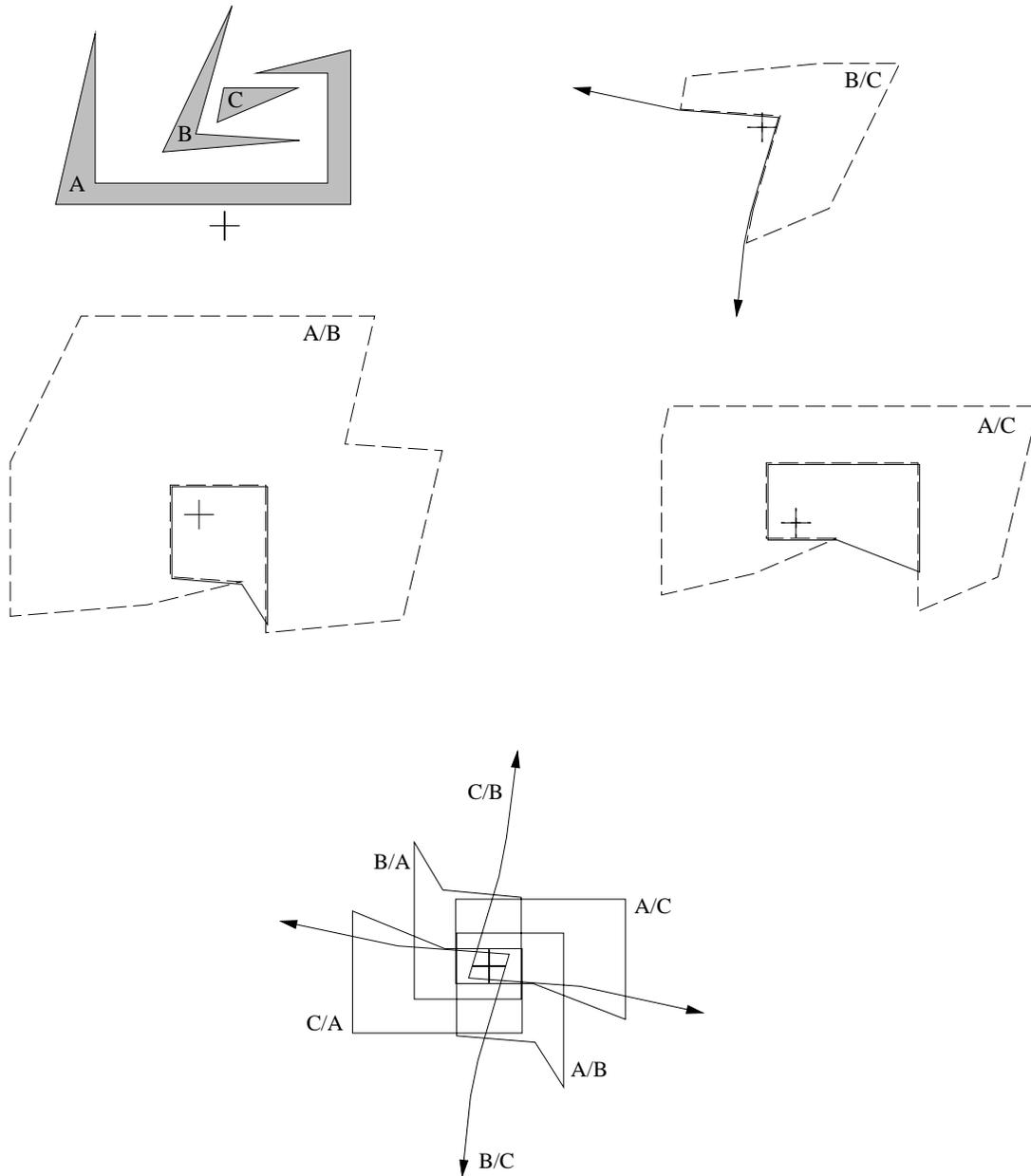


Figure 4: An assembly, C-obstacles for all pairs of parts (dashed lines), boundaries of the corresponding central shadows (solid), and the arrangement for the first translation

4 Computational Complexity

What is the time complexity of this algorithm? We present here a summary of the analysis; see [6] for a more detailed discussion. To simplify the presentation, we first assume that each part has at most some fixed number of vertices. We then give a refined analysis below.

4.1 Initial Analysis

Let the assembly A have n polygonal parts, each of constant maximum complexity. To determine the complexity (number of cells) in the arrangement of surfaces in (x, y, ϕ) -space, consider first the number of surfaces in the set \mathcal{S}_1 (generated by the first translation). Since each part has a constant number of vertices, the C-

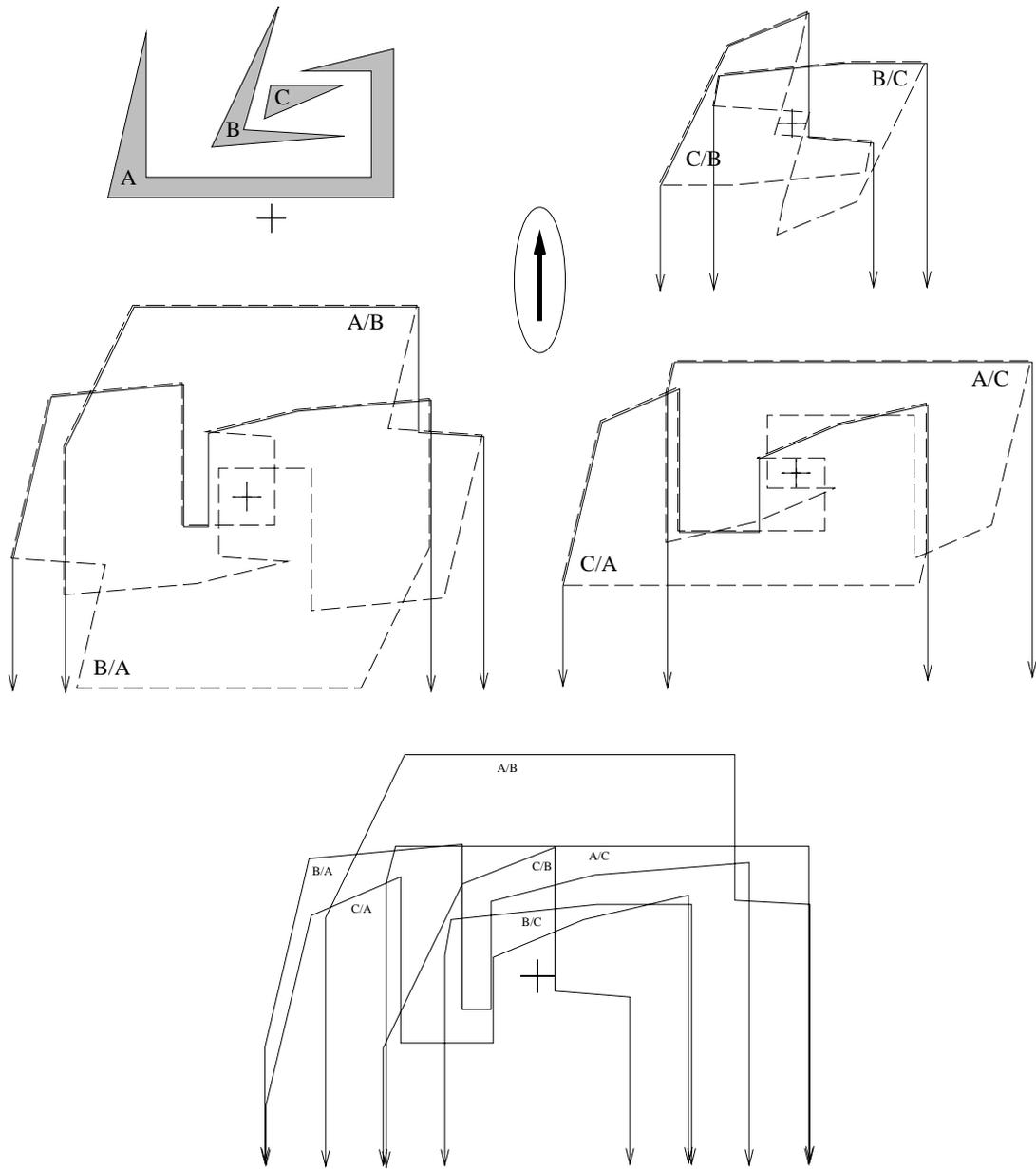


Figure 5: An assembly, ϕ -shadows for pairs of parts when $\phi = 90^\circ$, and a 2D slice of the arrangement for the second translation at $\phi = 90^\circ$

obstacle for two parts is also of complexity $O(1)$, and the same holds for the boundary of the central shadow of such a polygon. Each pair of parts creates one C -obstacle, so the number of surfaces in the set \mathcal{S}_1 is $O(n^2)$.

Now consider the surfaces in the set \mathcal{S}_2 generated by the second translation. To simplify the analysis, we will consider a superset of this collection. Namely,

for every C -obstacle C , we will consider the surfaces traced by the set of all the segments underlying the C -obstacle C as ϕ varies, together with the ruled surfaces traced by rays in the $\phi + 180^\circ$ direction extended from vertices of C . This set is clearly a superset of the surfaces in \mathcal{S}_2 .

Again, 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.

The surfaces (or more precisely, surface patches) in \mathcal{S}_1 and \mathcal{S}_2 are clearly algebraic of bounded degree. It is well known that the maximum number of cells in a 3D arrangement induced by m such surfaces is $O(m^3)$ (see, e.g., [4, 5, 11]). Since there are $O(n^2)$ surfaces in each of \mathcal{S}_1 and \mathcal{S}_2 , the maximum number of cells in the subdivision of (x, y, ϕ) -space is $O(n^6)$.

The algorithm requires that we visit all cells in the arrangement, at each step moving from a cell to one of its neighbors. This can be done in time very close to linear in the number of cells in the arrangement, i.e. roughly $O(n^6)$ time, with a simple spatial sweep algorithm [2] (the algorithm in [2] incurs an extra polylogarithmic factor, which is negligible here due to other steps of our algorithm).

The blocking graph for the initial cell in the traversal can be computed by comparing the point (x_0, y_0, ϕ_0) to each shadow, that is in time $O(n^2)$. Then as each cell boundary is traversed to generate the next blocking graph, at most a constant number of constraints is added or removed at each step,³ so 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 . Since $O(n^6)$ blocking graphs must be checked, this last step dominates the running time of the algorithm. We now have 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.*

4.2 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 . While the

³We are assuming here *general position* of the parts in the assembly. Certain technical modifications to the algorithm will be necessary to handle “degenerate” assemblies, but will not increase its asymptotic running time.

Minkowski sum of two polygons $P_i \ominus P_j$ may have complexity $O(n_i^2 n_j^2)$, the collection of segments underlying all these edges may have at most $O(n_i n_j)$ segments—one for every vertex of one part and edge of the other⁴. 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 the number of cells in the arrangement of (x, y, ϕ) -space is $O(N^6)$, and the time to compute it is roughly the same. The blocking graph for each cell in the arrangement must be checked for strong connectedness, giving us the following theorem.

Theorem 4.2 *Given a planar assembly consisting of n disjoint simple polygons, having a total of N vertices, it can be determined in $O(n^2 N^6)$ 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 removable subassembly and the path.*

Finally, 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 [8]. They group the sequence of graphs into phases and pre-process the common sub-graph for each phase. The method applies directly to checking the long sequence of blocking graphs in the above algorithm, thus reducing the running time to $O(n^{1.38} N^6)$.

5 Multiple Translations

In [6] we consider partitioning an assembly along a path consisting of a small number k of translations m_1, m_2, \dots, 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 examine the k -translation problem in a $(2k - 1)$ -dimensional space with coordinates $(x_1, y_1, \dots, x_{k-1}, y_{k-1}, \phi)$. The resulting partitioning algorithm is polynomial in the complexity of the assembly, but exponential in the number of translations k allowed. We refer the reader to [6] for details.

⁴For 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.

6 Discussion

The above result builds on two existing concepts: the concept of the NDBG [13] and the “interference diagram” [10]. 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 [6]. 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 high-speed 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] E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles, with applications to planning assemblies. In *Proc. of the 5th ACM Symp. on Computational Geometry*, pages 334–343, 1989.
- [2] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space (the full version). Technical Report RUU-CS-94-29, Utrecht University, 1994. Also in *Proc. of the 10th ACM Symp. on Computational Geometry*, 1994, pp. 1–10.
- [3] E. Dunn, Sales Engineer, Bodine Assembly and Test Systems. Personal communication, April 1994.
- [4] 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.
- [5] D. Halperin. *Algorithmic motion planning via arrangements of curves and of surfaces*. PhD thesis, Dept. of Computer Science, Tel-Aviv Univ., July 1992.
- [6] D. Halperin and R. H. Wilson. Assembly partitioning with a constant number of translations. Technical Report SAND94-1819, Sandia National Labs, 1994.
- [7] L. Kavraki, J.-C. Latombe, and R. H. Wilson. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229–235, 1993.
- [8] S. Khanna, R. Motwani, and R. H. Wilson. Graph Certificates, Lookahead in Dynamic Graph Problems, and Assembly Planning in Robotics. In preparation, 1994.
- [9] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [10] T. Lozano-Pérez and R. H. Wilson. Assembly sequencing for arbitrary motions. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, volume 2, pages 527–532, 1993.
- [11] R. Pollack and M. Roy. On the number of cells defined by a set of polynomials. *C.R. Acad. Sci. Paris*, (t. 316, Série I):573–577, 1993.
- [12] A. Schweikard and R. H. Wilson. Assembly Sequences for Polyhedra. To appear in *Algorithmica*.
- [13] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2), 1994.