IEEE Intl. Conf. on Robotics and Automation, 1995, pp. 2553–2560

A Simple and Efficient Procedure for Polyhedral Assembly Partitioning under Infinitesimal Motions*

Leonidas J. Guibas† Dan Halperin† Hirohisa Hirukawa‡ Jean-Claude Latombe†
Randall H. Wilson§

Abstract

We study the following problem: Given a collection A of polyhedral parts in 3D, determine whether there exists a subset S of the parts that can be moved as a rigid body by an infinitesimal translation and rotation, without colliding with the rest of the parts, $A \setminus S$. A negative result implies that the object whose constituent parts are the collection A cannot be taken apart with two hands. A positive result, together with the list of movable parts in S and a direction of motion for S, can be used by an assembly sequence planner. This problem has attracted considerable attention within and outside the robotics community. We devise an efficient algorithm to solve this problem. Our solution is based on the ability to focus on selected portions of the tangent space of rigid motions and efficiently access these portions. The algorithm is complete (in the sense that it is guaranteed to find a solution if one exists), simple, and improves significantly over the best previously known solutions. We report experimental results with an implementation of our algorithm.

1 Introduction

In this paper we study an instance of the *assembly partitioning problem* [16]: Given a collection A of non-overlapping polyhedral parts, does there exist an infinitesimal motion (translation and rotation) that can be applied to a subset S of the parts in A, that will move S as a rigid body without colliding with the rest of the parts, $A \setminus S$? A product is *two-handed* if it can be assembled by a sequence of operations each of which merges two rigid subassemblies of the product. A negative answer to the above question, means that A is not two-handed. It may be the case that A can be assembled by more than two hands, but then the assembly process usually becomes more costly, and indeed most industrial products are two-handed. It may also be the case that A cannot be assembled.

A positive answer to the question, together with a list of the parts in S and a direction of motion for S, can be used in an assembly sequence planner. It does not provide a complete specification for an assembly step, but it points out a plausible direction of motion. Since in each assembly step we wish to merge two subassemblies from a state in which they are separated, we must produce the specification of an extended motion and thus there is still more checking to be done in order to produce such a motion, if one exists. In spite of this shortcoming, infinitesimal motions are attractive in assembly planning because their analysis translates to handling linear constraints, even when allowing rotation (see, e.g., [5],[9],[17]). Fore more information on assembly planning see, e.g., [6],[16],[18].

In 1988, in his paper "On Planning Assemblies" [12], Natarajan conjectured that "two hands suffice to assemble any composite comprised of convex polyhedra in 3-space". In a surprising result, Snoeyink and Stolfi [14] have recently been able to disprove this conjecture: They gave an example consisting of thirty convex polyhedral parts that cannot be taken apart with two hands. The proof of the validity of the construction relies on a computer program that (up to symmetries in their construction) exhaustively tries every subset of the collection of parts against the rest of the parts for infinitesimal separation (that is, by showing that there is no possible infinitesimal translation and rotation for any division of the parts). Thus their algorithm for

*Work on this paper by L.J. Guibas, D. Halperin and J.-C. Latombe has been supported by NSF/ARPA Grant IRI-9306544, and by a grant from the Stanford Integrated Manufacturing Association (SIMA). Work on this paper by L.J. Guibas and D. Halperin has been also supported by NSF Grant CCR-9215219. Work on this paper by R.H. Wilson has been supported by Sandia National Laboratories, Laboratory Directed Research and Development Program, under DOE contract DE-AC04-94AL85000.

†Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305.

‡Electrotechnical Laboratory, Agency of Industrial Science and Technology, Ministry of International Trade and Industry, 1-1-4 Umezono, Tsukuba 305 JAPAN.

§Intelligent Systems and Robotics Center, Sandia National Laboratories, Albuquerque, NM 87185.

checking infinitesimal separability is exponential in the number of parts. This is typical of several existing assembly planning techniques that rely on a "generate-and-test" approach; see, e.g., [7]. We remark that the work by Snoeyink and Stolfi continues a long line of research, whose objective was to construct composites that are interlocked under various types of motions (see, e.g., [3], [4], [12]).

An efficient procedure for the infinitesimal partitioning problem was proposed by Wilson and Matsui [17] who devised a polynomial-time algorithm to solve this problem. Their solution is based on the non-directional blocking graph (NDBG) concept [15]. See Section 2 below.

In this paper we take a similar approach to that of Wilson and Matsui, but we derive a considerably more efficient algorithm. Our approach is based on the ability to efficiently access portions of the tangent space that are relevant to our problem. We call these portions *maximally covered cells*. Our procedure finds a representative point inside each such cell. We show, that those representative points are sufficient to cover all the potential families of feasible motions, thus making our algorithm complete, namely, if there exists a solution our algorithm will find it.

Our method is not restricted in dimension and it can be applied to various problems involving any number of degrees of freedom. It does, however, rely on the fact that the number of degrees of freedom is not too big. For infinitesimal rigid motions in three-dimensional space this number is five. We believe that our technique can be useful in other areas as well. In Section 6 we support this claim by showing how it can be used in model-based object recognition.

We have implemented the algorithm and we present experimental results together with several practical considerations.

The rest of the paper is organized as follows. In Section 2 we supply more background which is needed to explain our algorithm. In Section 3 we expose the ideas underlying our new approach. These are then used in the algorithm which we present in Section 4, where we also analyze its running time. In Section 5 we present experimental results produced by the implementation of the algorithm. In Section 6 we briefly review the application of the approach to model-based object recognition. Some concluding remarks and open problems are given in Section 7.

2 Background

The starting point of our new approach is similar to that of Wilson and Matsui [17]. In this section we briefly review some of the ingredients of their analysis that are needed here as well. We refer the reader to their paper [17] for more details.

The *non-directional blocking graph* (NDBG, for short) is a subdivision of the space of all allowable motions of separation into a finite number of cells such that inside any single cell the blocking relation between all pairs of parts is fixed. These blocking relations for a fixed motion (and hence for a cell in the subdivision) are gathered in a directed graph, the directional blocking graph (DBG), whose nodes $v_1, v_2, \ldots, v_n$ represent the n parts $P_1, P_2, \ldots, P_n$ in the assembly, and a directed arc from node v_i to node v_j means that the part P_i will collide with the part P_j if this motion is applied to P_i. A partitioning of the full assembly A into two subassemblies under a specific motion is possible, if and only if, the DBG for that motion is not *strongly connected.* For more details on the NDBG concept, see [15],[16].

Contacts between polyhedra consist of points, line segments, and planar polygonal contacts. The infinitesimal motion constraints arising from line segments and planar polygonal contacts can be reduced to equivalent finite sets of point-plane contact constraint. For example, the contact between a convex edge e of one polyhedron and a face f of another polyhedron is equivalent to two point-plane constraints, one at each end of the intersection segment of e and f. (For more details, see [5],[13],[17].) We therefore concentrate on point-plane contact constraints.

An infinitesimal motion ΔX of a polyhedron P_i can be described as a vector with three parameters for translation and three for rotation:

$$\Delta X = (\Delta x, \Delta y, \Delta z, \Omega_x, \Omega_y, \Omega_z)\ ,$$

where Ω_x, Ω_y, and Ω_z are the rotational components of ΔX around the x, y, and z axes, respectively. Now, consider the point-plane contact c between the vertex v_c of a polyhedron P_i and the face of a polyhedron P_j with outward normal n_c. The infinitesimal motion ΔX causes the vertex v_c of P_i to undergo a translation $J_c \Delta X$, where J_c is the 3×6 Jacobian matrix that relates the differential motion of P_i to the motion of v_c.

An infinitesimal motion ΔX such that $n_c^T J_c \Delta X < 0$ will cause the part P_i to penetrate into P_j at the contact point v_c. Therefore this point-plane contact will allow only local motions ΔX such that $n_c^T J_c \Delta X \geq 0$, and when this inequality holds we say that ΔX *obeys* the contact c. Equality (i.e., when $n_c^T J_c \Delta X = 0$) means that the infinitesimal motion causes a sliding of one polyhedron relative to the other at the contact. The set of infinitesimal motions allowed by all the point-plane constraints involving one polyhedron P_i is the intersection of the motions that obey each constraint individually.

Since two motions ΔX_1 and ΔX_2, with $\Delta X_1 = s \cdot \Delta X_2$ for a positive scalar s differ only in velocity, we restrict ourselves to motions ΔX such that $|\Delta X| = 1$. Hence our motions are all represented by points on the unit sphere $\mathcal{S}^5$ in six-dimensional space. Each point-plane contact defines a hyperplane that divides $\mathcal{S}^5$ in half, and determines a closed hemisphere (whose boundary is a great circle on $\mathcal{S}^5$) of infinitesimal motions that obey that specific contact.

For convenience, we choose a hyperplane Π tangent to $\mathcal{S}^5$ and centrally project the great circles introduced by the constraints on $\mathcal{S}^5$, onto that hyperplane. Now our constraints are transformed into closed halfspaces in R^5. This way we have only projected a hemisphere of $\mathcal{S}^5$ onto the special hyperplane Π. However, it is easily verified that with a little caution we do not lose any information by this transformation. Let Π' be the hyperplane parallel to Π and passing through the origin. We choose Π such that Π' does not cross vertices on $\mathcal{S}^5$ (i.e., points where 5 constraint hyperplanes or more meet), and then the projected hemisphere contains all the information necessary to find a partitioning if one exists, because the other hemisphere has a symmetric subdivision on it. In other words, if a point p on the sphere represents a motion that will separate S from $A \setminus S$, then the antipodal point of p will represent the separation of the same two subassemblies in precisely the opposite direction.

3 Maximally Covered Cells

Our novel and more efficient approach is based on several observations that we explain in this section.

First, we group the contact constraints for each ordered pair of parts (P_i, P_j). We denote the collection of closed halfspaces that represent constraints on the motion of P_i relative to P_j by Q_{ij}. It follows from the discussion in the previous section, that the intersection of all the constraints in Q_{ij} is the convex polytope representing the motion directions in which P_i will *not* collide into P_j. With a slight abuse of notation we will refer to Q_{ij} both as a collection of closed halfspaces and as the convex polytope that is the intersection of these halfspaces.

Next, we consider the subdivision of 5-space induced by all the constraints Q_{ij} for all the ordered pairs of parts in our assembly. It is evident that inside each cell of any dimension in this subdivision, the blocking relation, and hence the blocking graph (the DBG, see Section 2) is fixed. The crux of our new technique is the observation that we need to consider only some of the cells in this subdivision and that there is a way to access these cells directly without computing the entire subdivision.

Since our approach seems to be applicable in other settings as well, we describe it more generally from this point. Let Q denote the collection of K polytopes in d-dimensional space (in our application, these are the at most $n(n-1)$ polytopes Q_{ij} in 5-dimensional space, where n is the number of parts in the assembly). Let $\mathcal{A}(Q)$ denote the subdivision of d-space induced by this collection, namely the collection of relatively open cells of dimensions $0, 1, \ldots, d$ induced by the boundaries of all the polytopes in Q. For example, a d-dimensional cell in the subdivision is a maximal portion of d-space not meeting any boundary of any polytope in Q. Let N be the total number of facets in all the polytopes together, i.e., N is the overall number of constraint halfspaces.

Definition 3.1 *The* **covering set** *of a point p in R^d is the subset of polytopes in Q that contain p.*

We now turn to discuss *maximally covered cells.* Informally, a cell is maximally covered, if there is no way to augment its covering set just by crossing its boundary, or in other words:

> A cell is maximally covered if every point outside the cell and infinitesimally close to the cell is covered by only a subset of the polytopes that cover the cell.

Since we deal with *closed polytopes* it is equivalent to require that any point on the relative boundary of the cell will have the same covering set as its interior. Since a maximally covered cell and its relative boundary have the same covering set, dealing with both the cell and each of its bounding faces is redundant. Hence, we require that the closure of a maximally covered cell is a maximal connected region of d-space with that covering set. We summarize the above discussion in the following

Definition 3.2 *A cell C in the subdivision $\mathcal{A}(Q)$ is called a* **maximally covered cell** *if the covering set of any point on the relative boundary of C is the same as the covering set of its interior, and the closure of C is a maximal connected region of d-space with that covering set.*

To get a feeling for what this definition says, consider Figure 1, which depicts a collection of convex polygons in the plane. There are four maximally covered cells in the subdivision defined by the six polygons in the

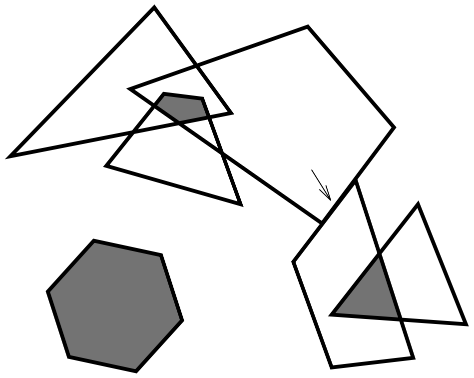

Figure 1: Maximally covered cells in a planar subdivision

figure. Three of the maximally covered cells are two-dimensional cells and they are shaded in the figure: the shaded hexagon is covered by one polygon and every point outside the hexagon and in its immediate neighborhood is not covered by any polygon. The shaded triangle is covered by two polygons, and the shaded pentagon is covered by three polygons. There is also one maximally covered cell that is one-dimensional. This is the segment of intersection between two polygon boundaries pointed to by the arrow. Every point along this intersection segment is covered by the two polygons (recall that we deal with closed polytopes) and every point in the neighborhood of this segment but not lying on it is either covered by a single polygon or by none.

Back to the original problem, we argue that if there is a solution (S, p) to our partitioning problem, namely there is a subset S of the parts in A, and a direction p of motion that will separate it infinitesimally from the rest of the parts, then there is a solution (S, p') such that p' is a point inside a maximally covered cell. To see why this claim is true, consider a partitioning (S, p) such that p is not inside a maximally covered cell. But then there is a point q on the boundary of the cell containing p that is covered by more polytopes than its interior. Since we deal with closed polytopes, by moving to the boundary of a cell C we cannot get out of any polytope covering the interior of C. We move to the point q on the boundary, and move further infinitesimally across that boundary into the highest-dimensional cell we can get to without crossing another constraint boundary, and denote the new point by p'. We claim that (S, p') is still a valid partitioning: this is evident because we only removed a blocking constraint when we have moved to p'. If the cell containing p' is maximally covered, then we are done, otherwise we continue as above. This process is finite, and we are guaranteed to stop inside a maximally covered cell.

What do we gain from the above observation? We show next that the number of maximally covered cells is potentially smaller than the overall number of cells in the entire subdivision.

Theorem 3.3 *The maximum number of maximally covered cells in $\mathcal{A}(Q)$ is $O(K^d)$.*

Proof: We choose a direction D in R^d such that every bounded maximally covered cell has a minimum point in direction D. (The unbounded cells of the subdivision can be easily shown to include only a negligible number of maximally covered cells.) This minimum point is a vertex v of the subdivision where at least d facets of polytopes in Q meet. There may be more than d facets meeting at this point in which case we choose exactly d facets lying in d distinct hyperplanes and we denote by Q' the set of up to d polytopes that contain these facets on their boundary. The point v is clearly the minimum point in the D direction in the intersection of the polytopes in Q', and this intersection is a unique convex polytope. Hence we can charge the maximally covered cell to this intersection polytope. There are at most $\binom{K}{d}$ such intersection polytopes, and the bound follows. $\square$

For infinitesimal separation, Wilson and Matsui consider the entire subdivision induced by the hyperplanes supporting the facets of polytopes in Q, therefore they consider $\Theta(N^5)$ cells. The overall number of cells in the subdivision $\mathcal{A}(Q)$ is $\Theta(N^2K^3)$ in the worst case [1]. The number of cells that our algorithm examines is $\Theta(K^5)$. Note that K is never bigger than N; in practical situations K is often much smaller than N. In the next section we show how we directly access these cells without computing the entire subdivision.

4 The Algorithm

4.1 Finding representative points

The idea behind the algorithm is the same as in the proof of Theorem 3.3, namely, the minimal vertex of any maximally covered cell in a fixed direction D is a minimal vertex in direction D of the convex polytope which is the result of intersecting at most d polytopes in the given collection Q.

We choose the positive direction along the coordinate X_d as the fixed direction. Our algorithm therefore looks for the X_d minimal vertex of the intersection of any set of up to d polytopes in Q. Fortunately, since we are only interested in a single representative point, we do not have to compute the intersection of the polytopes. Instead, we use linear programming (LP, for short), where the constraints are the halfspaces determining the polytopes, and the objective function we wish to minimize is X_d.

4.2 Overall algorithm

Recall that the set of points that the above algorithm produces contains all *candidate* representative directions. We still need to check for each point if it actually represents a feasible direction of collision-free infinitesimal motion. We do this by constructing the directional blocking graph (DBG) at each direction and checking it for strong connectivity.

Since we do not construct the entire subdivision of the space of possible motion directions, we cannot use adjacency relations between cells of the subdivision to incrementally update the DBG as we move from one cell to another, as in [17]. Rather, we construct the DBG from scratch at each point. To do that we build, for every polytope in our collection, a data structure that will enable us to efficiently determine whether a point is contained in the polytope or not. If a given point g is *not* contained in a given polytope Q_{ij} this means that we have to put an arc directed from the node v_i to the node v_j (corresponding to the parts P_i and P_j respectively) in the blocking graph. Otherwise, there is no arc between the two nodes.

To summarize, here is a sketch of the algorithm for the partitioning problem with infinitesimal motions where the space of directions has dimension d. The input is a set of K constraint sets Q_{ij} for each pair of ordered parts (P_i, P_j) in contact, and the output is a subset of parts and a direction to move it, if one exists, or INTERLOCKED otherwise. In the variable H we collect all the constraints defining a given subset R of the polytopes of one subproblem. The variable e will contain the answer point from the LP program or NULL if there is no feasible solution. The function $DBG(e)$ returns the DBG at the point e. The procedure $LP(H, X_d \downarrow)$ runs an LP for the constraints H and the objective function X_d.

1. *for* $i = 1, \cdots, d$
2. *for each subset* R *of input polytopes,*

   such that $\mid R \mid = i$
3. $H \leftarrow \bigcup_{r \in R}(Halfspaces\ in\ r)$
4. $e \leftarrow LP(H, X_d \downarrow)$
5. *If* $e = NULL$ *goto* 2, *else*
6. *If* $DBG(e)$ *is not strongly connected*
7. *Output* e *and movable subassembly*
8. *exit*
9. *Report* "INTERLOCKED"

A preliminary step, omitted in the algorithm description above, derives the point-plane contacts that are used to define the input hyperplane constraints. For that purpose, we use the algorithm devised by Hirukawa et al. [5]; see there for details.

Note that the algorithm may produce points in cells that are not maximally covered. Also, the set of candidate points produced may change if we choose a different direction D along which we look for minimal vertices. However, the algorithm is guaranteed to produce a point inside each maximally covered cell.

4.3 Complexity analysis

The main loop of our algorithm performs $O(K^d)$ iterations, and in each iteration the major steps performed are: (i) solving a linear programming problem, (ii) computing a DBG, and (iii) checking the DBG for strong connectivity. In this subsection we analyze the worst-case running time of the algorithm using the best known procedures for each step. Our implementation, with which the experimental results reported below were obtained, uses different procedures to implement steps (i) and (ii), for reasons of software availability and simplicity.

Lemma 4.1 *Solving all the linear programs together takes time* $O(K^{d-1}N)$.

Proof: We have to solve $O(K^d)$ linear programming problems. Recall that we assume that the dimension d is a small constant. We will focus on the problems that arise when we take all possible combinations of exactly d polytopes; this will dominate the running time of solving all the other LP problems. We use Megiddo's algorithm that runs in time linear in the number of constraints [8]. Thus asymptotically the overall running time of all the LP problems equals the overall number of constraints given to all these problems. Let us fix one constraint h, belonging to one polytope T (i.e., to one set of constraints). It is easily verified that this constraint will participate in $O(K^{d-1})$ LP problems, which is all the possibilities to choose the other $d-1$ polytopes that are not T. Since there are N constraints in all the polytopes together, the bound follows. □

Computing the DBG can be reduced to a collection of polytope membership queries. For one polytope with m facets in dimension d, the preprocessing takes $O(m^{\lfloor \frac{d}{2} \rfloor + \epsilon})$ expected time, and allows for query time $O(\log m)$ [10]. Let m_i be the number of facets of the ith polytope in the given collection. The total preprocessing time is

$$\sum_{i=1}^{K} O(m_i^{\lfloor \frac{d}{2} \rfloor + \epsilon}) \;=\; O(N^{\lfloor \frac{d}{2} \rfloor + \epsilon}) ,$$

and the overall query time is

$$O(K^d) \sum_{i=1}^{K} O(\log m_i) = O(K^{d+1} \log N) .$$

The time to check for strong connectivity is linear in the number of nodes and arcs in the DBG and hence it is dominated by its construction time.

We summarize

Theorem 4.2 *The partitioning of a polyhedral assembly under infinitesimal motion, where any direction of motion is defined by d parameters, can be computed in $O(K^{d-1}N + K^{d+1} \log N)$ time after preprocessing in $O(N^{\lfloor \frac{d}{2} \rfloor + \epsilon})$ expected time, where K is the number of ordered pairs of polyhedral parts in contact in the assembly (thus K is at most $n(n-1)$ for an assembly with n parts), and N is the total number of the contact constraints among the parts.*

Rephrased in our original setting, namely for partitioning under infinitesimal rigid motions we have

Theorem 4.2$'$ *The partitioning of a polyhedral assembly under infinitesimal rigid motion (translation and rotation), can be computed in $O(K^4 N + K^6 \log N)$ time after preprocessing in $O(N^{2+\epsilon})$ expected time, where K is the number of ordered pairs of polyhedral parts in contact in the assembly (thus K is at most $n(n-1)$ for an assembly with n parts), and N is the total number of the contact constraints among the parts.*

5 Experimental Results

The algorithm described above has been implemented and in this section we present selected experimental results. (We have conducted more experiments and these will be reported in a forthcoming full version of the paper.) The linear programming problems are solved using the MINOS package [11]. The contact constraints are computed by the algorithm described in [5].

The first example, given by Snoeyink and Stolfi [14], consists of six identical tetrahedra in contact and shown in Figure 2. They proved that no proper subset is separable by infinitesimal translation. We revisit this example, confirm their result with our program, and show that if we allow general infinitesimal motion (i.e., including rotation), then this construction can be partitioned. In this example, $K = 24$ and $N = 96$.

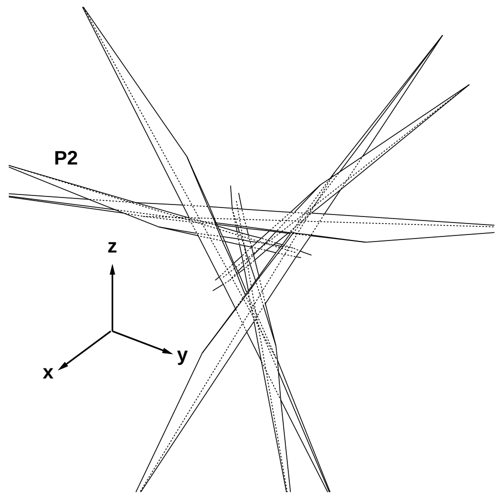

Figure 2: Six tetrahedra in contact [14]

In the case of translation only, where $d = 2$, 22 representative points are found by the algorithm, and all of the corresponding DBGs are confirmed to be strongly connected. In the case of rigid motions, where $d = 5$, 190 representative points are found, and only 118 of the corresponding DBGs are strongly connected. That is, some proper subset is separable in 92 directions of infinitesimal motion with rotation. Examples of the DBGs for $d = 2$ and $d = 5$ are shown in Figure 3. The DBG for $d = 2$ is strongly connected, and that for $d = 5$ is not. In the latter case, Part P2 is separable by the infinitesimal motion shown in the figure, where the direction of the motion is represented in the coordinates in Figure 2. This example requires 4.0 seconds CPU time to compute the constraints, 2.7 seconds to find the representative points, 27.0 seconds to check the strong connectivity of the DBGs on conventional workstations[1] for $d = 2$, and 4.0, 1286.9 and 240.5 seconds respectively for the case $d = 5$.

The second example is an engine of a model-aircraft whose parts are shown in Figure 4. Figure 5 shows the engine assembled from the parts. In this example, the

[1] The constraints computation as well as checking for strong connectivity of the DBGs are run on a *SUN SPARC Station 1+* and the algorithm for finding representative points in maximally covered cells is run on a *DEC 5000*.

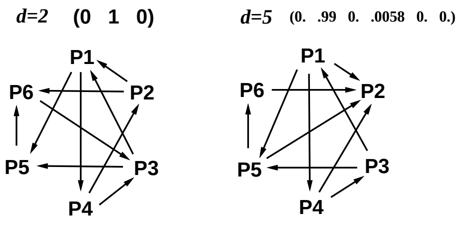

Figure 3: Examples of the DBGs

number of the parts is 12, the total number of their faces is $1,066$, $K = 24$, and $N = 1,112$.

In the case of infinitesimal translation, four representative points are found, and none of the corresponding DBGs is strongly connected. In the case of general infinitesimal motion, six representative points are found, and here also none of the DBGs is strongly connected. One of the DBGs, corresponding to an infinitesimal rotation, is shown in Figure 6. The representative point is represented by the coordinate system depicted in Figure 5. This example requires 1881.7 seconds to compute the constraints, 12.6 to find the representative points, and 59.9 for checking strong connectivity for $d = 2$, and 1881.7, 7880.1 and 120.25 seconds respectively for $d = 5$.

6 Application to Object Recognition

Another application of maximally covered cells arises in *model-based object recognition* when using computer vision. In a typical situation, one is given a 3D model of an object and a 2D image containing the object. Both the model and the image data are represented in terms of *geometric features*, consisting of elements like points, line segments, and curve normals. The goal is to determine the pose of the 3D object in space that best explains the features observed in the image. In terms of geometric features, this is a matching problem—but it is complicated by spurious image features, missing model features, and general geometric uncertainty in real data. An approach to this problem that has been developed by Cass [2] proceeds as follows. Define a *match* to be a pairing of a model feature and an image feature, and a *match set* as an arbitrary set of matches. A match defines a subset of *pose space* (the set of all possible placements of the 3D object with respect to the projection center) that causes the object feature in question to match under projection the image feature in question (within whatever error tolerance we use). In general, a match naturally corresponds to some feasible region of the pose space. A match set is geometrically consistent when there exists a subset of the pose space that simultaneously realizes all the matches in the set. If we think of the boundary of a feasible match region as a constraint surface, then the set of all possible matches defines an arrangement in pose space whose cells clearly reflect the consistent match sets. It is clear that for our recognition task, only the maximally covered cells are interesting: they correspond to object poses that succeed in aligning a (locally) maximal number of object and image features. It is among those that we need to select the best one.

7 Conclusions

We have presented a new, simple and efficient approach to the partitioning problem of polyhedral assemblies under general infinitesimal motions. The algorithm improves considerably over the best previously known algorithms for this problem, and appears to perform well in practice. The new solution seems to be of independent interest and we have shown how it can be used to solve a different problem arising in the context of object recognition.

Our algorithm was implemented and we have reported experimental results. Currently, our main goal is to further improve the running time of our program so that it can handle larger and more difficult examples. We are also looking for more applications where maximally covered cells in subdivisions induced by convex polytopes may be of interest.

Acknowledgment

The authors thank Michael A. Saunders for letting us use the MINOS package, and Rhea Tombropoulos for her generous assistance with the linear programming code. We thank Jack Snoeyink and Jorge Stolfi for valuable discussions concerning the contents of the paper. Finally, we thank Jack Snoeyink for supplying us with data of the constructions in [14].

References

[1] B. Aronov, M. Bern and D. Eppstein, Arrangements of polytopes, *manuscript*, 1991.

[2] T.A. Cass, Robust geometric matching for 3D object recognition, *Proc. Intl. Conf. on Pattern Recognition (ICPR)*, Jerusalem, 1994, pp. 477-482.

[3] R. Dawson, On removing a ball without disturbing the others, *Mathematics Magazine* **57** no. 1 (1984), pp. 27–30.

[4] L. Fejes Toth and A. Heppes, Uber stabile Körpersysteme, *Compositio Mathematica* **15** no. 2 (1963), pp. 119–126.

[5] H. Hirukawa, T. Matsui and K. Takase, Automatic determination of possible velocity and applicable force of frictionless objects in contact, *IEEE Trans. Robotics and Automation* **10** no. 3 (1994), pp. 309–322.

[6] L.S. Homem de Mello and S. Lee, editors, *Computer-Aided Mechanical Assembly Planning*, Kluwer Academic Publishers, Boston, 1991.

[7] L.S. Homem de Mello and A.C. Sanderson, A correct and complete algorithm for the generation of mechanical assembly sequences, *IEEE Trans. Robotics and Automation* **7** no. 2 (1991), pp. 228–240.

[8] N. Megiddo, Linear programming in linear time when the dimension is fixed, J. ACM **31** (1984), pp. 114-127.

[9] R.S. Mittikalli and P.K. Khosla, Motion constraints from contact geometry: Representation and analysis, *Proc. IEEE International Conference on Robotics and Automation*, Nice, 1992, pp. 2178–2185.

[10] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall, New York, 1993.

[11] B.A.Murtagh and M.A.Saunders, MINOS 5.4 USER'S GUIDE, Technical Report SOL 83-20R, Department of Operations Research, Stanford University, 1993.

[12] B.K. Natarajan, On planning assemblies, *Proc. 4th ACM Symposium on Computational Geometry*, 1988, pp. 299–308.

[13] M.S. Ohwovoriole and B. Roth, An extension of screw theory, *Trans. ASME, J. Mechanical Design* **103** (1981), pp.725-735.

[14] J. Snoeyink and J. Stolfi, Objects that cannot be taken apart with two hands, *Discrete and Computational Geometry*, **12** (1994), pp. 367–384.

[15] R.H. Wilson, *On Geometric Assembly Planning* Ph.D. Dissertation, Computer Science Department, Stanford University, March 1992.

[16] R.H. Wilson and J.-C. Latombe, Geometric reasoning about mechanical assembly, *Journal of Artificial Intelligence*, **71** no. 2, 1994, pp. 371–396.

[17] R.H. Wilson and T. Matsui, Partitioning an assembly for infinitesimal motions in translation and rotation, *Proc. IEEE International Conference on Intelligent Robots and Systems*, 1992, pp. 1311–1318.

[18] J.D. Wolter, *On the automatic generation of plans for mechanical assembly*, Ph.D. Thesis, The University of Michigan, Ann Arbor, 1988.

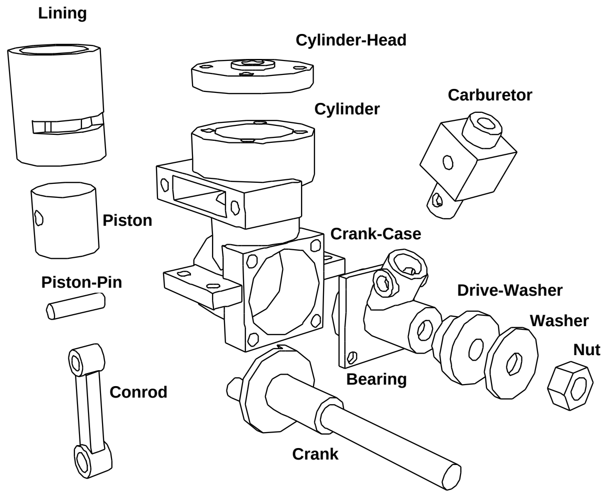

Figure 4: Parts of a model-aircraft engine

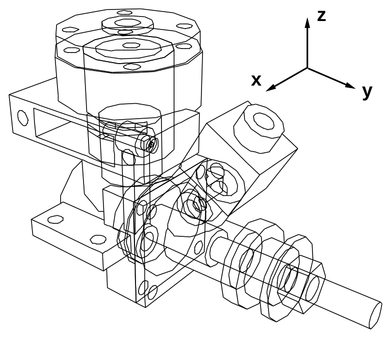

Figure 5: Assembled engine

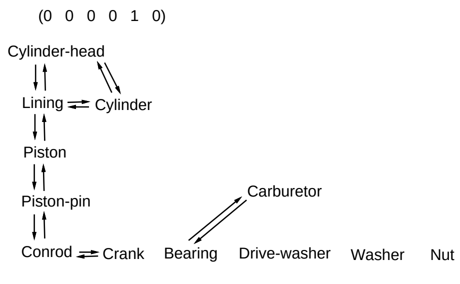

Figure 6: Example of a DBG for $d = 5$