General Translational Assembly Planning

Achim Schweikard, Fabian Schwarzer
Informatik
Technische Universität München, 80290 München
schweika@informatik.tu-muenchen.de

Abstract: *An exact and practical method for translational motion planning with many degrees of freedom is derived. It is shown that certain $D-$dimensional arrangements of hyperplanes can be searched in the following way: only a single connected component is traversed during the search, and the arrangement is searched as an arrangement of surface patches rather than full hyperplanes. This reduction in search effort allows for polynomial time bounds in appropriate cases. Heuristic and randomized planners cannot return an information about infeasibility of planning problems. The new method is exact and complete. Experiments obtained with an implementation suggest that it is capable of reaching the performance of the most advanced random planning schemes. In addition, insolvability can be established in practical cases.*

Keywords: *geometric reasoning, assembly planning, motion planning, complete algorithms, arrangement computation in D dimensions.*

Introduction

It is conjectured that few basic principles suffice to represent human abilities in geometric reasoning and motion planning. However, human reasoning is capable not only of finding a solution to a given motion planning problem rapidly, but also of quickly recognizing that no feasible solution exists. Intuitively, recognizing infeasibility seems more difficult than finding one motion, if one exists. To establish that the input problem is infeasible, a proof is required.

In this context we will address the following problem:

- Given an assembly of polygonal or polyhedral parts, decide whether one or more parts can be removed from the remaining set of parts by an arbitrary sequence of translations.

This includes lock-and-key configurations, where several groups of parts must move simultaneously into distinct directions, and/or change direction during motion. The number of translations in such a sequence is not limited.

Example 1: Parts P and Q in fig. 1 are movable, the container is fixed. The goal is to decide whether or not part P is removable, after an appropriate series of translational motions of both P and Q.

To explain why P is not removable, one would enumerate certain critical intermediate placements of both P and Q, and test for removability in each such placement. The main problem with reproducing such an intuitive approach in an algorithm is the decision about which of the possible intermediate placements are indeed critical.

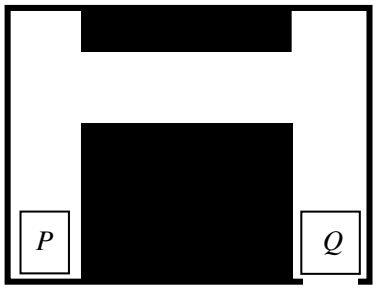

Figure 1: Detecting whether a part can be removed from other parts. Parts P and Q are movable, the container is fixed

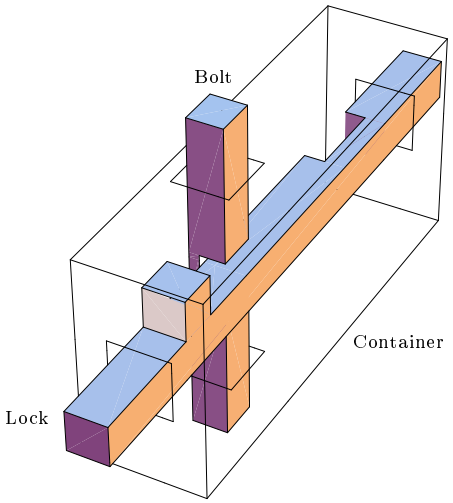

Figure 2: To test whether or not the bolt is removable, intermediate placements of the lock must be searched.

In the second example (fig. 2), intermediate positions of lock must be tested to decide that the bolt is not removable. There are comparatively few critical placements of the lock. In contrast, the complete enumeration of all distinct relative placements of vertices and bounding planes of parts will quickly become impossible even for simple cases.

We are thus interested in methods for reducing the set of critical relative placements of parts, *while retaining completeness.* To obtain a practical method, it is necessary to recognize the essential contacts between parts. But notice that we would detect infeasibility in error, if the set of essential contacts thus chosen was too small.

1 Related Work

The problem of finding a motion for removing an object from a set of other objects arises in assembly planning. In this context, Toussaint [12] surveys earlier methods for separating sets in two and three dimensions.

Natarajan [10] derives lower bounds on the number of simultaneous translations necessary for separating objects.

Pollack, Sharir and Sifrony [13] describe a near-optimal method for the case of two polygons translating in the plane. The analysis in [13] is based on computing the boundary of a single connected component in a planar arrangement of line segments.

Agarwal, de Berg, Halperin and Sharir [1] consider sequences of translations for separating polyhedra. In [1] the set of allowed motion directions is assumed to be given in advance.

Kavraki [5] shows that the problem of separating sets is NP-complete even for a very restricted class of planar objects.

Schweikard and Wilson [14] describe an algorithm for computing a *subassembly* S of a given assembly, such that S is removable by a single translation, if such a subassembly exists. The output consists of S and an appropriate removal direction d. Interestingly this computation is possible in polynomial time, even though the number of removable subassemblies is exponential in general. The algorithm in [14] will thus compute a valid subassembly and a removal direction if there is such a subassembly, but does not enumerate all possible subassemblies.

In [15] techniques for analyzing arrangements defined by radiation beams in radiosurgery are described. These techniques are based on searching so-called minimal and maximal cells in three dimensions and are related to the methods developed here. The minimal and maximal cells do not form a connected component, and for the application in [15] it is sufficient to search the minimal and maximal cells after computing the entire arrangement.

Guibas and Halperin et al. [3] describe methods for partitioning three-dimensional assemblies under infinitesimal motions. The algorithm in [3] allows for computing one allowed infinitesimal motion in polynomial time, and uses methods in [14]. An infinitesimal motion may not give a valid motion for removing parts, since parts may still collide during the motions extending the computed infinitesimal motion. However, the techniques in [3] include rotations.

The techniques in [14] are not used here, since they inherently require that all moving parts follow the same motion path.

Instead, the analysis of 3-dimensional arrangements given by pairwise Minkowski-differences of parts is used to guide the search of a $D-$dimensional arrangement.

The next section gives an informal description of the basic principles in this context. The main idea of the present approach is described in section 3. Indeed, there is a remarkably simple way to compute certain multi-dimensional arrangements as arrangements of *surface patches*, while avoiding the computation of the entire underlying arrangement of *hyperplanes*. However, this is only possible for a particular class of arrangements. The arrangements to be considered here all belong to this class. Section 4 derives an algorithm from these principles and proves completeness. Section 5 describes the implementation. Section 6 gives the analysis of the mentioned reductions. Section 7 describes experimental results. The experiments suggest that it is possible to decide about infeasibility in a complete way. The evaluation compares the derived methods to approximate and randomized motion planners. It is shown that the above principles lead to an exact motion planning algorithm of surprising performance in experiments.

2 Basic Concepts

Let $P_1, \ldots, P_k$ be an assembly of three-dimensional parts. Thus $P_1, \ldots, P_k$ are non-intersecting polyhedra in given spatial placement.

We allow for all parts to translate in space independently. The position of each part is given by three parameters. A space E^D of dimension D describes all simultaneous placements of all parts. Here $D = 3k$. A point in this space is called forbidden, if two or more parts intersect in the corresponding placement.

The origin in E^D represents the initial placement of all parts, and is not forbidden.

We consider pairwise Minkowski-differences ($C-$obstacles, [7]) for each pair of parts P_i, P_j. The $C-$obstacle for P_i and P_j is a three-dimensional region, defined by $C(P_i, P_j) = \{p_j - p_i \mid p_i \in P_i, p_j \in P_j\}$.

The $C-$obstacles determine a set of halfspaces $H_1, \ldots, H_s$ in E^D, such that the forbidden regions are bounded by the corresponding hyperplanes in the following sense.

$H_1, \ldots, H_s$ determine an *arrangement* A_D of halfspaces in E^D, and partition E^D into *cells*. A cell is a maximal connected region not containing any points on any of the hyperplanes bounding $H_1, \ldots, H_s$. Cells are thus open $D-$dimensional sets. Cells are regular, i.e. for each cell c either all or no points in c are forbidden. The partitioning given by $H_1, \ldots, H_s$ also defines cells of dimension less than D. Lower dimensional cells are regular as well.

Notice that A_D contains *unbounded* cells. An unbounded cell is a cell entirely containing a ray in its interior. A valid removal motion for one or more parts is a sequence of non-forbidden cells connecting the origin to one of the unbounded cells.

A direct way to obtain an answer to the problem stated above is the following: We compute a graph representation of the arrangement A_D ([2]). Graph nodes for unbounded cells are

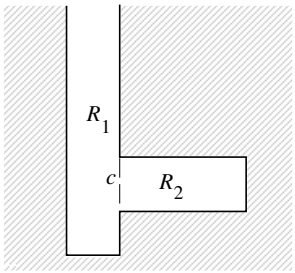

Figure 3: Door cell c. All points in both R_1, R_2 are visible from points in c.

labelled. Searching this graph will give an exact solution to the above problem.

However, this direct method cannot be used in practice, for the following reason: The graph representation of A_D has an exponential number of nodes, and it is not possible to store this graph even for a small number of polyhedra each with few faces.

To see whether parts are removable, it suffices to search a single connected component of the arrangement, namely the component containing the origin. It is obvious that searching only a single component will often reduce storage requirements to some extent, but this reduction is insufficient in practice. However, this simple idea provides a basis for a more effective reduction of search effort and storage requirements, described in the next section.

3 Floorgraphs

In the following we assume that all parts $P_1, \ldots, P_k$ are open sets. Thus placements of parts in which two parts are in contact, but do not overlap are not regarded as intersections. Motions can consist of segments where two or more parts slide along each other. We also assume that all parts are bounded sets.

Constraints for pairs of parts. For two parts P_i and P_j, the bounding planes of $C(P_i, P_j)$ define an arrangement of planes $A(P_i, P_j)$ in three-dimensional space.

We partition the complement of $C(P_i, P_j)$ into convex regions. Each convex region thus obtained is given as the intersection of halfspaces. Let $H_1^{ij}, \ldots, H_r^{ij}$ be the halfspaces stemming from P_i and P_j. These halfspaces are given as inequalities in the three parameters x, y and z.

Let R_1 and R_2 be two adjacent convex regions in the partitioning of the exterior of $C(P_i, P_j)$. Then there is a cell c in the (three-dimensional) arrangement $A(P_i, P_j)$ such that all points in R_1 are visible from any point in c, and all points in R_2 are visible from any point in c (fig. 3). We call c a *passage* or *door cell*.

A *floorgraph* is defined in the following way. Each convex region in the above partitioning of the exterior of $C(P_i, P_j)$ corresponds to one node in the floorgraph. If c is a door cell

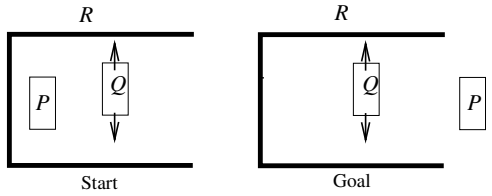

Figure 4: Deciding whether a part can be removed from other parts. Part Q is constrained to move in the vertical direction only

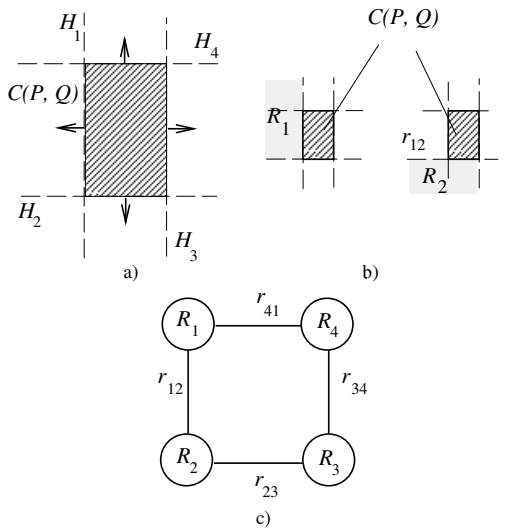

Figure 5: a) Halfspaces $H_1, ..., H_4$ bounding $C(P,Q)$. b) Regions R_i partitioning the exterior of $C(P,Q)$ into convex regions. Door cell r_{12}. c) Floorgraph for pair (P,Q).

shared by two regions R_1 and R_2, then c defines an edge in the floorgraph. To each node/edge we assign a set of defining inequalities and equalities.

Example 2: The assembly in fig. 4 contains three parts P, Q, R. R is the (fixed) container. P and Q are rectangular. $C(P,Q)$ is a rectangle (fig. 5-a), and the exterior of $C(P,Q)$ is a union of halfspaces $H_1, \ldots, H_4$. The orientations of $H_1, \ldots, H_4$ are indicated by arrows in the figure.

By convention, the halfspace obtained by reversing the inequality for H_i is denoted by H_i^-. Similarly, the bounding plane of H_i is denoted by $H_i^=$. The exterior of $C(P,Q)$ is thus partitioned into four regions $R_1, \ldots, R_4$, where $R_1, \ldots, R_4$ are given by

$R_1 : H_1^+ \cap H_2^-$, $R_2 : H_2^+ \cap H_3^-$,
$R_3 : H_3^+ \cap H_4^-$, $R_4 : H_4^+ \cap H_1^-$.

The floorgraph for $C(P,Q)$ is shown in fig. 5-c. The partitioning gives four door cells:

$r_{12} : H_1^+ \cap H_2^=$, $r_{23} : H_2^+ \cap H_3^=$,

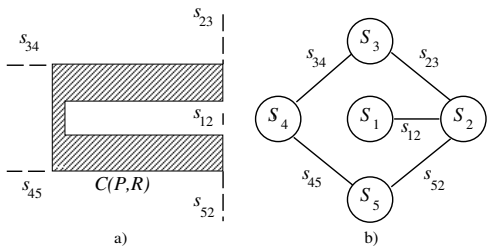

Figure 6: a) C-obstacle $C(P,R)$ for parts P, R in example 1. b) Floorgraph for pair (P,R).

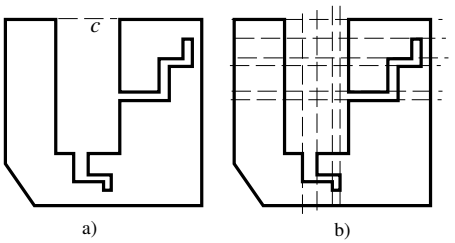

Figure 7: An arrangement of surface patches (a) has smaller number of cells than the corresponding arrangement of (extended) hyperplanes (b). c is a door cell.

$r_{34} : H_3^+ \cap H_4^=$, $r_{41} : H_4^+ \cap H_1^=$.

The floorgraph for $C(P,R)$ is shown in fig. 6. Since we constrain Q to move in the vertical direction only, the floorgraph for (Q,R) can be represented by a single node T_1.

Simultaneous motions of all parts. The $D-$dimensional arrangement A_D is constructed from the halfspace inequalities $H_1, \ldots, H_s$ in the following way.

The position of each part P_i is given by three parameters $p_x^{(i)}, p_y^{(i)}, p_z^{(i)}$. These parameters describe the placement of P_i with respect to a fixed initial placement. Thus a point $(p_x^{(1)}, p_y^{(1)}, p_z^{(1)}, \ldots, p_x^{(k)}, p_y^{(k)}, p_z^{(k)})$ describes a simultaneous placement of all parts $P_1, \ldots, P_k$.

Each halfspace inequality H_i contains the three variables x, y and z. For an inequality H stemming from a pair (P_i, P_j), we replace the variable x by the expression $p_x^{(i)} - p_x^{(j)}$. Similarly, substitute $p_y^{(i)} - p_y^{(j)}$ for y and $p_z^{(i)} - p_z^{(j)}$ for z.

After this substitution, a pair (P_j, P_i) gives the same inequalities as the pair (P_i, P_j). (This follows from the fact that $C(P_j, P_i) = -C(P_i, P_j)$ by definition). Thus it is sufficient to consider each pair (P_i, P_j), where $i < j$.

The inequalities are $\leq$-inequalities and not $<$-inequalities, because parts are open sets. We regard all inequalities thus obtained as $D-$dimensional inequalities, where at most six variables have non-zero coefficients.

We must fix the position of one (arbitrary) part, since we are testing for removability. Otherwise all ($D-$dimensional) cells in A_D would be unbounded. We thus set the parameters $p_x^{(k)}, p_y^{(k)}$ and $p_z^{(k)}$ of part P_k to zero.

In the next section it will be shown that floorgraphs allow for searching A_D in the following way:

- Only a single connected component within A_D is traversed. We do not need to enumerate all free cells in A_D.
- A_D is searched as an arrangement of *surface patches* (fig. 7), rather than an arrangement of full hyperplanes.

4 Searching floorgraphs

In the above construction, one floorgraph corresponds to each pair of parts P_i, P_j where $i < j$.

Let $G_1, \ldots, G_f$ be the floorgraphs thus obtained. Let $n_1, \ldots, n_f$ be nodes in the floorgraphs, where n_i is a node in G_i for each i. The tuple $S = (n_1, \ldots, n_f)$ will be called a $D-$node.

Thus, a $D-$node contains exactly one node of each floorgraph. For a node n in one floorgraph, let $C(n)$ be the set of defining constraint inequalities. Similarly $C(e)$ gives the defining constraints for an edge e in one floorgraph. If $x \epsilon E^D$ satisfies the constraints $C(n_1), \ldots, C(n_f)$ then S is called a feasible $D-$node.

The $D-$nodes cover free space in E^D, i.e. for each non-forbidden point a, there is a $D-$node $S = (n_1, \ldots, n_f)$ such that a satisfies $C(n_1), \ldots, C(n_f)$.

In particular, we can find a $D-$node for the origin in E^D.

We define a *successor* of a $D-$node in the following way.

Let

$$S = (n_1, ..., n_{i-1}, n_i, n_{i+1}, ..., n_f)$$

be a $D-$node. Then

$$S' = (n_1, ..., n_{i-1}, n_i', n_{i+1}, ..., n_f)$$

is a successor of S, if n_i' is a successor of n_i in the floorgraph G_i.

Similarly, we define a *successor-edge* of a $D-$node:

$$S' = (n_1, ..., n_{i-1}, e, n_{i+1}, ..., n_f)$$

is a successor edge of

$$S = (n_1, ..., n_{i-1}, n_i, n_{i+1}, ..., n_f)$$

if e is an edge emerging from n_i in the floorgraph G_i.

In general, a single $D-$node has several successors. Thus the successor relation on $D-$nodes defines a graph. We search this graph in depth first order, where all $D-$nodes previously visited are marked and not visited again:

1. Compute all floorgraphs for pairs P_i, P_j with $i < j$.

2. Compute a $D-$node S for the origin in E^D.
3. Set $L = \{S\}$.
4. Repeat until L = empty
   (a) If L = empty, return result 'infeasible', and stop.
   Set P = first node in L. Remove P from L.
   (b) If P is unbounded, return the path from O to P and stop.
   Set L' = successor edges of P.
   (c) For each entry S in L', test whether S is feasible. If so, store a point x satisfying the constraints in S, and a pointer from S to x.
   (d) Remove infeasible entries from L'. Replace each successor edge in L' by the corresponding successor node. Move all entries in L' to the front of L. Remove all previously visited nodes from L.

Notice that each intermediate point x stored with a $D-$node S (except the origin) is a *point in a door cell.* Thus each point x computed in 4-c satisfies the constraints in one floorgraph *edge* and $f-1$ floorgraph *nodes.* In step 4-d, the constraints for this edge are replaced by the constraints for the successor node in the corresponding floorgraph. New nodes generated at the initialization of the list L' in 4-b again have constraints for $f-1$ floorgraph nodes and one floorgraph edge. Segments of the paths can be contained entirely in cells of dimension less than D.

Let $S = (n_1, \ldots, n_f)$ be a $D-$node. In the remainder of this section we will regard S as a point set in E^D, i.e. S is identified with the set of points satisfying the constraints in $C(n_1), \ldots, C(n_f)$. Similarly, we will associate nodes and edges in floorgraphs with the corresponding point sets in E^3. This is done to simplify the notation in the following topological observations.

We must show that the sequence of points thus computed indeed yields a feasible path.

Lemma 1 *Let S_1 be a feasible $D-$node and S_2 be a feasible successor edge of S_1. Let x_1 be a point satisfying the constraints in S_1, and x_2 be a point satisfying the constraints in S_2. Then all points on the line segment connecting x_1 to x_2 in E^D are in free space.*

Proof: The proof directly follows from the convexity of the D-nodes.

Two $D-$nodes S and S' will be called adjacent, if their intersection is non-empty.

Lemma 2 *Let $S = (n_1, \ldots, n_f)$ and $S' = (n'_1, \ldots, n'_f)$ be two adjacent $D-$nodes. Then for each i with $n_i \neq n'_i$, the nodes n_i and n'_i are connected by an edge e in G_i. Both $C(n_1), \ldots, C(n_{i-1}), C(e), C(n_{i+1}), \ldots, C(n_f)$ and $C(n'_1), \ldots, C(n'_{i-1}), C(e), C(n'_{i+1}), \ldots, C(n'_f)$ are feasible.*

Proof. We require that the (3D) partitioning of the exterior of each $C-$obstacle be such that two distinct regions have disjoint interior. Thus all points in the intersection of two floorgraph nodes are points in door cells. Assume, $n_i \neq n'_i$. If $n_i \cap n'_i$ is empty, then $S \cap S'$ is also empty, contradicting the adjacency of S and S'. Thus n_i and n'_i are connected by an edge e in G_i. Let x be in $S \cap S'$. Then x satisfies both $C(n_1), \ldots, C(n_f)$ and $C(n'_1), \ldots, C(n'_f)$. All points satisfying both $C(n_i)$ and $C(n'_i)$ also satisfy $C(e)$, due to the above property of the 3D partitionings. Thus x satisfies $C(n_1), \ldots, C(n_{i-1}), C(e), C(n_{i+1}), \ldots, C(n_f)$ as well as $C(n'_1), \ldots, C(n'_{i-1}), C(e), C(n'_{i+1}), \ldots, C(n'_f)$. •

The completeness of the above method follows from lemma 2:

Lemma 3 *Let u be a continuous path connecting the origin to an unbounded cell, such that all points on u are in free space. Then there is a sequence $T_1, \ldots, T_r$ of $D-$nodes, which covers u, such that T_{j+1} is a successor of T_j in the search graph.*

Proof: The $D-$nodes cover free space in E^D. Thus u traverses a sequence $S_1, \ldots, S_r$ of $D-$nodes, covering u. The $D-$nodes are closed sets, so that (by continuity) for $j < r$, $S_j \cap S_{j+1}$ is non-empty, i.e. S_j and S_{j+1} are adjacent.

We must show that S_{j+1} will indeed be expanded, once S_j has been expanded. Thus it remains to be shown that S_{j+1} is either a direct successor of S_j in the search graph, or S_j and S_{j+1} are connected by a chain of feasible $D-$nodes in this graph.

Let $S_j = (n_1, \ldots, n_f)$ and $S_{j+1} = (n'_1, \ldots, n'_f)$. Assume $n_1 \neq n'_1$. By lemma 2, n_1 and n'_1 are connected by and edge e, and $C(e), C(n_2), \ldots, C(n_f)$ is feasible. Thus the $D-$node $S^{(1)}$ defined by $S^{(1)} = (n'_1, n_2, \ldots, n_f)$ is feasible. $S^{(1)}$ is a successor of S_j, since it is connected to S_j by the (feasible) edge e.

Since S_j to S_{j+1} are adjacent, there is a point x satisfying both $C(n_1), \ldots, C(n_f)$ and $C(n'_1), \ldots, C(n'_f)$. Thus x satisfies $C(n'_1), C(n_2), \ldots, C(n_f)$, i.e. the constraints for $S^{(1)}$. But then x is a common point of $S^{(1)}$ and S_{j+1}, so that $S^{(1)}$ is adjacent to S_{j+1}. Thus we can apply the same argument to the pair n_2, n'_2. If n_2, n'_2 are distinct, then the $D-$node $S^{(2)} = (n'_1, n'_2, n_3, \ldots, n_f)$ is feasible and adjacent to S_{j+1}. Furthermore $S^{(2)}$ is a successor of $S^{(1)}$. Repeating this replacement step at most f times, we will obtain a sequence of $D-$nodes connecting S_j to S_{j+1} in the search graph via a chain of feasible edges. •

Remark: Let n and n' be two non-adjacent nodes in one floorgraph G. Then the constraint hyperplanes in n do not partition the region corresponding to n' and vice versa. Similarly, a door cell between two regions is a patch, not intersecting regions determined by other nodes in G. In this sense, the regions in A_D corresponding to floorgraph nodes are bounded by surface patches rather than full hyperplanes.

5 Implementation

To obtain a practical algorithm, we must implement the test for feasibility in step 4-c of the above algorithm. This test is implemented as a simplex feasibility test ([9]). If positive,

the test returns a point x satisfying the given constraints. In the simplex test, we must ensure that variables may become negative. Here standard methods apply.

Furthermore, we must implement the test for boundedness in step 4-b. We must thus decide whether the current $D-$node contains a ray u. Let S be this $D-$node. u may have points in the boundary of S, i.e. on the defining hyperplanes of S, but must not cross these hyperplanes.

Let

$$H_1 : n_1 x - d_1 = 0, \ldots, H_r : n_r x - d_r = 0$$

be the equations of the hyperplanes defining S. $n_1, \ldots, n_r$ are the normal vectors of the hyperplanes $H_1, \ldots, H_r$. Let x_0 be the point in S computed in step 4-c. Assume these hyperplane equations are oriented such that x_0 is above (or in) all hyperplanes $H_1, \ldots, H_r$, i.e. $n_1 x_0 - d_1 \geq 0, \ldots, n_r x_0 - d_r \geq 0$. To decide whether there is a vector u such that $x_0 + tu$ does not cross one of $H_1, \ldots, H_r$ for any $t > 0$ it suffices to test whether there is a *non-zero* vector u with $n_1 u \geq 0, \ldots, n_r u \geq 0$.

Let $H : nx - d = 0$, where $d \neq 0$ be an arbitrary hyperplane. Then $H' : nx + d = 0$ is parallel to H. Both H and H' do not contain the origin. To find u we use two simplex tests. The first test decides whether there is a u with $nu - d = 0, n_1 u \geq 0, \ldots, n_r u \geq 0$. The second test decides whether there is a u with $nu + d = 0, n_1 u \geq 0, \ldots, n_r u \geq 0$. As above, in both tests we must allow for coordinates of u to become negative. S is unbounded if one of the tests succeeds. However, the test may fail to detect unboundedness. This condition occurs if the set of all vectors u representing valid removal directions is entirely contained in the hyperplane $nx = 0$. In the implementation we can ignore this condition, for the following reason. All parts are bounded sets. Thus free space in E^D contains a single unbounded component. This unbounded component contains at least one $D-$node with full dimension D.

Other tests for unboundedness could be based on simplex maximization or quadratic programming. The reason for choosing the above test is the following. The result of direct maximization depends on the particular implementation of the simplex method if S is unbounded. In contrast, the above test is independent of the implementation.

6 Analysis

The above method outputs a path - if one exists - as a sequence of line segments in E^D. Each segment represents a simultaneous translation of one or more parts, possibly in distinct directions. To find such a path, a tree of line segments is expanded internally.

We will first consider the number of node expansion steps taken by the algorithm. We decompose the faces of each part into triangles. Let n be the maximum number of triangles in each of the k parts. Then n is a bound for the number of vertices of each part. To compute the pairwise Minkowski-differences, it suffices to compute the Minkowski-differences for pairs of triangles on faces, and we will obtain $O(n^2)$ inequalities for each pair of parts. We obtain a total of $O(k^2 n^2)$ halfspaces in A_D. Each step will reach a new cell in A_D, and cells will not be visited again. A_D has at most $O((k^2 n^2)^D)$ cells [2]. Here $D = 3k - 3$ for polyhedral assemblies and $D = 2k - 2$ in the planar case.

Lemma 4 *The number of node expansions in the above algorithm is bounded by $O((k^{2D} m^D)$, where m is the maximum number of nodes in a connected component of a floorgraph and k is the number of parts.*

Proof. Each node in each floorgraph represents a convex (generalized) cylinder in E^D. (The cylinders are called generalized cylinders because their bounding hypersurfaces are linear, not curved). The set of feasible points for a $D-$node is the intersection of $O(k^2)$ such cylinders. At each step, we proceed from one *convex* cell of this cylinder arrangement to the next *convex* cell. Notice that the cylinder arrangement contains non-convex cells as well. At most one unbounded $D-$node will be examined, so we must only account for bounded nodes. There is a vector v in E^D such that each bounded convex cell has exactly one extremal vertex in direction v. Each vertex is the intersection of exactly D hyperplanes, except in cases, where hyperplanes are not in general position. After a sufficiently small displacement of all hyperplanes, we can reach a placement, in which each $v-$extremal vertex is the intersection of exactly D distinct hyperplanes. Of course this displacement is only done for accounting purposes, not in the program. The displacement of each hyperplane can be made sufficiently small such that none of the convex cells of the cylinder arrangement will vanish. Each v-extremal vertex of each convex cell is the intersection of at most D cylinders. There are $O(k^2 m)$ cylinders. We consider the number of subsets with at most D elements in this set of size $O(k^2 m)$. This number is bounded by

$$\binom{k^2 m}{1} + \ldots + \binom{k^2 m}{D} \leq \sum_{i=1}^{D} \frac{(k^2 m)^i}{i!}$$

$$\leq (k^2 m)^D \sum_{i=1}^{D} \frac{1}{i!} \leq (k^2 m)^D e,$$

where e is the Euler constant.

The first step in this chain of inequalities is the least obvious, but will become clear if we observe that for any $a \geq i$

$$\binom{a}{i} = \frac{a!}{i!(a-i)!} = \frac{1 \cdot \ldots \cdot (a-i)(a-i+1) \cdot \ldots \cdot a}{i! \cdot (1 \cdot \ldots \cdot (a-i))} \leq \frac{a^i}{i!}.$$

Thus there are $O((k^2 m)^D)$ extremal vertices of convex cells in the cylinder arrangement. Each such convex cell has one $v-$extremal vertex, i.e. there are $O((k^2 m)^D)$ convex cells in the cylinder arrangement.•

Since k and D are related, the above bound on the number of node expansions can be reduced:

(Again m is the maximum number of nodes in a connected component of a floorgraph and k is the number of parts.)

We consider only the case of three-dimensional parts.

Lemma 5 *The number of node expansions in the above algo-*

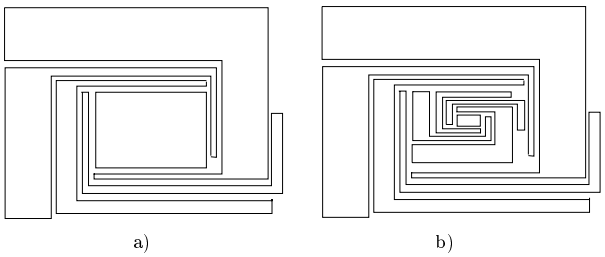

Figure 8: a) An assembly with interlocking parts. b) Successively replacing the rectangular center part in fig.-a by a copy of the original assembly gives a series of interlocking assemblies (See also tab. 1).

rithm is bounded by $O(k^{f(D)}m^{g(D)})$, where f and g are functions with $f(D) < D$ and $g(D) < D$ for any D larger than a constant D_0 .

Proof: From the proof of the previous lemma, the expression $\binom{k^2m}{1} + \ldots + \binom{k^2m}{D}$ is a bound for the number of node expansion steps. We must only consider ordered pairs of parts (P_i, P_j) where $i < j$, i.e. the value $q = \binom{\frac{k^2}{2}m}{1} + \ldots + \binom{\frac{k^2}{2}m}{D}$ equally bounds the number of node expansions. If k is sufficiently large, we have $\binom{\frac{k^2}{2}m}{i} \leq \binom{\frac{k^2}{2}m}{D}$ for each $i = 1, \ldots, D$. By Stirling's formula,

$q = O(D\frac{m^D}{2^D}\frac{k^{2D}}{\sqrt{D}D^De^{-D}})$. Rearranging gives

$q = O(\sqrt{D}\frac{m^D}{2^D}k^{2D-D\frac{\log D-1}{\log k}}) = O(k^{f(D)}m^{g(D)})$,

with $f(D) = 2D - D\frac{\log D-1}{\log k} + \frac{1}{2}$ and $g(D) = D(1 - \frac{\log 2}{\log m})$. In the three-dimensional case $D = 3k-3$ and $\frac{\log D-1}{\log k} > 1$ for sufficiently large k and D, so that $f(D) < D$ and $g(D) < D$. •

From the number of expansion steps, one can directly obtain a bound for the running time of the algorithm:

Let $LP(s, D)$ be the number of steps required for solving a linear program with s constraints and D variables. Then each node expansion will require at most $O((k^2r)LP(k^2r, D))$ steps, where r is the maximum number of constraints in each floorgraph node or floorgraph edge. The total running time is thus bounded by $O((k^2m)^D)k^2rLP(k^2r, D))$.

In this bound we have not accounted for the precomputation of pairwise floorgraphs. The analysis of this preprocessing step is straightforward and follows methods in [11] and [14].

Remark: The above lemma expresses the number of examined $D-$nodes in terms of the maximum number m of nodes in the current connected component of floorgraphs, rather than in the number n of triangles in faces of parts. In practice, the current connected component of many floorgraphs (such as the one for the lock/container in fig. 2) consists of a single node.

The parts in fig. 8 interlock. Here $m > 1$. However, the total number of examined $D-$nodes is 1, since all successors of nodes in floorgraphs are unreachable. Assuming $LP(s, D)$ is polynomial, we obtain a polynomial time bound for cases in which the number of examined $D-$nodes remains constant.

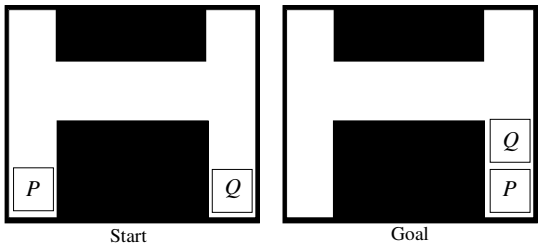

Figure 9: Moving part P from start to goal position.

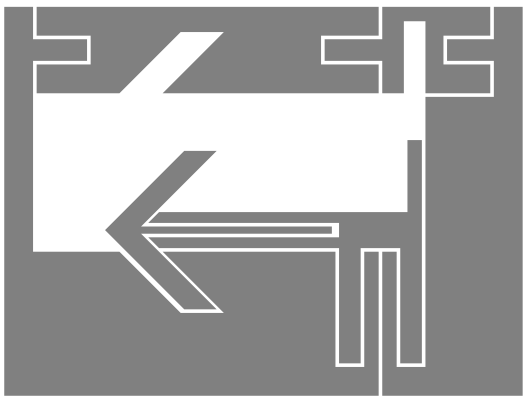

Figure 10: Simulataneous motion of parts into distinct directions is required for removing parts. Clearance between parts in the actual experiment was set to zero.

7 Experiments

To find practical limitations, the above algorithm was implemented on a Unix-workstation HP 700 in C++ based on the LEDA-library [8].

For comparison, a random planner based on the principles in [4, 6] was implemented for the mentioned hardware environment. Both methods were then applied to the example in figure 9. The running time of approximate planners depends upon the width of the smallest passage the planner must find. Here the ratio between edge length of moving parts and width of passages was set to 9/10. In this case the random planner takes on the order of 8 seconds to find a solution. This computing time increased if the width of the passages was reduced. The running time for the exact method was not affected by this width, and a the computing time for finding a path is equally below 1 second. Infeasibility (after appropriate changes of the container) was detected in the same amount of time.

We next consider the example in fig. 8. The parts in this assembly interlock. This condition was detected by the program in 1.3 seconds. We replace the central rectangle in this

k	5	9	13	17
t_{prep}	0.6	2.8	6.5	11.6
t_{search}	0.6	3.2	8.7	19.3
t_{search}/k^5	0.00019	0.000054	0.000023	0.000013

Table 1: Computing times for establishing infeasibility in the example in fig. 8, with growing number k of parts (in seconds CPU-time).

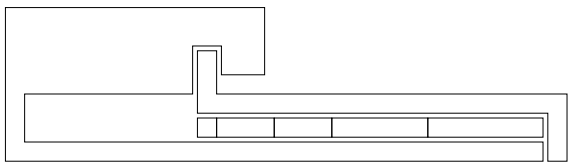

Figure 11: To find a motion for removing a part, the algorithm must solve PARTITION for integers ([11]).

assembly by an identical (but smaller) copy of the original assembly in fig. 8. The new assembly has 9 parts (fig. 8-b). Repeating this replacement step several times, we obtain a series of interlocking assemblies, with growing number of parts. Our experiments confirm that the running time in this case is polynomial (table 1).

In the example in fig 10 it is necessary to move parts into distinct directions simultaneously. In the experiment, the parts were slightly enlarged, so that the clearance between appropriate parts was zero. In this case the running times were 6.40 seconds for preprocessing (computing all floorgraphs) and 11.1 seconds for finding a removal motion. This example was not solvable with any of the random planners, even when the goal position was given in advance.

An example in [11] provides a simple proof for the NP-hardness of the problem considered here. Specifically, the 'PARTITION'-problem for integers must be solved to find a removal motion in fig. 11. The running time for finding such a motion in this example was 22.8 seconds including 0.78 seconds preprocessing. 1143 D-nodes are expanded in this case, and the output path consists of 224 segments. In this case, the ordering in which the $D-$nodes were expanded was not a strict depth-first ordering. Instead, better performance was achieved by expanding the $D-$node with largest distance from the origin in E^D first. This simple heuristic is implemented by moving the node to be expanded first to the front of list L in step 4-d of the above algorithm.

Preliminary experience with the exact method suggests that it is capable of reaching the performance of advanced random planning schemes. Insolvability can be established. But notice that the exact method is still limited to translational motion planning, which is not the case for random planners. The implementation of the exact method is comparatively simple and consists of very little program code given standard packages for arrangement computation and linear programming. Experimentally we have observed that the storage requirements are comparatively small.

8 Conclusions

Assembly motions requiring many changes of directions or velocities increase assembly costs, and are often impractical due to fixturing and stability problems. From a practical point of view, finding one assembly motion - if there is such a motion - is the main objective of planning methods. In this context it seems useful to fix an upper bound for the number of allowed changes in direction/velocity of parts. The number of $D-$nodes along a path bounds the number of direction/velocity changes during the motions. However, the direct computation of paths with minimum number of direction/velocity changes has not yet been explored in the context of the above methods.

References

[1] P. K. Agarwal, M. de Berg, D. Halperin, M. Sharir. Efficient Generation of k-Directional Assembly Sequences. *Proc. 7th ACM-SIAM Symp. Discr. Algorithms (SODA)*, 122-131, 1996.

[2] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer, Heidelberg, 1987.

[3] L. Guibas, D. Halperin, H. Hirukawa et al. Polyhedral assembly partitioning using maximally covered cells in arrangements of convex polytopes. *To appear: Intl. J. Comp. Geometry and Applications.*

[4] L. Kavraki. Randomized preprocessing of configuration space for fast path planning. *IEEE Intl. Conf. Rob. Automation*, 2138–2145, 1994.

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

[6] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[7] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[8] K. Mehlhorn, S. Naeher. LEDA: A Platform for Combinatorial and Geometric Computing. *Max-Planck-Institut fuer Informatik, Saarbruecken, 1995.*

[9] J. J. More', S. J. Wright. Optimization Software Guide. *Frontiers in Applied Mathematics*, 14, SIAM, 1993. See also: "http://www.mcs.anl.gov/home/ otc/Guide/SoftwareGuide/".

[10] B. K. Natarajan. On planning assemblies. In *Proc. of the ACM Symp. on Computational Geometry*, pages 299–308, 1988.

[11] J. O'Rourke. Computational Geometry in C. Cambridge University Press, Cambridge, 1994.

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

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

[14] A. Schweikard and R. H. Wilson. Assembly Sequences for Polyhedra. *Algorithmica*, 13(6): 539–552, June 1995.

[15] A. Schweikard, R. Tombropoulos, L. Kavraki, J. R. Adler, J.-C. Latombe. Treatment Planning for a Radiosurgical System with General Kinematics. *IEEE Conference Robotics and Automation*, 1720–1727, May 1994.