Assembling Polyhedra with Single Translations

by

Randall Wilson and Achim Schweikard

Department of Computer Science
Stanford University
Stanford, California 94305
Assembling Polyhedra with Single Translations

Randall H. Wilson, Achim Schweikard

Stanford University
Stanford, CA 94305

Office of Naval Research
Resident Representative
Stanford University
Rm.202 McCullough Bldg.
Stanford, CA 94305-4055

available to the public

The problem of partitioning an assembly of polyhedral objects into two subassemblies that can be separated arises in assembly planning. We describe an algorithm to compute the set of all translations separating two polyhedra with \( n \) vertices in \( O(n^4) \) steps and show that this is optimal. Given an assembly of \( k \) polyhedra with a total of \( n \) vertices, an extension of this algorithm identifies a valid translation and removable subassembly in \( O(k^2n^4) \) steps if one exists. Based on the second algorithm a polynomial time method for finding a complete assembly sequence consisting of single translations is derived. An implementation incorporates several changes to achieve better average-case performance; experimental results obtained for composite objects consisting of isothetic polyhedra are described.
Assembling Polyhedra with Single Translations

Randall H. Wilson, Achim Schweikard
Robotics Laboratory, Department of Computer Science
Stanford University
Stanford, CA 94305-4110

October 17, 1991

Abstract

The problem of partitioning an assembly of polyhedral objects into two subassemblies that can be separated arises in assembly planning. We describe an algorithm to compute the set of all translations separating two polyhedra with n vertices in $O(n^4)$ steps and show that this is optimal. Given an assembly of $k$ polyhedra with a total of $n$ vertices, an extension of this algorithm identifies a valid translation and removable subassembly in $O(k^2 n^4)$ steps if one exists. Based on the second algorithm a polynomial time method for finding a complete assembly sequence consisting of single translations is derived. An implementation incorporates several changes to achieve better average-case performance; experimental results obtained for composite objects consisting of isothetic polyhedra are described.

Introduction

The problem of finding sequences of motions for the assembly of a given object consisting of polyhedral parts arises in assembly planning. This problem can be regarded as a motion planning problem with multiple moving objects. In this general form, the problem involves many degrees of freedom. Since known methods for motion planning allowing general motions are exponential in the number of degrees of freedom, it is useful to restrict the type of motion considered. Here we will impose the following restrictions:

- Each step in an assembly sequence concerns two subassemblies. Two subassemblies which have been joined in a previous step are not moved relative to each other in subsequent steps.
- At each step in an assembly sequence, a single translation moves the first subassembly to its final position relative to the second subassembly.
These restrictions embody practical constraints often imposed on assembly sequences by manufacturing processes; complicated assembly motions and operations joining more than two subassemblies make assembly more difficult and raise manufacturing costs.

For assemblies of rigid parts, an assembly plan can be obtained by reversing a valid disassembly plan. In this context, we will address the following two subproblems:

1. Given two polyhedra, compute the set of all single translations separating these polyhedra.

2. Given an assembly $A$ of several polyhedra, decide whether there is a direction $d$ and a subassembly $S \subseteq A$ such that a translation along $d$ separates $S$ from the remaining parts $A \setminus S$.

Given a solution to the second problem, we will show that complete assembly sequences for polyhedral parts can easily be computed.

Two examples for the second problem are shown in figures 1 and 2. None of the polyhedra in figure 1 can be separated from the remaining parts by a translation involving a single object, but there are subassemblies which can be moved simultaneously. In figure 2 any subset of the cubes $P_1, \ldots, P_4$ can be removed by a single simultaneous translation from the remaining objects. This second example shows that the number of removable subassemblies is exponential in general. Hence, it is not practical to compute all removable subassemblies explicitly.

However, we show that it can be determined in polynomial time whether a removable subassembly exists. Specifically, we describe an optimal algorithm for solving the first problem above. The method is then extended to derive a polynomial time algorithm to solve the second problem. Using this procedure,
we derive a polynomial time method for finding complete assembly sequences. Finally, an implementation of the above algorithms and the results of various assembly planning experiments are described.

1 Related Work

A survey of earlier methods for separating sets in two and three dimensions is given in [14]. In [11] lower bounds on the number of simultaneous translations necessary for separating objects are derived. Dawson [3] shows that two or more star-shaped objects can always be separated by translating the objects in different directions simultaneously. In addition, it is shown in [3] that for some assemblies of convex polyhedra, no individual parts are removable by a single translation.

Homem de Mello and Sanderson [6] give a method to calculate the polyhedral convex cone containing the infinitesimal translations allowed by a set of planar contacts in space. A polynomial-time algorithm to identify subassemblies that are connected and can be translated a small distance relative to the rest of the assembly is described in [16]. Both of these methods consider only contact, and thus cannot find collision-free extended translations.

Krishnan and Sanderson [7] address problem 1 by mapping the set of all unit translations onto a two-dimensional grid, and marking grid elements that correspond to collisions between two polyhedra. Any unmarked elements then represent valid removal translations. However, this method is only accurate to the size of the grid, and cannot be used to find translations involving contacts between the two parts.

Pollack, Sharir and Sifrony [12] consider sequences of translations to separate
polygons. The algorithm [12] is limited to planar assemblies of two parts, but is able to find separating motions consisting of several distinct translations.

Toussaint [15] describes an algorithm for separating two simple polygons by a single translation; this is the planar case of problem 1 above. Similarly, Arkin, Connelly and Mitchell [1] address the planar version of problem 2. They use the concept of monotone paths among polygonal obstacles to identify a removable subassembly of simple polygons in the plane. The methods in [1] do not extend directly to the three-dimensional case. However, Mitchell has independently shown that directions for partitioning an assembly can be found in polynomial time [10]. In this paper we give an algorithm to find the set of translations separating two polyhedra in \( O(n^4) \) time and show that this is optimal. A method for finding complete assembly sequences based on this algorithm is analyzed theoretically as well as empirically and improvements for applications are described. The analytic bounds derived consider the number of parts, the number of vertices in the representation of the parts, and the size of the coordinates in the input description.

2 Separating Two Polyhedra

In this section a method for finding the set of translations separating two polyhedra will be derived. Let \( P \) and \( Q \) be closed and disjoint polyhedra in given spatial placement, and let \( P \) and \( Q \) be represented as unions of at most \( n \) tetrahedra, i.e. \( P = \bigcup_{i=1}^{n} T_i \) and \( Q = \bigcup_{j=1}^{m} U_j \) where \( r, s \leq n \). A translation separating \( P \) from \( Q \) is a vector \( d \) such that \( P \) can be translated to infinity in direction \( d \) without intersecting \( Q \), i.e. \( p + td \) is not in \( Q \) for each point \( p \) in \( P \) and each \( t \) in \([0, \infty)\).

The set of all translations of \( P \) can be represented by the points on the unit sphere \( S^2 \) in three-dimensional space. For each pair of tetrahedra \( T_i, U_j \), the configuration obstacle \( C(T_i, U_j) \) is the set of placements of \( T_i \) such that \( T_i \) intersects \( U_j \) [9]. The set of translations along which \( T_i \) collides with \( U_j \) is the projection of \( C(T_i, U_j) \) on the unit sphere; let \( R_{ij} \) denote this region.

The regions \( R_{ij} \) are bounded by segments of great circles on \( S^2 \). The set of great circles for all \( R_{ij} \) determines open regions on \( S^2 \) called faces. Each face is a maximal connected component on the sphere not intersecting any great circle in this set. The faces are regular in the following sense: the pairs of tetrahedra \( T_i, U_j \) from \( P \) and \( Q \) that collide in direction \( d \) are constant for all translations \( d \) in a face \( f \). Let \( p(f) \) be the number of pairs of tetrahedra that collide along translations in face \( f \). If the segment \( e \) of a great circle lies between two faces \( f \) and \( f' \), we have the following crossing rules for \( p(f) \) and \( p(f') \):

- If \( e \) belongs to the boundary of a region \( R_{ij} \) and \( f' \) is on the interior side of \( e \), then \( p(f') = p(f) + 1 \).
• If \( f' \) is on the exterior side of region boundary \( e \), then \( p(f') = p(f) - 1 \).

• If \( e \) is not on the boundary of any region \( R_{ij} \), then \( p(f') = p(f) \).

In some cases, several edges may coincide. If two faces \( f \) and \( f' \) are separated by an edge \( e \) bounding several regions, then \( p(f') = p(f) - g + h \), where \( g \) is the number of regions on the same side of \( e \) as \( f \) and \( h \) is the number of regions on the same side of \( e \) as \( f' \). The set of translations separating \( P \) from \( Q \) is the union of all faces \( f \) for which \( p(f) = 0 \).

In the algorithm below, translations are represented as points on two parallel planes instead of points on the unit sphere. Configuration obstacles \( C(T_i, U_j) \) are projected to the planes \( z = 1 \) and \( z = -1 \) using a central projection from the origin. The regions \( R_{ij} \) are planar regions bounded by line segments and rays. The supporting lines of these segments and rays define an arrangement in each plane, represented by a graph. The nodes in the graph represent faces, edges, and vertices of the arrangement, and links connect adjacent elements. Edges on region boundaries are oriented with respect to the interior of the corresponding region, while edges obtained by extended supporting lines are marked as such.

The algorithm to find all faces representing valid translations proceeds as follows. For each plane \( z = 1 \) and \( z = -1 \),

1. For each pair of tetrahedra \( T_i, U_j \), compute the projection \( R_{ij} \) of \( C(T_i, U_j) \) on the plane.

2. Calculate the arrangement of lines determined by the boundaries of the regions \( R_{ij} \), orienting the edges as described above.

3. For an arbitrarily selected face \( f_0 \), compute the number \( p(f_0) \) of regions \( R_{ij} \) containing \( f_0 \).

4. Perform a depth-first traversal of all the faces in the arrangement by stepping from \( f_0 \) to neighboring faces. To step from a face \( f \) to a neighboring face \( f' \), calculate \( p(f') \) from \( p(f) \) using the crossing rules above. After visiting a face, it is marked and not visited again. For each face \( f \) where \( p(f) = 0 \), output the face \( f \) and continue.

Since \( P \) and \( Q \) consist of at most \( n \) tetrahedra each, there are at most \( n^2 \) regions \( R_{ij} \), each with a constant number of edges. Therefore step 1 requires \( O(n^2) \) operations. An arrangement of \( m \) lines in the plane can be computed in \( O(m^2) \) time \([2, 4]\) and has \( O(m^2) \) cells. Here \( m = n^2 \), so the number of cells and the computing time for step 2 are \( O(n^4) \). Each region \( R_{ij} \) has a constant number of edges, so testing the initial face \( f_0 \) for inclusion in all regions requires \( O(n^2) \) operations. Finally, the depth-first search steps over each edge at most twice, each step taking constant time. Since the number of arrangement edges is \( O(n^4) \), the computing time for step 4 is \( O(n^4) \).
The optimality of this algorithm directly follows from an example given by Pollack, Sharir and Sifrony [12]. The example in [12] concerns two polygons $P$ and $Q$ with $r$ and $s$ edges respectively; the number of connected components in the complement of the configuration obstacle corresponding to $P$ and $Q$ is proportional to $r^2s^2$ (figure 3). In our case the polygons $P$ and $Q$ are regarded as polyhedra of zero volume, and $r, s = n$; the following holds equally if $P$ and $Q$ are polyhedra with sufficiently small thickness. We place $P$ in a plane $p$ and $Q$ in a plane parallel to $p$, but distinct from $p$. Then the plane containing the configuration obstacle of $P$ with respect to $Q$ does not contain the origin, so the projection of the configuration obstacle of $P$ with respect to $Q$ on the sphere $S^2$ partitions $S^2$ into $\Omega(n^4)$ connected components. Therefore the set of translations separating $P$ from $Q$ consists of $\Omega(n^4)$ connected components.

Instead of decomposing polyhedra into tetrahedra in the above method, we can decompose the faces of the polyhedra into triangles, and find the configuration obstacles for pairs of triangles. In fact, such triangulations are often computed in geometric modeling systems. The faces of a part with $n$ vertices can be triangulated in $O(n \log n)$ steps into $O(n)$ triangles [5], so that the total number of steps in the above algorithm remains $O(n^4)$. We now have the following lemma:

**Lemma 1** The set of all translations separating $P$ from $Q$, where $P$ and $Q$ are polyhedra each with $n$ vertices can be found in $O(n^4)$ steps and this is optimal.

To find separating translations in which parts touch each other, open polyhedra can be considered in the above algorithm. In this case the edge and vertex cells of an arrangement may correspond to valid directions. A count $p(c)$
of colliding tetrahedra is associated with every face, edge, or vertex cell in the arrangement. The arrangement can be computed and the cells traversed in $O(n^4)$ steps.

### 3 Partitioning an Assembly

Problem 2 concerns an assembly $A$ of polyhedra $P_1, \ldots, P_k$. The method of the previous section can be extended to find a translation $d$ and a proper subassembly $S$ of $A$ such that $d$ separates $S$ from the remaining parts $A \setminus S$.

As above, each pair of tetrahedra $T_i$ and $U_j$ from different polyhedra define a region $R_{ij}$ of the unit sphere $S^2$. A directed graph $G(f)$ with weighted arcs is associated with each face $f$ in the corresponding arrangement on $S^2$. The nodes of each graph represent the objects $P_1, \ldots, P_k$. The weight of an arc from $P_i$ to $P_j$ in $G(f)$ is the number of pairwise intersections of tetrahedra from $P_i$ and $P_j$ during any translation $d$ in $f$. Arcs with weight zero are removed from $G(f)$.

The graphs $G(f)$ and $G(f')$ for neighboring faces $f$ and $f'$ sharing an edge $e$ are related by the following crossing rules:

- If $e$ is a boundary segment of a projected configuration obstacle from tetrahedra in $P_i$ and $P_j$, and $f'$ is in the interior of the region, then the weight of the arc from $P_i$ to $P_j$ is one greater in $G(f')$ than in $G(f)$.
- If $e$ is a boundary segment and $f'$ is outside of this region, the weight of the arc from $P_i$ to $P_j$ is one less in $G(f')$ than in $G(f)$.
- If $e$ is the extension of a boundary segment, $G(f') = G(f)$.

Similar to the case of two polyhedra, if several edges coincide then $G(f')$ differs from $G(f)$ by the sum of the changes for the coinciding edges.

A proper subset $S$ of $A$ can be removed along a direction $d$ in a face $f$ if and only if there are no arcs in $G(f)$ from nodes in $S$ to nodes in $A \setminus S$. A node $P_i$ is a successor of $P_j$ in the graph $G(f)$ if $i = j$ or there is a path in $G(f)$ from $P_i$ to $P_j$. The predecessors of a node are defined similarly. If the set of successors of every node in $G(f')$ is the entire set of graph nodes, then there is no subassembly that can be removed using directions in face $f$. However, it suffices to compute the sets of successors and predecessors of a single arbitrary node $P_1$:

- If the set of successors and the set of predecessors of $P_1$ are both equal to $A$, then there is no proper subassembly of $A$ that can be removed in a direction in $f$. This follows from the transitivity of the successor relation.
- If the set of successors $S_1$ of $P_1$ is a proper subset of $A$, then $S_1$ is a removable subassembly of $A$. 7
If the set of predecessors \( S_2 \) of \( P_1 \) is a proper subset of \( A \), then by definition no arcs connect nodes in \( A \setminus S_2 \) to nodes in \( S_2 \). Therefore \( A \setminus S_2 \) is a removable subassembly of \( A \).

To find a removable subassembly of \( A \), we again project the configuration obstacles \( C(T_i, U_j) \) onto two planes \( z = 1 \) and \( z = -1 \). However, if a translation \( d \) separates a subassembly \( S \) from \( A \setminus S \), then \(-d\) separates the subassembly \( A \setminus S \) from \( S \). Thus it suffices to search only one planar arrangement. This gives rise to the following algorithm for finding a removable subassembly:

1. Calculate the arrangement of regions \( R_{ij} \) on the plane \( z = 1 \).
2. Compute the graph \( G(f_0) \) for an initial face \( f_0 \) of the arrangement.
3. Perform a depth-first traversal over the arrangement, computing \( G(f) \) for each new face \( f \). If in any graph \( G(f) \) the successors or predecessors of \( P_1 \) are a proper subset \( S \) of \( A \), output \( S \) and a translation \( d \) in \( f \).

The arrangement of projected regions can be calculated in \( O(n^4) \) steps, and the initial graph \( G(f_0) \) can be found in \( O(n^2) \) steps. Finding the set of successors or predecessors of a node in one graph requires \( O(k^2) \) steps. There are \( O(n^4) \) faces in the arrangement, so traversing them all requires \( O(k^2 n^4) \) operations.

We now have the following lemma:

**Lemma 2** Let \( A = \{P_1, \ldots, P_k\} \) be a set of \( k \) polyhedra with a total of \( n \) vertices. It can be decided in \( O(k^2 n^4) \) steps whether there is a proper subassembly of \( A \) that can be translated to infinity without intersecting the remaining parts. An appropriate subassembly and direction can be computed in the same number of steps.

As an example, consider the simple configuration of four cubes aligned along the \( x \)-axis in figure 2. The corresponding planar arrangement consists of 12 polygons in each of the planes \( z = 1 \) and \( z = -1 \); several of these polygons coincide. Figure 4 shows the plane \( z = 1 \). The projected configuration obstacle corresponding to cubes \( P_1 \) and \( P_4 \) is the region \( R(1, 4) \) and is bounded by a line segment and two rays.

Figure 5a shows the graph \( G(R(1, 4)) \). \( R(1, 4) \) is contained in \( R(1, 2) \) and \( R(1, 3) \), so there are arcs in the graph from node 1 to nodes 2, 3, and 4, each of weight 1. \( R(1, 4) \) is contained in \( R(2, 4) \), \( R(2, 3) \), and \( R(3, 4) \). Since node 4 has no successors, it is a removable subassembly for translations in \( R(1, 4) \). If cubes \( P_2 \) and \( P_4 \) represent a single part \( P_{24} \), the graph in figure 5b results. Nodes 24 and 3 form a strongly connected component, so cubes 2, 3, and 4 must be removed simultaneously for translations in \( R(1, 4) \).
Figure 4: The arrangement for the assembly in figure 2

Figure 5: Graphs for region $R(1,4)$ where (a) $P_1, \ldots, P_4$ can be moved independently (b) $P_2$ and $P_4$ must be moved simultaneously

4 Finding Assembly Sequences

The above method can be used to decide whether there is a complete assembly sequence for an object with polyhedral parts. Here each subassembly can only be removed by a single translation, but each translation in the sequence can involve one or more parts.

Lemma 3 It can be decided in $O(k^3n^4)$ steps whether polyhedra $P_1, \ldots, P_k$ can be separated completely using motions where each subassembly is removed from the remaining objects by a single translation.

Proof: Assume that $A = \{P_1, \ldots, P_k\}$ can be disassembled. Applying the method of the previous lemma to $A$ gives two subassemblies, each consisting of one or more parts. Each application increases the number of subassemblies by
one, and the final number of subassemblies is \( k \). Therefore the above method is applied \( k - 1 \) times. □

Finally, let \( d \) be a bound for the number of binary digits used to represent the coordinates of vertices in the input assembly. The size of all intermediate values occurring in the computation is bounded by \( O(d) \). Thus using standard algorithms for rational arithmetic we obtain \( O(k^2n^4d^2) \) as a time bound for finding an assembly sequence with the above algorithm. Here all computations can be performed without loss of accuracy.

5 Experimental Evaluation

The above methods were implemented in C on a DECstation 5000 using floating-point arithmetic, with a number of modifications giving practical improvements. The program was tested on randomly generated assemblies to estimate its characteristics in the average case and find practical bounds on its application.

5.1 Implementation

A drawback of the algorithm above is the storage requirement: the arrangement may take \( O(n^4) \) space to store, which is impractical for complicated assemblies. Furthermore, the number of cells is increased dramatically by computing the arrangement of the supporting lines instead of just the boundary segments themselves. The topological sweep-line algorithm in [2, 4] sweeps over an arrangement of \( m \) lines in \( O(m) \) space and optimal \( O(m^2) \) time, but cannot be extended directly to the case of line segments instead of lines.

Our implementation addresses these problems by performing a vertical line sweep [8, 13] over the arrangement of \( O(n^2) \) line segments. This algorithm only stores \( O(n^2) \) of the cells of the arrangement at one time, and has running time of \( O((n^2 + I) \log n) \), where \( I = O(n^4) \) is the number of intersections between segments.

An imaginary vertical line passes over the arrangement. The cells cut by the sweep line in its current position are kept in a sorted list; the initial list is found by sorting the lines by slope. Start points and end points of segments and intersections between two segments are events, kept in a priority queue sorted by \( x \)-value. As the sweep-line moves from left to right, events are processed and the list of cut cells is changed accordingly. Each event can be processed in \( O(\log m) \) time, so the total running time is \( O((m + I) \log m) \), where \( I \) is the number of intersection events. In our case \( m = n^2 \). Thus the arrangement calculation requires \( O((n^2 + I) \log n) \) steps, where \( I = O(n^4) \).

The vertical sweep-line algorithm maintains the graph \( G(f) \) for each face cut by the vertical line. The graphs for faces intersecting the initial sweep-line are propagated down from an initial face at the top of the sweep-line. To process an
event, the graph for a new face is calculated by stepping from the face above it in the vertical line, as described in section 3. Thus the graphs for all faces in the arrangement are calculated and checked without keeping the whole arrangement in memory. The total computing time for finding an appropriate subassembly using the modified algorithm is $O(k^2n^4 + n^4\log n)$.

Figure 6 illustrates the processing of an intersection event. The interior of region $R(2, 3)$ is below edge $e_1$, and edge $e_2$ is the lower boundary of region $R(1, 3)$. The graphs for faces $f_1$, $f_2$, and $f_3$ have already been computed; all the graph links have weight one. When the sweep line processes the intersection of $e_1$ and $e_2$ at point $p$, the face $f_4$ is entered. Edge $e_2$ is between $f_1$ and $f_4$ in the new sweep line, so $G(f_4)$ is computed by stepping over $e_2$ from $G(f_1)$. The interior of $R(1, 3)$ is above $e_2$, so $G(f_4)$ is obtained from $G(f_1)$ by deleting the link from node 1 to node 3. Nodes 1 and 2 form a strongly connected component of $G(f_4)$, so the corresponding parts are a removable subassembly.

The implementation generates configuration obstacles for isothetic three-dimensional solids (rectangloids or boxes) instead of tetrahedra; however, the arrangement computation applies to the general case.

5.2 Experiments

To evaluate the practical computing bounds on the implementation, $n$ random disjoint boxes were generated and linked together to form $k$ complex objects for different values of $n$ and $k$. Removable subassemblies were identified for these
Table 1: Computing times for partitioning composite objects consisting of isothetic rectangular solids (units: seconds of CPU-time and 1024 Bytes)

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>t</th>
<th>$10^3t/n^4$</th>
<th>t_{min}</th>
<th>t_{max}</th>
<th>s_{max}</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>0.2</td>
<td>0.78</td>
<td>0.1</td>
<td>0.2</td>
<td>16210</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0.6</td>
<td>0.15</td>
<td>0.5</td>
<td>0.6</td>
<td>19644</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>5.1</td>
<td>0.078</td>
<td>4.7</td>
<td>5.4</td>
<td>30908</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>48.7</td>
<td>0.046</td>
<td>39.4</td>
<td>48.7</td>
<td>44163</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>283.6</td>
<td>0.016</td>
<td>281.0</td>
<td>284.2</td>
<td>61816</td>
</tr>
<tr>
<td>128</td>
<td>16</td>
<td>1150.7</td>
<td>0.0042</td>
<td>1120.8</td>
<td>1243.7</td>
<td>88264</td>
</tr>
</tbody>
</table>

Figure 7: An assembly of four random boxes

Table 1 shows a random configuration of four boxes.

Table 1 shows the computing times and storage requirements observed. For each value of n and k, 32 samples were run and the average, minimum, and maximum running times recorded ($t$, $t_{min}$, and $t_{max}$, respectively), along with the maximum storage needed ($s_{max}$). In all cases the entire arrangement and all graphs were computed instead of stopping at the first removable subassembly found.

6 Conclusions

Several extensions of the described methods might be considered. Unconnected subassemblies usually require more complicated fixtures and more difficult ma-
nipulation than do connected subassemblies. As a result, connected subassemblies are often preferred in manufacturing planning. By analyzing a connection graph of the assembly, the above algorithm can be extended to generate only subassemblies which are connected.

In practice, an arrangement of fewer segments would result from projecting the configuration obstacles of complete polyhedra. The projected configuration obstacle for polyhedra $P_1$ and $P_2$ is the union of all projected configuration obstacles $R_{ij}$ of two tetrahedra $T_i, U_j$ from $P_1$ and $P_2$. The configuration obstacles could also be found using more direct methods [8].

Finally, other types of motions could be considered. For instance, a sequence of translations might be allowed to separate subassemblies, or spatial screw displacements could be considered instead of translations.

Acknowledgements

This research was funded by DARPA contract N00014-88-K-0620 (Office of Naval Research) and the Stanford Integrated Manufacturing Association (SIMA). The authors would like to thank Prof. Dr. Herbert Edelsbrunner for his suggestions concerning the computation of planar arrangements.

References


