

# The International Journal of Robotics Research

<http://ijr.sagepub.com>

---

## Automatic Generation of High-Level Contact State Space

Jing Xiao and Xuerong Ji

*The International Journal of Robotics Research* 2001; 20; 584

DOI: 10.1177/02783640122067552

The online version of this article can be found at:  
<http://ijr.sagepub.com/cgi/content/abstract/20/7/584>

---

Published by:

 SAGE Publications

<http://www.sagepublications.com>

On behalf of:



Multimedia Archives

Additional services and information for *The International Journal of Robotics Research* can be found at:

**Email Alerts:** <http://ijr.sagepub.com/cgi/alerts>

**Subscriptions:** <http://ijr.sagepub.com/subscriptions>

**Reprints:** <http://www.sagepub.com/journalsReprints.nav>

**Permissions:** <http://www.sagepub.com/journalsPermissions.nav>

**Citations** (this article cites 10 articles hosted on the SAGE Journals Online and HighWire Press platforms):  
<http://ijr.sagepub.com/cgi/content/refs/20/7/584>

---

**Jing Xiao**  
**Xuerong Ji**

Computer Science Department  
University of North Carolina at Charlotte  
Charlotte, NC 28223, USA  
xiao@uncc.edu  
xji@lucent.com

# Automatic Generation of High-Level Contact State Space

## Abstract

*A divide-and-merge approach is introduced for automatic generation of high-level, discrete contact state space, represented as contact state graphs, between two contacting polyhedral solids from their geometric models. Based on the fact that a contact state graph is the union of the subgraphs called a goal-contact relaxation (GCR) graph, the approach consists of algorithms (1) to generate a complete GCR graph automatically given the most constrained contact state in the GCR graph and (2) to merge GCR graphs automatically. The algorithms are implemented for cases in which the most constrained contact state in a GCR graph consists of up to three principal contacts. The ability to capture and represent contact state information effectively and efficiently is essential for robotic operations involving compliant motions, for simulation of contact motions, and for haptic interactions.*

**KEY WORDS**—contact state graphs, contact formations, polyhedral solids, goal-contact relaxation graphs, robotic assembly

## 1. Introduction

The work in this paper is motivated by two related goals: one is to enable a computer to capture and represent high-level contact state information effectively and efficiently, and the other is to facilitate automatic planning and control of compliant motions.

A high-level contact state as opposed to a low-level contact configuration captures the topological and physical characteristics of contact often common to two or more contact configurations. For instance, the contact state of “a coffee mug sitting on a table” means the bottom surface of the mug contacting the top surface of the table, which is usually shared by infinitely many mug configurations relative to the table, and within these configurations, contact motions do not change degrees of freedom. A graph of such discrete,

high-level contact states, where an arc links adjacent contact states, is often needed in many tasks that require information about contact geometry, including automatic assembly or fine-motion planning (Buckley 1989; Dakin and Popplestone 1992; Desai 1989; Xiao and Volz 1989; McCarragher 1996; Sturges and Laowattana 1995; Kang et al. 1997), virtual prototyping, haptic interactions, and so on. Such a contact state graph is crucial for stratification of compliant control strategies (De Schutter et al. 1999; Bruyninckx and Schutter 1998; Shekhar and Khatib 1987) and for simulation of collision responses and contact motions. However, the information is usually fed manually into the system as input (i.e., contact states and the relations among contact states are enumerated and presented to the system manually). This is tedious for even tasks of simple geometry (Sturges and Laowattana 1995) and is practically infeasible for complex tasks due to the huge number of different contact states. Therefore, automatic generation of contact state graphs is necessary. Although it is relatively straightforward to generate such graphs automatically for convex polyhedra (Hirukawa, Papegay, and Matsui 1994), the problem remains open for nonconvex objects.

Many robotic tasks involve contact motions. Researchers have long discovered that by taking advantage of contact constraints through compliance, uncertainties of motion can be reduced along with reduced degrees of freedom (Inoue 1974; Lozano-Pérez, Mason, and Taylor 1984; Mason 1982; Whitney 1985). It is clearly desirable to be able to plan and control contact motions automatically.

From a classic motion-planning point of view, planning contact motions means planning motions on the surface of configuration space obstacles (C-obstacles) (Lozano-Pérez 1983). The key is to know the C-obstacles. However, computing C-obstacles exactly in high-dimensional configuration space remains a formidable problem. Most of the work in the literature is limited to three-dimensional C-obstacles (i.e., C-obstacles of two-dimensional objects) (Avnaim, Boissonnat, and Faverjon 1988; Brost 1989; Rosell, Basañez, and Suárez 1997; Sacks and Bajaj 1998), and only a few studies

concern the approximation of C-obstacles of three-dimensional polyhedra (Donald 1985; Joskowicz and Taylor 1996). On the other hand, contact motions, unlike collision-free motions, require exact knowledge of contact configurations. Hence, a recent trend is to explore contact motion planning without explicitly computing C-obstacles (Hirukawa 1996).

We reckon that the goals of acquiring high-level contact state information and of planning general contact motion effectively and efficiently can be best achieved by tackling the following two related problems (of two levels):

1. Automatic generation of a high-level contact state graph.

One of the many uses of such a contact state graph (as introduced earlier) is to allow easy generation of a high-level plan for contact motion as a sequence of contact state transitions through graph search.

2. Planning contact motions between two known adjacent contact states.

This problem can be further decomposed into (a) planning an instantaneous contact transition (between the two adjacent contact states) and (b) planning contact motions within the same known contact state, which is a motion-planning problem of lower dimension and smaller scope.

We focus on the first problem and present a novel approach toward automatically generating discrete contact state graphs for general polyhedral objects. Our approach is characterized by directly exploiting both topological and geometrical knowledge of contacts in the physical space of objects and by dividing the problem into simpler subproblems of generating and merging special subgraphs.

The paper is outlined below. In Section 2, we review the notion of contact formation (CF) in terms of principal contacts (PCs) (Xiao 1993) to characterize contact states, and we define contact states as CF-connected regions of contact configurations. We also examine the neighboring relations between contact states and characterize the contact state space as a contact state graph. In Section 3, we describe our approach for automatic generation of the so-called goal-contact relaxation (GCR) graphs (Xiao 1997), which are subgraphs of a contact state graph, and automatic merge of the GCR graphs to form a contact state graph between arbitrary polyhedra. In Section 4, we describe and discuss the implemented algorithms, their scopes of application, and some implementation results, and refer to the multimedia extensions for more implementation material, including sample data, codes, and displays. We summarize the work and discuss further research to conclude the paper in Section 5.

## 2. Contact State Space

One important question is how and at what level of abstraction a contact state should be defined and described. Topological

representation is most commonly used. Usually, certain types of topological contact primitives are defined in terms of contacting topological surface elements, and a contact state is characterized in terms of the topological contact primitives formed. In this section, we first review PCs, which are higher level topological contact primitives compared to other commonly used contact primitives, and explain why we use such primitives to describe a topological contact state. We then give the geometrical interpretations of the topological contact primitives, discuss connectivity, and define contact states.

### 2.1. Topological Contact Primitives

Three types of topological contact primitives have been used most often in the literature. One representation is the point-contact notion for polyhedra introduced by Lozano-Pérez (1983) and Donald (1985) in their attempt to construct C-obstacles. Here, for two polyhedra  $A$  and  $B$  in contact, contact primitives are defined as point contacts of the following types: for two-dimensional polygons, type A (Edge $_A$ -Vertex $_B$ ) and type B (Vertex $_A$ -Edge $_B$ ); and for three-dimensional polyhedra, type A (Face $_A$ -Vertex $_B$ ), type B (Vertex $_A$ -Face $_B$ ), and type C (Edge $_A$ -Edge $_B$ , which are not collinear).

Another representation was introduced in Desai et al. (1988) and Desai (1989), which used a single contact between a pair of topological surface elements (i.e., faces, edges, and vertices) of two polyhedra as primitives (known as elemental contacts [ECs]).

The notion of PCs (Xiao 1993) are higher level primitives compared to both the point-contact primitives and the ECs. Denoting the boundary elements of a face as the edges and vertices bounding the face, and the boundary elements of an edge as the vertices bounding the edge, a PC can be defined topologically as follows.

**DEFINITION 1.** A PC denotes the contact between a pair of surface elements (i.e., faces, edges, or vertices) that are not the boundary elements of other contacting surface elements (if there is more than one pair in contact).

There are 10 types of PCs as shown in Figure 1. Each nondegenerate PC is associated with a contact plane, defined by a contacting face or the two contacting edges in an e-e-cross PC. Each degenerate PC is characterized as between two convex edge or vertex elements and not being associated with a contact plane. Such PCs hardly occur in practice. Note that except for the v-v type, every other degenerate PC is associated with a contact line, defined by a contacting edge. The normal of a contact line can be specified as the one along the direction of the sum of the outward normals of the faces forming the contacting edge.

A PC between a convex edge/vertex and a concave edge/vertex is regarded not as a single PC but as a contact state consisting of nondegenerate PCs, taking into account the roundness of edges/vertices and the small deformation due to contact in reality, as shown in Figure 2.

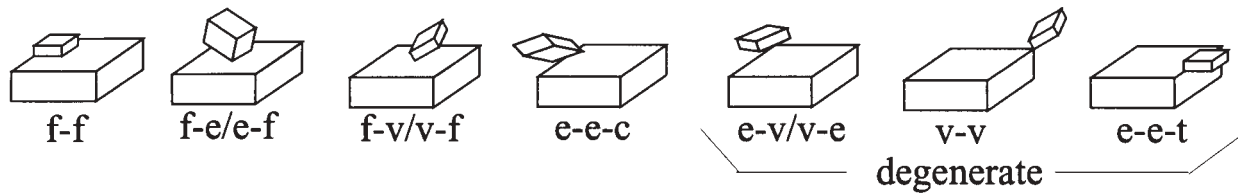


Fig. 1. Principal contacts: face-face (f-f), face-edge/edge-face (f-e/e-f), face-vertex/vertex-face (f-v/v-f), edge-edge-cross (e-e-c), edge-vertex/vertex-edge (e-v/v-e), vertex-vertex (v-v), edge-edge-touch (e-e-t).

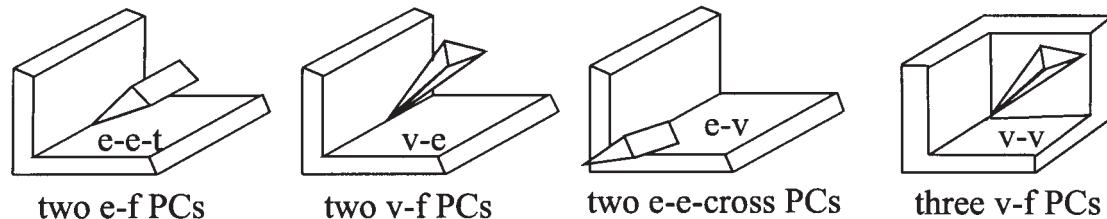


Fig. 2. Contact cases described in terms of nondegenerate principal contacts (PCs).

That PCs are higher level primitives compared to both the point-contact primitives and the ECs is most evident from the fact that every contact between two convex polyhedra forms a single PC, but not so for either the point-contact primitives or the ECs. This leads to important advantages in characterizing contact states by PCs:

- PCs lead to the more concise description and smallest number of topological contact states. A contact between two convex polyhedra forms a single PC; on the other hand, if more than one PC is formed in a contact, there must be at least one nonconvex object in the contact. Using PCs as contact primitives simplifies geometric reasoning of contact constraints (Xiao and Zhang 1997).
- PCs are sufficient to capture precisely the reduced degrees of freedom as the result of contact, which is essential for implementing compliant motions.
- PCs as higher level contact primitives enable more robust recognition in the presence of sensing uncertainties. For a nondegenerate PC (Fig. 1), one good property is that its change to another nondegenerate PC is often coincident with the discontinuity of the general contact force (i.e., force and torque), which means that the change can be captured by force/torque sensing rather robustly in spite of sensing uncertainty.

## 2.2. Topological CFs

DEFINITION 2. A topological CF between two contacting polyhedra is defined as the set of PCs formed.

Based on the types of PCs in a CF, which characterize the extent of contact constraints, we can classify CFs into different types.

DEFINITION 3. Two CFs are of the same type if they consist of the same number of the same types of PCs.

For example, given objects  $A$  and  $B$ , if  $CF_1$  consists of one f-f PC and one e-f PC between  $A$  and  $B$ , and  $CF_2$  consists of a different f-f PC and a different e-f PC between  $A$  and  $B$ , then the two CFs are different CFs but they are of the same type.

Because a PC between two objects  $A$  and  $B$  describes certain contact characteristics satisfied by one or more relative contact configurations of  $A$  to  $B$ , which we simply call contact configurations, we can define the geometrical representation of a PC as the following:

DEFINITION 4. The geometrical representation of a PC,  $Geo_{PC}$ , denotes the set of contact configurations that satisfy the contact condition described by the PC's topological definition.

Subsequently, the geometrical representation of a CF can be defined as the following:

DEFINITION 5. The geometrical representation of a CF,  $Geo_{CF}$ , denotes the set of contact configurations that satisfy every contact condition represented by every PC in the CF.

From the above definitions, we can obtain the following corollary.

COROLLARY 1.  $Geo_{CF}$  is the intersection of the geometrical representations of the PCs in the topological CF.

It is obvious that the geometrical representations of all the CFs between two objects partition the contact configuration space (of one object with respect to the other).

### 2.3. Contact States and Connectivity

Based on the proof by Hopcroft and Wilfong (1986), if there is a way to move two objects from one contact configuration to another, then there is a way to do so with the objects remaining in contact throughout the motion. With this in mind, we now discuss the connectivity of CFs.

Within a CF, the connectivity question is whether the  $Geo_{CF}$  forms a single connected region or multiple connected regions of configurations. In many cases,  $Geo_{CF}$  is a single connected region of configurations, and thus the CF uniquely determines a contact state. However, there are also many cases in which  $Geo_{CF}$  consists of multiple connected regions of configurations. That is, from a contact configuration in one connected region of  $Geo_{CF}$ , there is no path consisting of only configurations satisfying the CF, or CF-compliant path, leading to a contact configuration in another connected region of  $Geo_{CF}$ . In such a case, each connected region of  $Geo_{CF}$ , called a CF-connected region, is considered a separate contact state. More formally:

**DEFINITION 6.** A CF between two objects,  $CF$ , and a contact configuration  $C$  in a  $CF$ -connected region (in  $Geo_{CF}$ ), denoted as a pair  $\langle CF, C \rangle$ , characterize the  $CF$ -connected region and define a contact state between the objects.

With the above definition,  $\langle CF_k, C_i \rangle$  and  $\langle CF_k, C_j \rangle$  denote two different  $CF_k$ -connected regions.

We now consider connectivity between contact states of different CFs,  $\langle CF_i, C_i \rangle$  and  $\langle CF_j, C_j \rangle$ .

**DEFINITION 7.** If from a contact configuration in  $\langle CF_i, C_i \rangle$  to a contact configuration in  $\langle CF_j, C_j \rangle$  there exists a path consisting of only a segment of contact configurations in  $\langle CF_i, C_i \rangle$  succeeded by a segment of contact configurations in  $\langle CF_j, C_j \rangle$ , then  $\langle CF_i, C_i \rangle$  and  $\langle CF_j, C_j \rangle$  are generally defined neighboring contact states and  $CF_i$  and  $CF_j$  are generally defined neighboring CFs.

Because CFs characterize discrete contact states topologically, we can map the above configuration-based definition of neighboring CFs to a topological description in terms of how PCs (or the topological surface elements of the PCs) are related. We first define a containment relation between PCs.

**DEFINITION 8.** Let  $PC_i = (a^A - b^B)$  and  $PC_j = (c^A - d^B)$  be two PCs between two polyhedra  $A$  and  $B$ .  $PC_i$  contains  $PC_j$  if and only if one of the following cases holds:

1.  $c^A$  is on the boundary of  $a^A$ , and  $d^B$  is on the boundary of  $b^B$ .
2.  $c^A$  is on the boundary of  $a^A$ , and  $d^B$  is  $b^B$ .
3.  $c^A$  is  $a^A$ , and  $d^B$  is on the boundary of  $b^B$ .

With the above definition, we can define a containment relation between CFs.

**DEFINITION 9.** For two CFs,  $CF_i$  and  $CF_j$ , such that  $CF_i \neq CF_j$ ,  $CF_i$  contains  $CF_j$  if and only if

- $card(CF_i) \geq card(CF_j)$ , where  $card(*)$  returns the cardinality.<sup>1</sup>
- For every PC in  $CF_j$ , either it also belongs to  $CF_i$  or it is contained by a unique PC in  $CF_i$ , and no two PCs in  $CF_j$  are contained by the same PC in  $CF_i$ .

Clearly, the following corollary holds.

**COROLLARY 2.** For two CFs,  $CF_i$  and  $CF_j$ , if  $CF_j \subset CF_i$ , then  $CF_i$  contains  $CF_j$ .

Now, we can describe neighboring relations between PCs and between CFs in terms of the corresponding containment relations.

**THEOREM 1.** If a single-PC CF  $\{PC_i\}$  is a generally defined neighboring CF of another single-PC CF  $\{PC_j\}$ , then one of the PCs must contain the other. If  $PC_i$  contains  $PC_j$ , then  $PC_j$  is called a less-constrained neighbor (LCN) of  $PC_i$ , and  $PC_i$  is called a more-constrained neighbor of  $PC_j$ .

**Proof.** Let  $A$  and  $B$  be the objects in contact. Assume that there is no containment relation between  $\{PC_i\}$  and  $\{PC_j\}$ , which are two single-PC CFs between  $A$  and  $B$ . Then, the two contacting surface elements of either object ( $A$  or  $B$ ) with respect to the two PCs are neither the same nor adjacent. This implies that to change one surface element to the other, the relative contact motion will have to result in a different contacting surface element, which means a different PC, en route; that is,  $\{PC_i\}$  and  $\{PC_j\}$  are not neighbors. The theorem is proven by refutation.

Note that a PC of the v-v type (Fig. 1) (which is degenerate) does not contain any other PC, and thus it is a least constrained PC.

**THEOREM 2.** If two CFs,  $CF_i$  and  $CF_j$  ( $CF_i \neq CF_j$ ), are generally defined neighboring CFs, then one of them must contain the other. If  $CF_i$  contains  $CF_j$ , then  $CF_j$  is called an LCN of  $CF_i$ , and  $CF_i$  is called a more-constrained neighbor of  $CF_j$ .

**Proof.** The proof is similar to the proof of Theorem 1, based on the general definitions of neighboring CFs and that of the CF containment.

Note that if a CF consists of a single v-v type of PC, then it does not contain any other CF and is a least constrained CF. Figure 3 shows examples of neighboring PCs and neighboring CFs.

For better stratification, we further narrow our definition of neighboring CFs by limiting the containment relations they can have.

1. Based on Definition 2, the cardinality here gives the number of PCs in a CF.



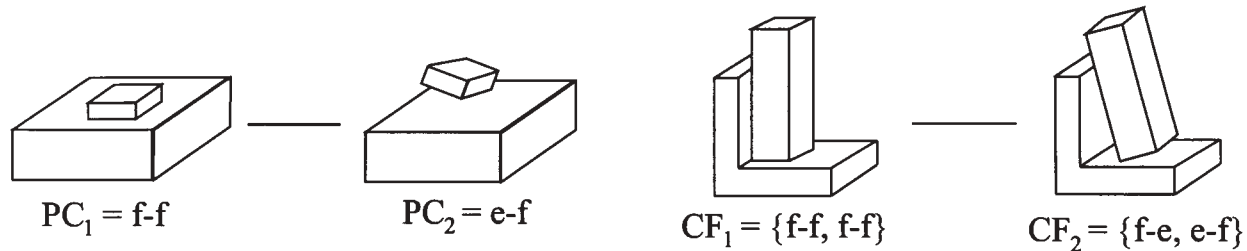


Fig. 3. Examples of neighboring principal contacts (PCs) and contact formations (CFs).

DEFINITION 10.  $CF_i$  is a specially defined neighboring CF of  $CF_j$  if and only if there is a containment relation between  $CF_i$  and  $CF_j$ , and every containment between two PCs of  $CF_i$  and  $CF_j$  satisfies either condition 2 or condition 3 of Definition 8 but not condition 1. Consequently, two generally defined neighboring contact states  $\langle CF_i, C_s \rangle$  and  $\langle CF_j, C_t \rangle$  are now also specially defined neighboring contact states and are simply called neighboring contact states.

Based on the proof by Hopcroft and Wilfong (1986) (explained at the beginning of this subsection) and the above definitions, we can now define the entire contact state space.

DEFINITION 11. The contact state space between two polyhedra is a simple, connected graph  $\mathcal{G}$  that consists of all the distinct contact states as nodes and arcs connecting every pair of neighboring contact states.

### 3. Generation of Contact State Graphs

Given two (or more) objects in contact, to construct the contact state graph  $\mathcal{G}$  automatically requires the handling of two issues: (1) how to generate valid CFs, or how to tell whether a set of PCs forms a geometrically valid CF, and (2) how to find CF-connected regions (i.e., contact states) and the neighboring relations between them. Both are difficult issues in general<sup>2</sup> if considered in isolation. Our work exploits their connections and handles the two issues simultaneously, as detailed in the following subsections.

#### 3.1. Divide-and-Merge Approach

Our approach is to divide  $\mathcal{G}$  into certain subgraphs, automatically generate the subgraphs, and merge the results.

The kind of subgraph we generate consists of a contact state of a seed CF,  $CF_g$ , and those of all the less-constrained CFs, which we call a GCR graph of  $CF_g$  (Xiao 1997). Starting from such a goal contact state,  $\langle CF_g, C_g \rangle$ , the GCR graph can be grown by repeatedly “relaxing” contact constraints to obtain contact states of LCN CFs. The process terminates

2. Only in the case of two convex objects are the issues trivial: every valid CF consists of a single PC, and every possible PC is a valid CF. In addition,  $Geo_{CF}$  for each CF is a single CF-connected region.

when there is no new contact state to be added to the graph. Clearly, such a process will always terminate. As for the goal CFs, they are the locally most constrained CFs, many of which indicate goal or intermediate goal CFs of an assembly. Given the goal contact states,  $\mathcal{G}$  can be obtained either partially or completely by merging the corresponding GCR graphs.

We prefer to form  $\mathcal{G}$  from GCR graphs because a GCR graph is much easier to generate automatically than an arbitrary subgraph of  $\mathcal{G}$ , taking advantage of the following two facts:

1. All CFs in the GCR graph of  $CF_g$  can be hypothesized topologically from the topological expression of  $CF_g$ .
2. The transition motion from a CF to an LCN is often simpler than that to a more-constrained neighboring CF, and in many cases the motion can be infinitesimal (see Section 3.3).

Our algorithm for constructing a GCR graph is outlined below.

#### Algorithm GCR-Gen:

Add  $\langle CF_g, C_g \rangle$  to an empty FIFO queue *open*;

WHILE *open* is not empty DO

BEGIN

- $\langle CF_c, C_c \rangle \Leftarrow$  contact state removed from *open*;
- hypothesize LCN CFs of  $CF_c$  (Section 3.2);
- find feasible LCN contact states to  $\langle CF_c, C_c \rangle$  from hypothesized LCN CFs (Sections 3.3 and 3.4);
- link each feasible LCN contact state found to  $\langle CF_c, C_c \rangle$ , and if it is already in the (partially generated) GCR (i.e., it was generated previously), merge the two copies of the same contact state to one, else, add the new LCN contact state to *open*.

END

Figure 4 shows the process to construct a GCR graph. From the seed node, the construction proceeds from top to bottom, connecting feasible nodes and discarding infeasible ones.

As an example, Figure 5 shows a GCR graph of a two-dimensional peg-in-hole assembly where the contact state with an asterisk is the seed node. Note that the clearance between the peg and the hole was greatly exaggerated so that different contact states can be seen clearly.

Once the GCR graphs are generated, we use a simple algorithm to automatically merge GCR graphs into a single contact state graph. In the following subsections, we provide step-by-step descriptions of the divide-and-merge approach.

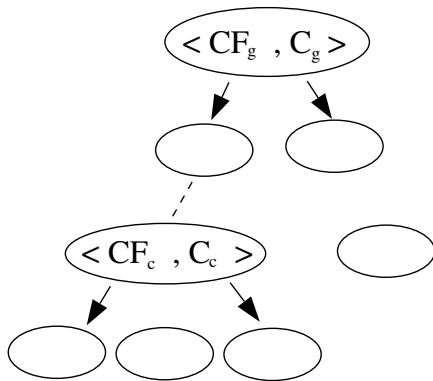


Fig. 4. Construction of a goal-contact relaxation graph.

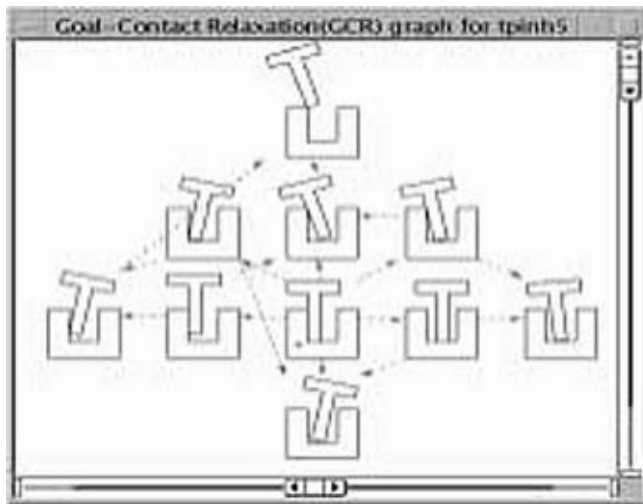


Fig. 5. A goal-contact relaxation graph of a two-dimensional peg-in-hole assembly where the node with an asterisk is the seed node.

### 3.2. Hypothesizing LCNs

Given a valid CF =  $\{PC_i\}_{i=1}^n$ , where  $n \geq 1$ , its possible LCN CFs can be hypothesized by applying one of the following actions to each  $PC_i$  of the CF according to Definitions 8-10 (Section 2.3):

- remove**  $PC_i$
- change**  $PC_i$  to a PC it contains (i.e., an LCN PC)
- keep**  $PC_i$ ,

provided that **remove** or **keep** is not applied to all PCs in the CF simultaneously to result in an empty set or the CF itself. This implies that for a single-PC CF, only **change** can be applied. Let  $\mathcal{A} = \{\text{remove, change, keep}\}$ . A set of actions that can be applied to an  $n$ -PC CF to hypothesize LCNs of the CF can be expressed as  $\alpha_n = \{x_i | x_i \in \mathcal{A}, i = 1, \dots, n, \exists x_i \neq \text{keep}, \exists x_i \neq \text{remove}\}$ . It can be shown that the number of such  $\alpha_n$ 's is

$$R(n) = \frac{(n+1)(n+2)}{2} - 2.$$

To be efficient, our strategy hypothesizes only LCNs that are topologically feasible for a CF, considering the interdependence of contact constraints between PCs. For example, given two polyhedra  $A$  and  $B$  in a CF of two face-face PCs,  $CF_c = \{f_1^A - f_1^B, f_2^A - f_2^B\}$ , it is impossible, regardless of specific object and contact geometry, to **change** one face-face PC and **keep** another face-face PC. Therefore, the action set  $\{\text{change, keep}\}$  is never applied to a CF of the type consisting of two face-face PCs because it will not generate topologically feasible results. As another example, if  $(e_1^A - f_2^B)$  is a valid edge-face PC between  $A$  and  $B$  and  $v_1^A$  is a nonconvex vertex of  $e_1^A$ , then  $(v_1^A - f_2^B)$  is not a topologically feasible LCN PC. In general, for each  $\alpha_n$ , we use rules to specify the types of  $n$ -PC CFs that can be applied by  $\alpha_n$  to produce topologically feasible LCN hypotheses. We have implemented such rules for  $n = 2, 3$  (Section 4.1).

Once a topologically feasible LCN is hypothesized, the next task is to check whether it represents a geometrically valid CF. In the next subsection, we discuss how to determine feasible contact states from such hypothesized LCN CFs.

### 3.3. Neighboring Relaxation and Feasibility Check

In general, we call the process of changing a CF to one of its neighboring CFs without encountering other CFs a neighboring transformation. Such a process is accomplished by a contact motion changing one contact state  $\langle CF_p, C_p \rangle$  to another, which is called a neighboring transformation motion. A neighboring transformation motion usually consists of two parts:

- *Reconfiguration*, which changes contact configurations within the same CF;

- *Transition*, which changes a contact configuration in a CF to one in a neighboring CF and is an infinitesimal motion.

For the case in which the transformation is from a CF to one of its LCN CFs, a neighboring transformation motion is further called a neighboring relaxation motion.

Recall that in our approach, from a valid contact state  $\langle CF_P, C_P \rangle$ , an LCN CF of  $CF_P$ ,  $CF_Q$ , is first hypothesized. The task is then to check whether it can result in a geometrically valid neighboring contact state. This task is equivalent to checking whether there exists a feasible neighboring relaxation motion leading  $\langle CF_P, C_P \rangle$  to  $\langle CF_Q, C_Q \rangle$ , where  $C_Q$  is a valid contact configuration under  $CF_Q$ . This problem can be further divided into two subproblems:

1. Checking whether there exists a feasible finite reconfiguration motion from  $C_P$  to  $C'_P$  within the same  $CF_P$ ;
2. Checking whether there exists a feasible infinitesimal transition motion from  $\langle CF_P, C'_P \rangle$  to  $\langle CF_Q, C_Q \rangle$ .

Note that such a contact motion is feasible if there is no collision before the goal is achieved. Figure 6 shows a two-dimensional (polygonal) example of a feasible neighboring relaxation motion from  $\langle \{(e_1^A - e_2^B)\}, C_P \rangle$  to  $\langle \{(v_1^A - v_2^B)\}, C_Q \rangle$  via  $C'_P$ .

### 3.3.1. Reconfiguration

Subproblem 1 (of finding feasible reconfiguration motion) is generally a motion-planning problem in a configuration space of reduced degrees of freedom (DOFs) and reduced scope as the space is constrained by a known CF; thus, we call it a CF-compliant motion-planning problem. The cases with the highest DOFs occur when the CF simply consists of a single vertex-face PC and the CF-compliant motion has five DOFs. Usually, the more PCs a CF has, the fewer DOFs the CF-compliant motion has.<sup>3</sup> In addition to the general advantages of reduced DOF and reduced scope offered by a CF, this compensating nature (i.e., higher DOFs with simpler CF, and lower DOFs with more complex CF) reduces the complexity of motion planning. For our particular reconfiguration problem (i.e., subproblem 1), planning is further simplified because it is an any-path (as opposed to a best-path) problem and the path destination is often not restricted to a specific configuration.

On the other hand, CF-compliant motion planning poses a special challenge that collision-free motion planning does not: how to make sure that a path generated is CF compliant (i.e., consisting of only those configurations satisfying the constraints of the CF). To tackle the problem, we developed an effective and efficient strategy to generate CF-compliant configurations randomly (Ji and Xiao 2001a) and a

3. Except for cases with redundant PCs, such as those with collinear faces in contact.

method to produce CF-compliant interpolations between two CF-compliant configurations (Ji 2000; Ji and Xiao 2001b). These techniques enable us to apply a randomized planner, such as one based on the probabilistic road map approach (Kavraki et al. 1996; Kavraki and Latombe 1998), to generate CF-compliant paths free of other collisions (collisions other than the contact described by the CF). In our implementation, we also adapted the general collision-detection package RAPID (Lin and Manocha 1996) to detect only those other collisions additional to the desired CF (Ji 2000; Ji and Xiao 2001b). Figures 7 and 8 (Extension 7<sup>4</sup>) show two examples of planning results. In both figures, (a) shows the initial and the end configurations and (b) shows the planned motion sequence. Note that in both cases, there are obstacles that can cause collisions in addition to the CF.

### 3.3.2. Transition for Relaxation

In addition to a clear division of motion between a finite reconfiguration and an infinitesimal transition, neighboring relaxation has another important advantage over neighboring transformation to a more-constrained CF. In many cases, especially when a CF involves multiple PCs with nonparallel contact planes, there is no need for finite reconfiguration; that is, if the LCN is feasible, there is a neighboring relaxation motion consisting of only an infinitesimal transition. Figure 9 shows some examples in which all the neighboring relaxations require only infinitesimal transition motions.

Subproblem 2 (i.e., to find a feasible infinitesimal transition motion) is generally an easier problem. Because the transition is toward an LCN CF, the infinitesimal transition motion partially breaks the current contact to change it to a less-constrained one. There are three types of infinitesimal transition motions, as shown in Figure 9: infinitesimal translation (IT), infinitesimal rotation (IR), and infinitesimal combined translation and rotation (IC).

Note that an infinitesimal motion is characterized by the axis of the motion and the direction of velocity. Such a motion can be hypothesized in topological terms of the contact elements involved in the change. Figure 10 shows some examples. In Figure 10a, an IT motion can be partially specified as parallel to the contact plane  $CP_2$  and in a direction  $v$  pointing away from the contact plane  $CP_1$ . In Figure 10b, the IR motion can be completely specified as about the edge of  $A$  in the edge-face PC to maintain the PC, and the direction of motion (or velocity) is the one that will not cause penetration of  $A$  into  $B$ . In Figure 10c, the IC motion can be specified as a combination of IT motion of Figure 10a and IR motion of Figure 10b. Note that the above candidates of infinitesimal motions are general for any CFs consisting of an edge-face PC and a face-face PC with nonparallel contact planes. We use a number of such general and concise rules according to

4. Please see the Index to Multimedia Extensions at the end of this article.



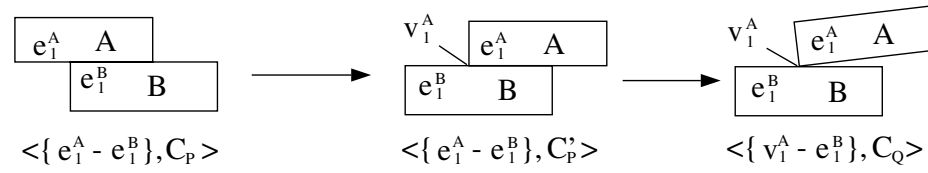
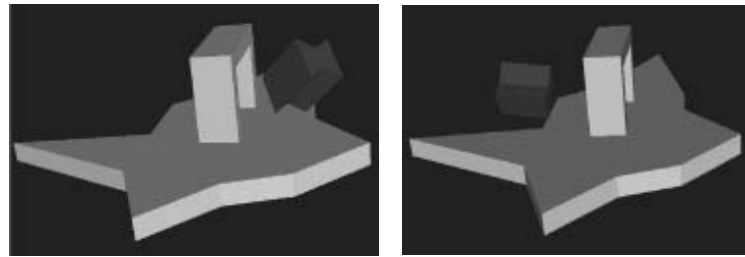
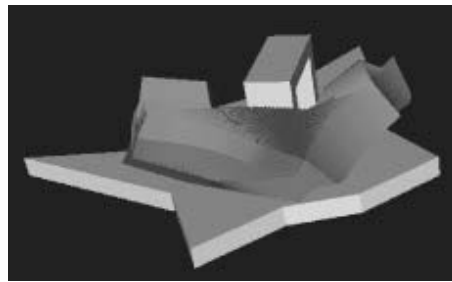


Fig. 6. Neighboring relaxation motions.

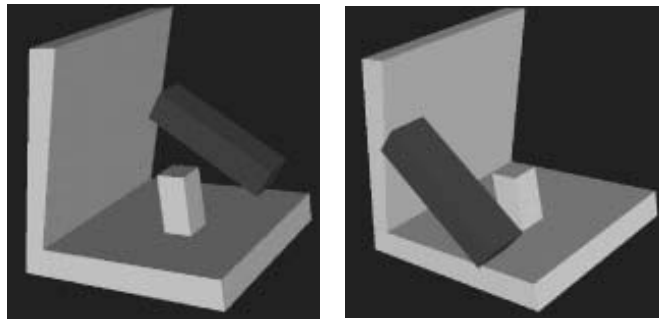


(a)

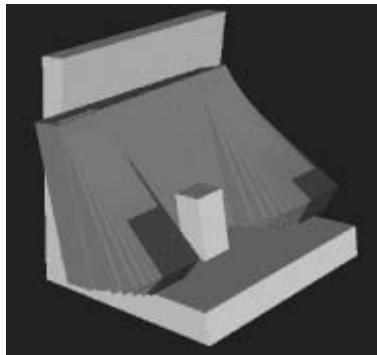


(b)

Fig. 7. Reconfiguration motion in a contact formation consisting of one vertex-face principal contact.



(a)



(b)

Fig. 8. Reconfiguration motion in a contact formation consisting of two edge-face principal contacts (see also [Extension 7](#)).

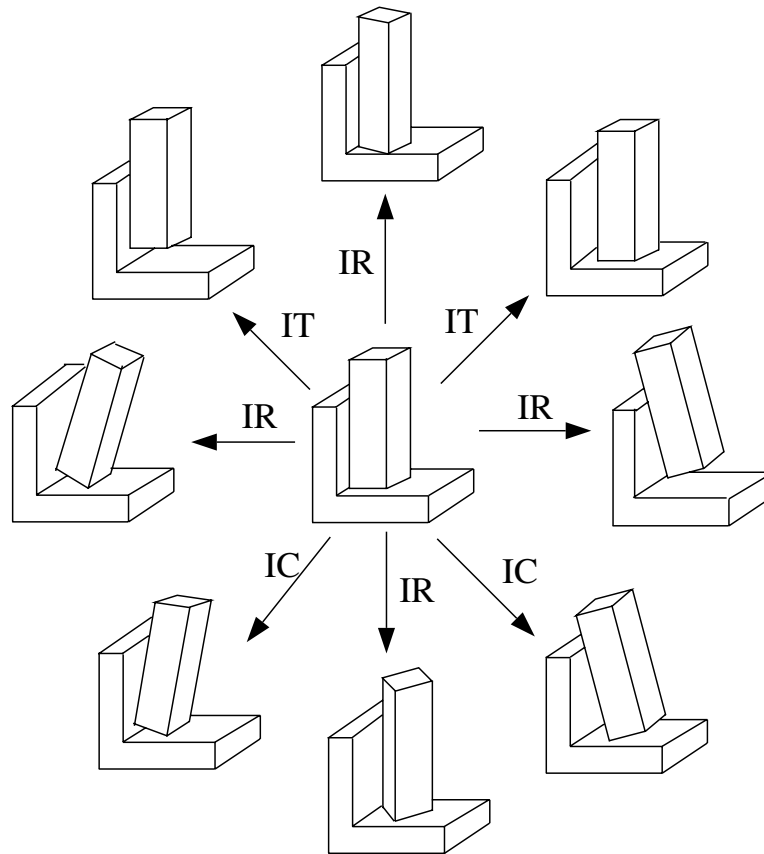


Fig. 9. Examples in which only infinitesimal transition motions are needed. IT = infinitesimal translation, IR = infinitesimal rotation, IC = infinitesimal combined translation and rotation.

different CF types to determine candidates of infinitesimal transition motions.

Whether a topologically determined candidate infinitesimal motion is indeed geometrically feasible depends on the specific geometries of the objects. Figure 11 shows an example in which the topological information is the same for all three cases but the feasibilities of the transition motions are different for different geometries. Thus, there is a need to check the feasibility of an intended infinitesimal transition motion.

Let  $A$  and  $B$  be two polyhedra in contact, and suppose the neighboring relaxation motion is intended on  $A$ . Let  $\mathbf{n}$  denote the normal of a contact plane pointing toward the static object  $B$ . Given the type, axis, and direction of an infinitesimal motion, we use the following algorithms to check its feasibility.

**IT.** It is feasible if the direction of translation, in terms of the linear velocity vector  $\mathbf{v}$ , does not penetrate through all the contact planes (and/or contact lines) into  $B$ :  $\mathbf{v} \cdot \mathbf{n} \leq 0$  means no penetration through the contact plane with normal  $\mathbf{n}$ .

**IR.** The specified rotation is feasible if the tangent velocity vector  $\mathbf{v}$  at each contacting vertex (of either object) does not

penetrate through the corresponding contact plane with normal  $\mathbf{n}$ ; that is,  $\mathbf{v} \cdot \mathbf{n} \leq 0$ .<sup>5</sup>

**IC.** An IC motion can be implemented as being equivalent to either (i) an IR followed by a guarded straight-line translation (GT) or (ii) an IT followed by a guarded rotation (GR) (see Fig. 12). Note that a guarded motion is terminated by a collision/contact. It is feasible if either (i) or (ii) is feasible. The amount of GT or GR is determined by the  $\delta$  amount of the preceding IR or IT, respectively.<sup>6</sup>

The algorithms were implemented in the form of a predicate function,

$$\text{feasible}(\text{type}, \text{axis}, \text{dir}, \delta), \quad (1)$$

which returns “true” if the specified infinitesimal motion is feasible.

5. Note, however, that if the vertex belongs to  $B$ , then the IR can be feasible only when  $\mathbf{v} \cdot \mathbf{n} < 0$ .

6. Theoretically,  $\delta$  can be arbitrarily small, but it is finite in implementation.

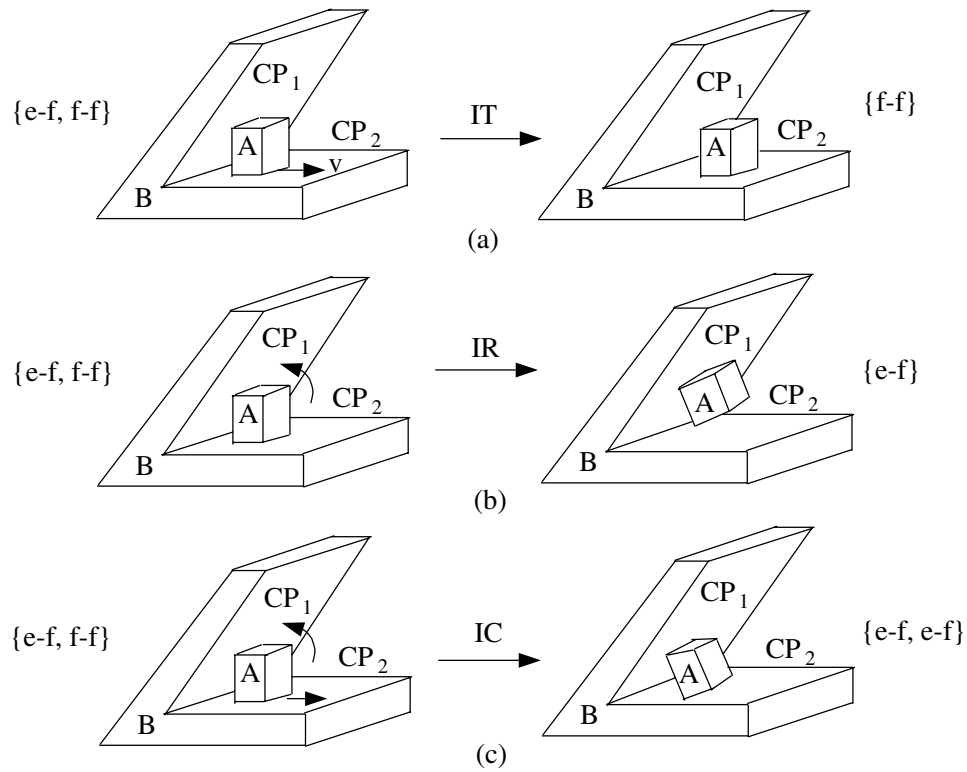


Fig. 10. Infinitesimal transition motions can be specified topologically. IT = infinitesimal translation, IR = infinitesimal rotation, IC = infinitesimal combined translation and rotation.

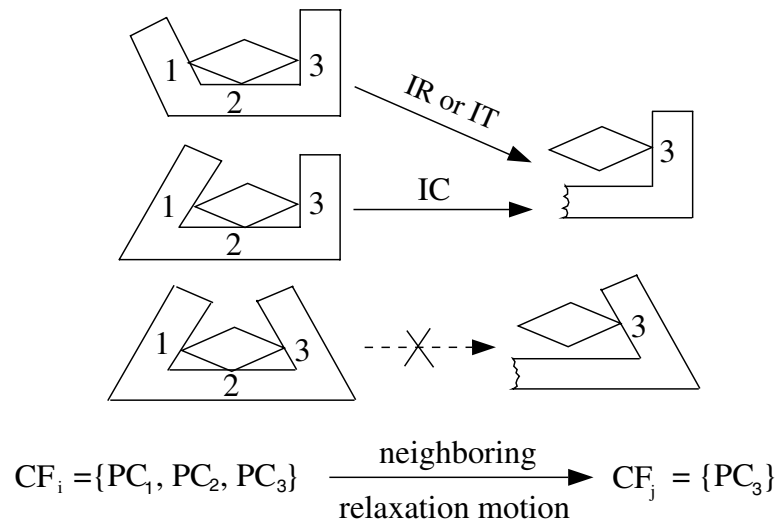


Fig. 11. The feasibility of a possible neighboring relaxation motion depends on specific contact geometry. IT = infinitesimal translation, IR = infinitesimal rotation, IC = infinitesimal combined translation and rotation.

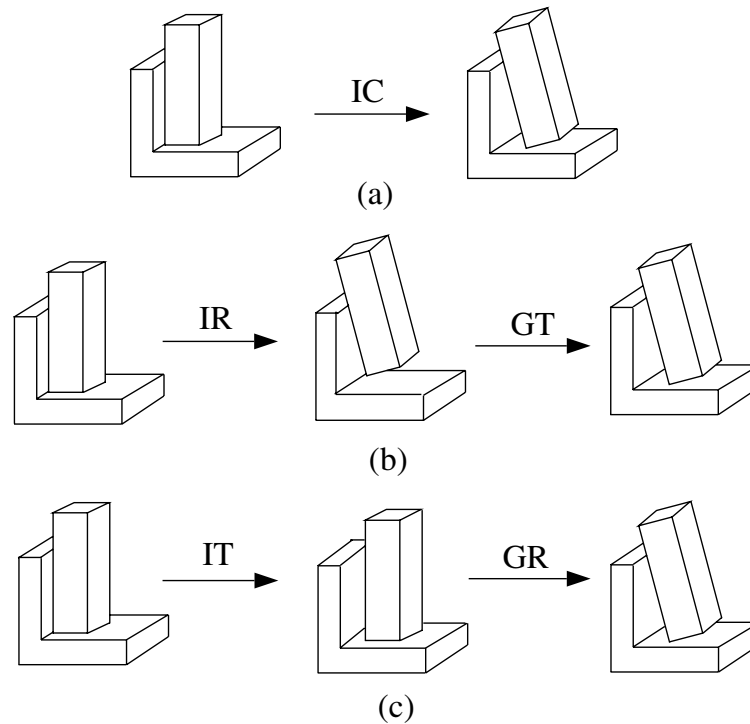


Fig. 12. Equivalent motions of infinitesimal combined translation and rotation (IC) type of motions. IT = infinitesimal translation, IR = infinitesimal rotation, GT = guarded straight-line translation, GR = guarded rotation.

### 3.4. Merging Nodes and GCR Graphs

Given any contacting objects, there are usually a number of CFs that cannot be obtained by relaxing some contact states of other CFs (i.e., they cannot be found in the GCR graphs of some other CFs). These CFs are the locally most-constrained CFs and, thus, the natural candidates of goal (or seed) CFs for generating GCR graphs. A larger contact state graph can be created by merging GCR graphs. The entire contact state graph  $\mathcal{G}$  is obtained if all GCR graphs are merged. There are two ways to combine multiple GCR graphs. One is to do it sequentially by growing a new GCR graph to meet an existing graph (which could consist of one or more GCR graphs) where there are shared states. Another is to generate each GCR graph independently and then merge the GCR graphs in a separate phase.

The key issue in automatic merge is to determine whether one node in one graph represents the same contact state as a node in the other graph so that the two nodes should be merged in the combined graph. Clearly, only the nodes sharing the same CF can possibly represent the same contact state. By dividing nodes in a contact state graph  $G_1$  (which can be either a GCR graph or the result of some merged GCR graphs

into levels based on their CF types [Definition 3]), searching a node with a given CF in  $G_1$  can be reduced to simply searching the nodes in the same level (i.e., of the same type of CFs as the given CF) in  $G_1$ . Once two nodes are found sharing the same CF, say,  $\langle CF_i, C_1 \rangle$  and  $\langle CF_i, C_2 \rangle$ , the problem is to determine whether they represent the same  $CF_i$ -connected region; in other words, whether there exists a  $CF_i$ -compliant path connecting  $C_1$  and  $C_2$ , as addressed in more detail in the following subsection.

If two nodes  $\langle CF_i, C_1 \rangle$  and  $\langle CF_i, C_2 \rangle$  can be merged, merge is done by linking all the (more constrained and less constrained) neighboring contact states of  $\langle CF_i, C_2 \rangle$  to  $\langle CF_i, C_1 \rangle$  and discarding  $\langle CF_i, C_2 \rangle$  (Fig. 13).

### 3.5. Handling CFs with Multiple Connected Regions

As described in Section 2.3, the geometric representation  $Geo_{CF}$  of a CF (Section 2.2) may consist of more than one CF-connected region, and each CF-connected region represents a separate contact state. One way to detect automatically whether a given CF has multiple CF-connected regions in its  $Geo_{CF}$  is through CF-compliant motion planning. One could use the same randomized planner for reconfiguration

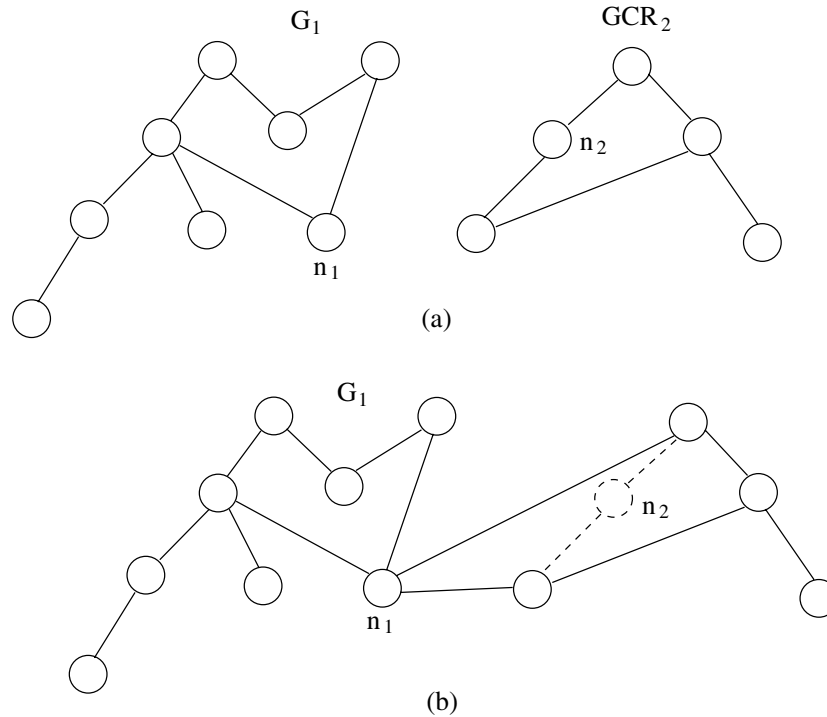


Fig. 13. Merging of goal-contact relaxation (GCR) graphs: (a) before merging, (b) after merging  $n_1$  and  $n_2$  and discarding  $n_2$ .

(Section 3.3.1) to explore the connectivities of CF-compliant configurations (i.e., to build a roadmap on  $Geo_{CF}$  exhaustive enough so that the connectivity of the roadmap reflects the connectivity of  $Geo_{CF}$ ). However, one can imagine that it will be quite expensive computationally if all CFs have to be treated this way.

Noticeably, in our divide-and-merge approach to generate contact states, such detection of connectivity is transformed into solving the following smaller and simpler problems for nonseed CFs:

- (1) During the generation of one GCR graph, from the current contact state  $\langle CF_c, C_c \rangle$ , whether a generated LCN CF should be split into two or more LCN states of  $\langle CF_c, C_c \rangle$  (Fig. 14);
- (2) Determine whether a contact state  $\langle CF_i, C_1 \rangle$  should be merged with another existing contact state  $\langle CF_i, C_2 \rangle$  of the same CF.

Fortunately, only for certain degenerate  $CF_c$ 's is the state-splitting scenario of (1) possible. First, the two contacting objects, say  $A$  and  $B$ , have to satisfy that at least one dimension of  $A$  (i.e., distance between two surface elements) is exactly the same as certain dimensions of  $B$ . Next, if we do not consider any CF involving degenerate PCs (Section 2.1), since these PCs have zero probability of occurrence in practice, then

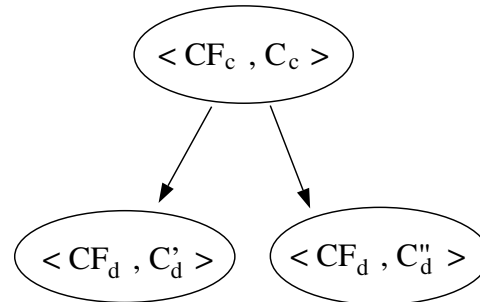


Fig. 14. State-splitting scenario.

the set of necessary conditions for the state-splitting scenario is as follows:

- $CF_c$  is not a single-PC CF;
- No infinitesimal translation can break any PC in  $CF_c$ ;
- In  $CF_c$ ,  $A$  is “squeezed” by  $B$  where certain dimensions between the squeezing elements of  $B$  are the same as certain dimensions between the elements of  $A$  in contact (with  $B$ ).

For an  $n$ -PC  $CF_c$ , where  $n > 2$ , the contact planes of the PCs form a prismatic tunnel with parallel side edges. For



a two-PC  $CF_c$ , the above necessary conditions manifest to a pair of PCs whose contact planes are parallel with opposite normals and the distance between the two contact planes equals the dimension of  $A$  between the two elements of  $A$  in contact (with  $B$ ). Figure 15 shows two two-dimensional (i.e., polygon) examples with state-splitting scenarios. Note that if we add a depth to the objects in the examples of Figure 15 to make them three-dimensional, then state splitting may not happen. Because state splitting is rare and easily identifiable, the involved states can be input to the program for generating a GCR graph as exceptions. In all other cases, for each LCN CF, only one contact state, produced by one feasible relaxation motion, will be linked to the parent contact state, and there is no need to explore the LCN connectivity here because further checking will be handled when solving (2).

Problem (2) can be encountered both in the generation of one GCR graph and in the merging of GCR graphs. In this case, the detection of connectivity is reduced to checking whether there exists a CF-compliant path to connect one known configuration  $C_1$  to another known configuration  $C_2$ . If a path exists, then the query can often be answered quickly before the connectivity of the entire  $Geo_{CF}$  is explored rather exhaustively. Figure 16 shows a two-dimensional example in which the two nodes with the same CF represent two CF-connected regions and cannot be merged.

We have also begun to study (Johnston and Xiao 2000) how the topological and geometrical characteristics of contacting objects relate to disconnectedness of a CF in order to quickly determine whether a CF has disjoint contact states without the need of searching for CF-compliant motion and dealing with the completeness problem of the search when such motion does not exist. This is an interesting topic in itself but is beyond the scope of this paper.

## 4. Implementation and Results

In implementation, we limited our current attention to contact state graphs with CFs of  $\leq 3$  PCs between arbitrary polyhedra  $A$  and  $B$  such that even the most constrained CF has no more than three PCs. This is often sufficient practically because a three-PC CF of many types, such as one with three f-f PCs, three f-e PCs, or two f-f PCs and one f-v PC, and so on, can be fully constrained (i.e., with zero DOFs) with nonparallel contacting features, since the PCs provide six independent constraint equations. Although theoretically if all PCs are of f-v or e-e-cross types, at most six PCs are needed to fully constrain a CF; in many cases, a CF with more than three PCs often involves redundant PCs and can be found to be equivalent to a CF with fewer PCs (Xiao and Zhang 1997).

If the seed CF of a GCR graph consists of  $\leq 3$  PCs, then the total number of CFs in the GCR graph cannot exceed  $(2 + u)^3$ , where  $u$  is the maximum number of edges and vertices bounding a face involved in the seed CF. The actual

number of feasible CFs is often much smaller because of geometric constraints, as evident from the example results to be presented.

### 4.1. Implemented Algorithms

In our implementation of the GCR generation algorithm **GCR-Gen** (Section 3.1), we used rules to implement the steps of finding LCNs for a given CF, which can be found in Appendix B. For each  $\alpha_n$  combination of actions of types **remove**, **change**, and **keep** (see Section 3.2), a rule describes the possible LCNs that are topologically feasible for the type of CF and the related possible neighboring relaxation motions from the CF to each LCN candidate. It decides whether there is a need for reconfiguration and calls the corresponding motion planner if this is so. It also calls the predicate function *feasible* (Section 3.3.2) to determine the feasibility of the transition motions. Finally, it returns the feasible LCN contact states, each of which is associated with a representative contact configuration. Note that a single-PC LCN contact state cannot be of the degenerate type e-e-touch (Fig. 1)—if there are only two edges in contact, **GCR-Gen** classifies such a contact state as that of a single e-e-cross PC.

Most operations in **GCR-Gen** take constant time or are bounded by constant time except for those requiring CF-compliant motion planning for reconfiguration. We designed and implemented a CF-compliant motion planner separately, as briefly introduced in Section 3.3.1 and detailed in Ji (2000) and Ji and Xiao (2001).

We also implemented a much simpler straight-line reconfiguration planner for cases in which reconfiguration can be achieved by CF-compliant straight-line translations so that there is no need for **GCR-Gen** to call a full-fledged CF-compliant motion planner. There are many such cases of objects  $A$  and  $B$ , including, for example, all the cases given in Figure 17. The time complexity of the straight-line planner is that of collision detection (between  $A$  and  $B$ ) along a given direction. **Extension 5** gives the code of **GCR-gen** using such a straight-line reconfiguration planner. For a sample input file, see **Extension 1**. The generated GCR graph is output into a binary .gcr file (see **Extension 2**), where two display files of the corresponding contact states obtained from the .gcr file are also provided. **Extension 3** displays the output GCR graph.

For the algorithm of merging GCR graphs, our current program **GCR-Merge** does not call CF-compliant motion planning to check connectivity (Section 3.5). Thus, the current code (**Extension 5**) simply merges all nodes sharing the same CF into one node by requiring that the  $Geo_{CF}$  of every CF consist of a single CF-connected region, which, however, is easily satisfied in many cases. **Extension 4** displays an example result of merge.

We have also developed, with David Johnston of our group being the primary designer and programmer, a display

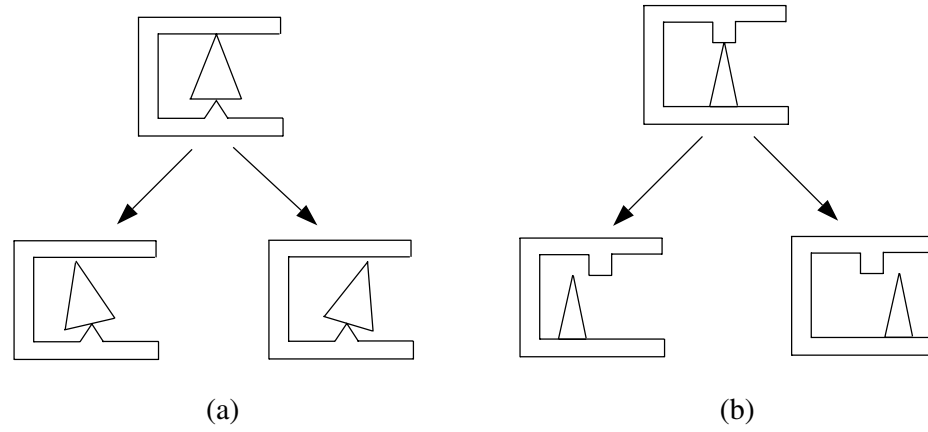


Fig. 15. Examples with state-splitting scenarios.

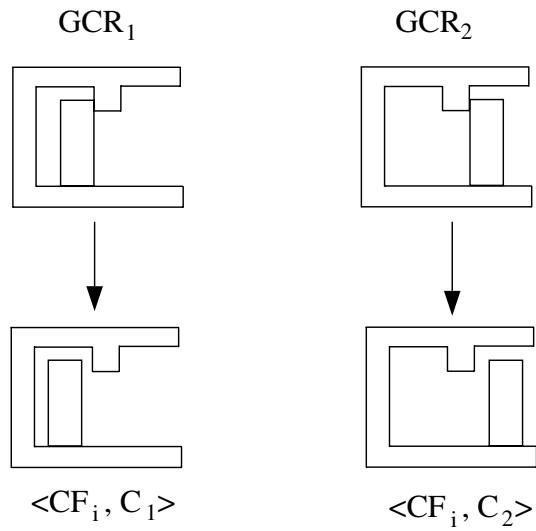


Fig. 16. Two nodes have the same contact formation but cannot be merged.










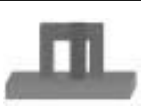














#PC	Seed CF	#nodes in GCR	time (s)	Seed CF	#nodes in GCR	time (s)	Seed CF	#nodes in GCR	time (s)
1PC		81	0.34		63	0.31		54	0.56
2PC		143	6.2		165	3.7		92	1.7
		116	5.3		138	3.2		76	3.3
		194	38.9		122	3.4		126	6.8
3PC		208	20.4		162	17.0		127	14.3
		124	5.6		89	4.6		65	3.5
		252	9.3		199	8.2		77	2.0
		170	14.0		252	5.5		131	27.4

Fig. 17. Several experimented examples. PC = principal contact, GCR = goal-contact relaxation, CF = contact formation.

program to view the contact state graph and the contact states between polyhedra. It runs on SUN/Sparc machines and Windows NT and allows user interaction to choose what to view and in what fashion, as demonstrated in [Extension 3](#). [Extension 6](#) provides the executable codes. The above algorithms are all implemented in C.

#### 4.2. Results

We now present and discuss some results obtained from running the implemented algorithms. Figure 18 shows a snapshot of a GCR graph between two three-dimensional polyhedra as viewed from running the interactive display program. The GCR graph contains 186 nodes. Each actual contact state can be displayed in a small window by clicking the corresponding node, as shown by the foreground images. Node 1 shows the seed contact state of the GCR graph, and other images show some of the LCN contact states of the seed node. Generation of the GCR graph takes 5 seconds.

We tested **GCR-Gen** on more than 30 different examples involving polyhedra pairs of vastly different shapes. Figure 17 summarizes the number of nodes and running times of some of the example GCR graphs generated, run on a SUN Ultra 10 workstation (which is rated at 12.1 SPECint95 and 12.9 SPECfp95).

Figure 19a shows an example of a small cube contacting the interior of a large S-shaped tunnel with square cross sections (where the cube is shown in solid and the S-tunnel is shown in wire frame).<sup>7</sup> In this case, there are two types of locally most-constrained CFs:

- Type 1, consisting of three face-face PCs between the cube and the S-tunnel, which can be formed by pushing the cube against a corner of the S-tunnel (as shown in Fig. 19a);
- Type 2, consisting of two face-face PCs between the cube and the S-tunnel, which can be formed by pushing the cube against two adjacent side walls of a straight segment of the tunnel.

Starting from eight such most-constrained CFs as seed nodes, we applied our program to generate eight GCR graphs and merged them automatically. The first row of Table 1 shows the results. From the resulted contact state graph, we further applied our CF-compliant random sampling strategy (Ji and Xiao 2001b) to generate 10 random configurations for each contact state. Figure 19b displays the accumulation of all such configurations.

Figure 20 shows an example with a small triangular wedge contacting a large star, where Figure 20a shows two contact states and Figure 20b shows a path of contact configurations

for contact motion between the two contact states. This example demonstrates the idea of decomposing contact motion planning into two levels introduced in Section 1. At the high level, we obtained a contact state graph automatically by running **GCR-gen** to generate GCR graphs from five seed contact states and by running the merge algorithm to merge the GCR graphs. Seed contact states are formed by fitting the wedge into each corner of the star, as shown in Figure 20a. The results are shown in the second row of Table 1. We then used a breadth-first search program to find a sequence of contact state transitions connecting the two states in Figure 20a in the contact state graph. The sequence found contains 13 nodes. Then, at the low level, we applied CF-compliant motion planning (Section 3.3.1) to each contact state in the sequence to generate a CF-compliant path of configurations and connected these CF-compliant path segments to form the final path in Figure 20b ([Extension 8](#)).

In all the examples presented, the time for generating a GCR graph is on the order of seconds, and merge usually takes even less time (as evident from Table 1).

## 5. Summary and Future Research

We have introduced a general approach for modeling and automatic construction of contact state space in terms of contact state graphs of the objects in contact. A contact state is characterized by a topological CF and a representative contact configuration under the CF, such that it represents a CF-connected region of contact configurations. A contact state graph is partially or completely formed by merging GCR graphs. Each GCR graph originates from a seed contact state, which is from a locally most-constrained CF.

With the currently implemented algorithms **GCR-Gen** and **GCR-Merge**, the entire contact state graph  $\mathcal{G}$  between two convex polyhedra can be generated completely and fully automatically. This is because the seed CFs of all the GCR graphs are simply all the f-f PCs, which can be easily enumerated automatically and input to **GCR-Gen** to generate all the GCR graphs, and the results can be input to **GCR-Merge** to obtain  $\mathcal{G}$ .

For nonconvex polyhedra, the current implementation can generate complete GCR graphs and merge them to obtain a larger graph or even the entire contact state graph  $\mathcal{G}$  if (1) the seed contact states are given to **GCR-Gen**, (2) each seed CF consists of  $\leq 3$  PCs, and (3) the  $Geo_{CF}$  of every CF consists of a single CF-connected region.

How to relax the above restrictions of the current implementation is naturally the concern of some further research topics. One topic is how to obtain the locally most-constrained CFs (i.e., the seed CFs for GCR graphs), semiautomatically or automatically, from the geometric descriptions of the contacting objects. This is related to the issue of characterizing

7. This example is courtesy of Nancy Amato of Texas A&M University.

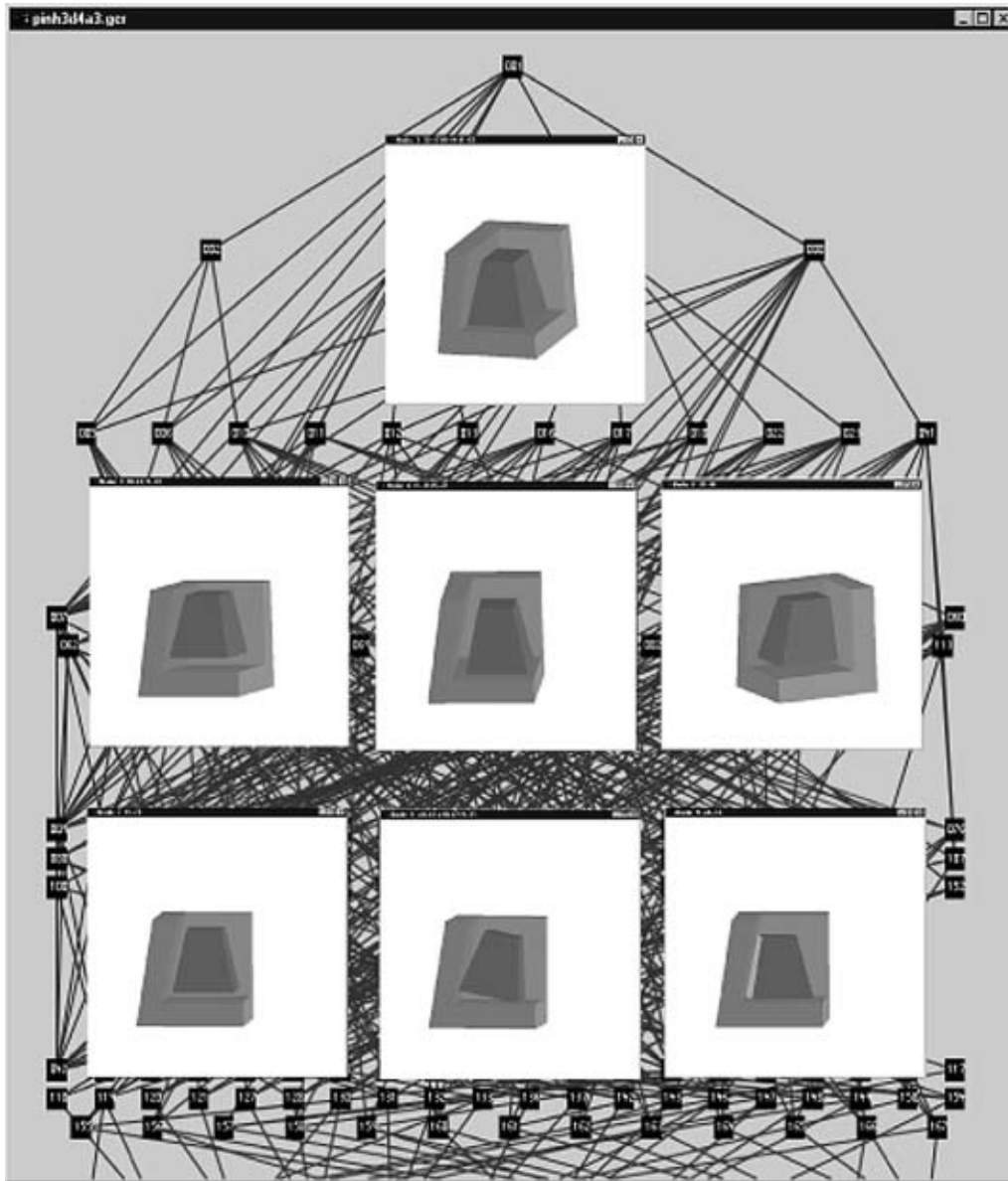


Fig. 18. Snapshot of a goal-contact relaxation graph between two polyhedra.

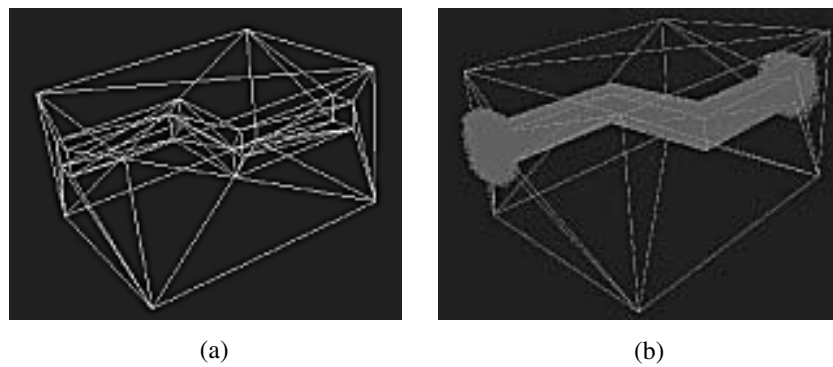


Fig. 19. S-tunnel example: (a) a small cube contacting an S-shaped tunnel, (b) the accumulation result of 8250 random contact configurations, with 10 for each automatically generated contact state.



**Table 1. Results of the S-Tunnel and Star Examples**

	No. of GCRs	No. Nodes per GCR	No. of Nodes after Merge	$t_{GCR}$ (s) <sup>a</sup>	$t_{merge}$ (s) <sup>b</sup>
<b>S-Tunnel</b>	8	172 or 98	825	$\leq 28$	0.96
<b>Star</b>	5	127	590	$\leq 1.17$	0.31

NOTE: GCR = goal-contact relaxation.

a. Time to generate a GCR graph.

b. Time to merge the GCR graphs.

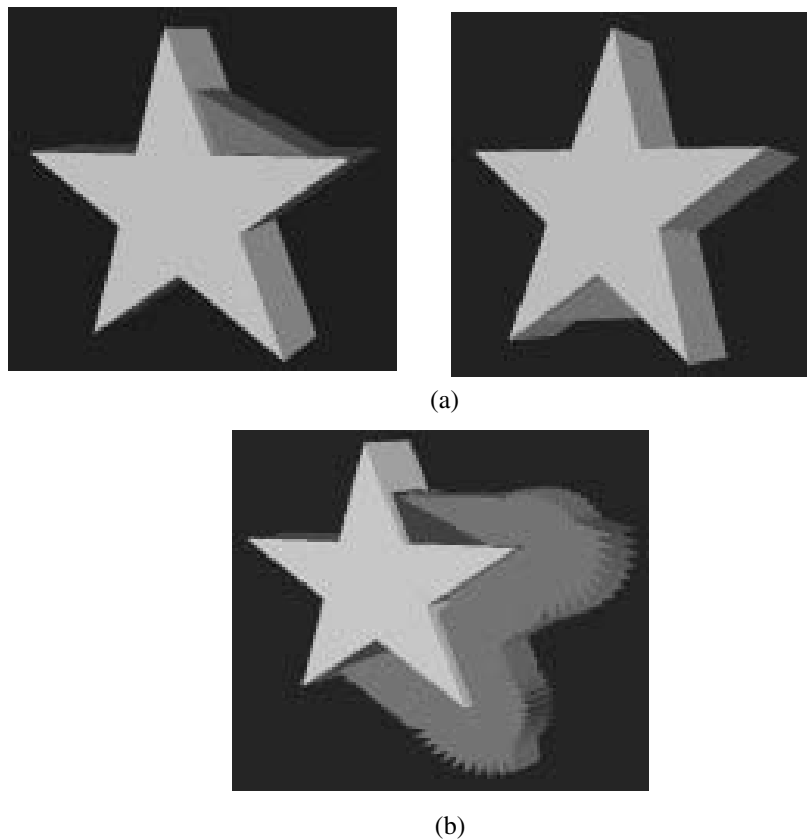


Fig. 20. Star example: (a) two seed contact configurations, (b) a path of contact configurations connecting the two seed contact configurations, which spans 13 contact states (see also [Extension 8](#)).

geometrical complexity of objects from a general perspective. For specific tasks, because only a certain portion of the (complete) contact state graph is useful, the selection of the relevant locally most-constrained CFs can take advantage of task-oriented knowledge. Moreover, the goals or intermediate goals of an assembly or a manipulation task are usually the locally most-constrained CFs. Further study is clearly needed.

Another issue is how to handle CFs of (arbitrarily) many PCs. Such CFs often include PCs that are not independent contributors of contact constraints and thus can be collapsed and represented by a fewer number of some equivalent PCs. The key task is to find an automatic way of finding equivalent PCs, which has been studied for two-dimensional polygons (Xiao and Zhang 1997) but not for three-dimensional polyhedra. Thus, it is another topic to be studied.

As mentioned in Section 3.5, we (Johnston and Xiao 2000) have begun to study how to decide whether the  $Geo_{CF}$  of a CF has more than one CF-connected region directly from the topological and geometrical characteristics of contacting objects so that exhaustive search for connectivity can be avoided. We have made considerable progress for two-dimensional objects but still need further research for three-dimensional objects.

As introduced in Section 1, a high-level contact state graph is not only needed in many tasks, both in the real and the virtual world, but also essential for simplifying contact motion planning and for enabling stratification for compliant control. Our approach effectively relates and simplifies both the problem of automatic generation of contact state graphs and that of planning contact motion: as we have shown, to accomplish the former, planning within a CF or CF-compliant motion planning is often needed, whereas the accomplishment of the former reduces the complexity of the latter with the two-level problem decomposition (Section 1). It is clear that for both problems, motion planning is largely reduced to the smaller problem of CF-compliant motion planning, which is easier to handle. We think that this approach could be promising for extension into more general contact cases involving multibody objects or nonpolyhedral objects.

The work also seems to have the potential to facilitate general motion planning (Latombe 1991). For high-dimensional motion planning of collision-free paths, randomized approaches (such as the probabilistic road map approach) (Kavraki et al. 1996; Kavraki and Latombe 1998; Xiao et al. 1997) and those based on evolutionary computation (Hocaoglu and Sanderson 1998) are often most viable because they rely on random sampling in the C-space to avoid the formidable task of building the C-obstacles explicitly. Such approaches could use the information of contact states and representative contact configurations to characterize C-obstacles and increase the effectiveness and efficiency of sampling.

## Appendix A: Rules for Generating Less-Constrained Neighbor Contact States in GCR-Gen

Let  $A$  and  $B$  be two polyhedra. We use  $a-b$  to represent a principal contact (PC), where  $a$  and  $b$  are elements (vertices, edges, and faces) of  $A$  and  $B$ , respectively, and we use  $\{PC_i\}_{i=1}^n$  to represent a contact formation (CF) with  $n$  PCs.

If  $p$  is a boundary element of  $q$  (Section 2.1) and  $p$  is convex, then it is denoted by  $p \subset \partial q$ . For a configuration of  $A$ , if element  $a$  of  $A$  is on or intersects element  $b$  of  $B$  (i.e.,  $a \cap b \neq \emptyset$ ), then it is denoted by  $a \in b$ .

Recall that each nondegenerate PC uniquely determines a contact plane (Section 2.1), and we use  $CP_i$  to represent the contact plane of  $PC_i$  and use  $\vec{n}_{CP_i}$  to denote the normal of  $CP_i$ . For a line PC  $PC_i$ , we use  $CL_i$  to represent the contact line of the PC. Note that degenerate PCs are generated only if they form single-PC CFs. For two-PC or three-PC CFs, all the PCs in a CF are nondegenerate.

As described in Section 3.3, any less-constrained neighbor (LCN) contact state of a  $\langle CF_i, C_i \rangle$  can be obtained from  $C_i$  by one of the following motions: (1) an infinitesimal transition, which is either an infinitesimal rotation (IR), an infinitesimal translation (IT), or a infinitesimal combined motion (IC); (2) a finite reconfiguration motion (i.e., CF-compliant motion) followed by an infinitesimal transition motion. The motions of (2) are denoted by FM.

For an arbitrary motion in the above, the following predicate indicates whether it is feasible:

$$feasible(type\ axis, direc, \delta/\Delta),$$

where it returns 1 if the motion is feasible and 0 otherwise. The following parameters are passed to *feasible*:

- *type* can be IR, IT, IC, or FM;
- *axis* is the direction for translation, the axis for rotation, or empty for FM motion (which indicates a call to a reconfiguration planner [see Section 4.1]);
- *direc* denotes the direction (+ or -) of the axis (empty for FM motion);
- $\delta$  indicates a prespecified amount of infinitesimal motion, and  $\Delta$  indicates the straight-line distance required of an FM motion, which is calculated.

We now describe the rules for generating LCN contact states of a given contact state  $\langle CF, C \rangle$  below. Each rule hypothesizes topologically feasible LCN CFs based on the type of  $CF$  and passes to the function *feasible* the information of possible neighboring relaxation motions to each hypothesized LCN state. A true LCN state is found if *feasible* returns 1.

**For Single-PC CFs**

- SR( $\{a-b\}$ ): (change one PC)
  1.  $\{a'-b\}$  is an LCN iff  
( $a$  is a face or an edge and  $a' \subset \partial a$ ) AND  
(feasible( $FM, , , \Delta$ ) while  $a' \in b$ )
  2.  $\{a-b'\}$  is an LCN iff  
( $b$  is a face or an edge and  $b' \subset \partial b$ ) AND  
(feasible( $FM, , , \Delta$ ) while  $b' \in a$ )

**For Two-PC CFs**

1. PR1( $\{a-b, PC_2\}$ ): (keep one PC, remove one PC)  
IF  $\neg(CP_1 \parallel CP_2)$ , THEN
  - $\{a-b\}$  is an LCN iff  
feasible( $IT, \vec{d}_{CP_2}, +/-, \delta$ ),  
where,  $\vec{d}_{CP_2}$  refers to the projection of  $\vec{n}_{CP_2}$  on  $CP_1$
 ELSE,
  - $\{a-b\}$  is an LCN iff  
( $a-b$  is not an f-f with an edge  $e$  or vertex  $v$  involved) AND  
feasible( $IR, v/e, +/-, \delta$ )  
OR  
( $a-b$  is an f-f PC) AND  
feasible( $FM, , , \Delta$ ) while  $PC_1$  is maintained  
and  $PC_2$  is broken)
2. PR2( $\{a-b, PC_2\}$ ): (change one PC, keep one PC)  
IF ( $a-b$  is an f-f or f-e/e-f PC) AND  $\neg(CP_1 \parallel CP_2)$   
THEN
  - (a)  $\{a'-b, PC_2\}$  is an LCN iff  
( $a' \subset \partial a$ ) AND  
(feasible( $FM, , , \Delta$ ) while  $a' \in b$  and  $PC_2$  is  
maintained) AND  
( $a'$  is an edge and  $a' \perp CP_2$ , let  $d$  be  $a'$ ) OR  
( $a'$  is a vertex, let  $d$  be a line from  $a'$  and  
 $\perp CP_2$ , and  $a$  and  $d$  are not collinear)) AND  
feasible( $IR, d, +/-, \delta$ )  
OR  
(PC is not f-f with an edge  $e$  or vertex  $v$  involved)  
AND  
( $a' \subset \partial a$ ) AND feasible( $IC, e/v, +/-, \delta$ ))
  - (b)  $\{a-b', PC_2\}$  is an LCN iff  
similar to the above by exchanging  $a$  and  $b$ .
3. PR3( $\{a-b, PC_2\}$ ): (change one PC, remove one PC)  
IF  $\{a-b\}$  is an f-f or f-e/e-f PC, THEN

- (a)  $\{a'-b\}$  is an LCN iff  
( $a' \in \partial a$ ) AND (feasible( $FM, , , \Delta$ ) while  $a' \in b$   
and  $PC_2$  is maintained) AND  
(feasible( $IR, a', +/-, \delta$ ) while  $PC_2$  is broken)
  - (b)  $\{a-b'\}$  is an LCN iff  
similar to the above by exchanging  $a$  and  $b$ .
4. PR4( $\{a-b, c-d\}$ ): (change both PCs)  
IF both are f-f or f-e/e-f PCs, THEN
    - (a)  $\{a'-b, c'-d\}$  is an LCN iff  
( $a' \subset \partial a$  and  $a' \in b$ ) AND  
(feasible( $IR, a', +/-, \delta$ ) while  $c-d$  is changed  
to  $c'-d$ )  
OR  
( $\neg(CP_1 \parallel CP_2)$ ) AND ( $a' \subset \partial a$  and  $c' \subset \partial c$ )  
AND  
( $a' \in b, c' \in d$ ) AND feasible( $IC, a'/c', +/-, \delta$ ))
    - (b)  $\{a-b', c'-d'\}$  is an LCN iff  
similar to (a) by exchanging  $a$  with  $b$ , and  $c$  with  
 $d$ .
    - (c)  $\{a'-b, c-d'\}$  is an LCN iff  
similar to (a) by exchanging  $c$  with  $d$ .
    - (d)  $\{a-b', c'-d\}$  is an LCN iff  
similar to (a) by exchanging  $a$  with  $b$ , and  $c$  with  
 $d$ .

**For Three-PC CFs**

1. TR1( $\{PC_1, PC_2, PC_3\}$ ): (keep two PCs, remove one)
  - (a) IF  $\neg(CP_1 \parallel CP_2)$  and they intersect at  $CL$   
THEN
    - $\{PC_1, PC_2\}$  is an LCN iff  
( $\neg(CL \parallel CP_3)$ ) AND feasible( $IT, CL, +/-, \delta$ )  
OR  
( $CL \parallel CP_3$ ) AND (No PC is f-f) AND  
(feasible( $IC, CL_1/CL_2, +/-, \delta$ ) while  $PC_3$   
is broken))  
OR  
( $CL \parallel CP_3$ ) AND (at least one PC is f-f)  
AND  
(feasible( $FM, , , \Delta$ ) while  $PC_3$  is broken))
  - (b) IF only  $CP_1 \parallel CP_2$  THEN
    - $\{PC_1, PC_2\}$  is an LCN iff  
(feasible( $IT, \vec{d}_{CP_3}, +/-, \delta$ ) while  $PC_3$  is  
broken,  
where  $\vec{d}_{CP_3}$  is the projection of  $\vec{n}_{CP_3}$  on  
 $CP_1$ ))

- (c) IF  $CP_1 \parallel CP_2$  and  $CP_2 \parallel CP_3$  THEN
- $\{PC_1, PC_2\}$  is an LCN iff  
(Rule PR1 can be applied to  $\{PC_1, PC_3\}$  to yield  $\{PC_1\}$  while  $PC_2$  is maintained)
  - OR  
(Rule PR1 can be applied to  $\{PC_2, PC_3\}$  to yield  $\{PC_2\}$  while  $PC_1$  is maintained)
2. TR2( $\{PC_1, PC_2, PC_3\}$ ): (keep one PC, remove two)
- (a)  $\{PC_1\}$  is an LCN iff  
 $\neg(CP_1 \parallel CP_2)$  AND  $\neg(CP_1 \parallel CP_3)$  AND  
*feasible*(IT,  $\vec{d}_{CP_23}$ ,  $+/-$ ,  $\delta$ ),  
 where  $\vec{d}_{CP_23}$  is the projection of  $\vec{d}$  on  $CP_1$ , and  
 $\vec{d} = \vec{n}_{CP_2} + \vec{n}_{CP_3}$ )
- (b)  $\{PC_1\}$  is an LCN iff  
 $(PC_1$  is not an f-f PC with edge  $e$  or vertex  $v$  involved) AND  
*feasible*(IR,  $v/e$ ,  $+/-$ ,  $\delta$ )
3. TR3( $\{PC_1, PC_2, PC_3\}$ ): (keep two PCs, change one)  
 IF  $PC_3$  is an f-f or f-e/e-f PC THEN
- $\{PC_1, PC_2, PC'_3\}$  is an LCN iff  
 $((CP_1 \parallel CP_2)$  AND  
*feasible*(IR,  $r$ ,  $+/-$ ,  $\delta$ ), where  $r \perp CP_1$  and  $r \perp CP_2$ ))
  - OR  
 $((PC_1$  and  $PC_2$  are not f-f PCs) AND  $(\neg(CP_1 \parallel CP_2))$  AND  
*feasible*(IC,  $CL_1/CL_2$ ,  $+/-$ ,  $\delta$ ) while  $PC_3$  is changed to  $PC'_3$ ))
4. TR4( $\{PC_1, PC_2, PC_3\}$ ): (keep one PC, change two)  
 IF  $PC_1$  and  $PC_2$  are f-f or f-e/e-f PCs THEN
- $\{PC'_1, PC'_2, PC_3\}$  is an LCN iff  
 (Rule PR4 can be applied to  $\{PC_1, PC_2\}$ , while  $PC_3$  is maintained)
5. TR5( $\{PC_1, PC_2, PC_3\}$ ): (change two PCs, remove one)  
 IF  $PC_1$  and  $PC_2$  are f-f or f-e/e-f PCs THEN
- $\{PC'_1, PC'_2\}$  is an LCN iff  
 (Rule PR4 can be applied to  $\{PC_1, PC_2\}$ , while  $PC_3$  is broken)
6. TR6( $\{PC_1, PC_2, PC_3\}$ ): (change one PC, remove two)  
 IF  $PC_1$  is an f-f or f-e/e-f PC THEN
- $\{PC'_1\}$  is an LCN iff  
 (Rule PR3 can be applied to  $\{PC_1, PC_2\}$ , while  $PC_3$  is broken)
7. TR7( $\{PC_1, PC_2, PC_3\}$ ): (keep one, change one, and remove one)  
 IF  $PC_2$  is an f-f or f-e/e-f PC THEN
- $\{PC_1, PC'_2\}$  is an LCN iff  
 (Rule PR2 can be applied to  $\{PC_1, PC_2\}$ , while  $PC_3$  is broken)
8. TR8( $\{PC_1, PC_2, PC_3\}$ ): (change three PCs)  
 IF all the three are f-f or f-e/e-f PCs THEN
- $\{PC'_1, PC'_2, PC'_3\}$  is an LCN iff  
 (Rule PR4 can be applied to  $\{PC_1, PC_2\}$ , while  $PC_3$  is changed to  $PC'_3$ )

## Acknowledgments

This work was supported by the National Science Foundation (IIS-9700412 and CDC-9726424). The authors are in debt to David Johnston for his contribution in writing the interactive three-dimensional display program for viewing contact state graphs. The authors would also like to thank Jean-Claude Latombe, Oussama Khatib, and Jean-Paul Laumond for helpful discussions. Finally, the authors are grateful to the anonymous reviewers for their efforts and comments.

## References

- Avnaim, F., Boissonnat, J. D., and Faverjon, B. 1988. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. *Proceedings of the IEEE International Conference on Robotics and Automation*, April, pp. 1656–1661.
- Brost, R. 1989. Computing metric and topological properties of configuration-space obstacles. *Proceedings of the IEEE International Conference on Robotics and Automation*, May, pp. 170–176.
- Bruyninckx, H., and Schutter, J. 1998. Modeling and specification of compliant motions with two and three contact points. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1938–1943.
- Buckley, S. J. 1989. Planning compliant motion strategies. *International Journal of Robotics Research* 8(5):28–44.
- Dakin, G., and Popplestone, R. 1992. Simplified fine-motion planning in generalized contact space. *IEEE International Symposium on Intelligent Control*, pp. 281–287.
- Desai, R. 1989. On fine motion in mechanical assembly in presence of uncertainty. Ph.D. thesis, University of Michigan.
- Desai, R., Xiao, J., and Volz, R. 1988. Contact formations and design constraints: A new basis for the automatic

- generation of robot programs. In *NATO ARW: CAD Based Programming for Sensor Based Robots*, ed. B. Ravani, 361–395. New York: Springer-Verlag.
- De Schutter, J., Bruyninckx, H., Dutré, S., De Geeter, J., Katupitiya, J., Demey, S., and Lefebvre, T. 1999. Estimating first order geometric parameters and monitoring contact transitions during force controlled compliant motion. *International Journal of Robotics Research* 18(12):1161–1184.
- Donald, B. R. 1985. On motion planning with six degree of freedoms: Solving the intersection problems in configuration space. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 536–541.
- Hirukawa, H. 1997. On motion planning of polyhedra in contact. *Algorithms for Robotic Motion and Manipulation: Workshop on the Algorithmic Foundations of Robotics*, Boston: A. K. Peters, pp. 381–392.
- Hirukawa, H., Papegay, Y., and Matsui, T. 1994. A motion planning algorithm for convex polyhedra in contact under translation and rotation. *Proceedings of the IEEE International Conference on Robotics and Automation*, May, pp. 3020–3027.
- Hocaoglu, C., and Sanderson, A. 1998. Evolutionary path planning using multiresolution path representation. *Proceedings of the IEEE International Conference on Robotics and Automation*, May, pp. 318–323.
- Hopcroft, J., and Wilfong, G. 1986. Motion of objects in contact. *International Journal of Robotics Research* 4(4):32–46.
- Inoue, H. 1974. Force feedback in precise assembly tasks. Report No. AIM-308, AI Lab, Massachusetts Institute of Technology.
- Ji, X. 2000. On generation of contact state space and contact motion planning. Ph.D. thesis, University of North Carolina at Charlotte.
- Ji, X., and Xiao, J. 2001a. On random sampling in contact configuration space. In *New Directions in Algorithmic and Computational Robotics*, ed. B. R. Donald, K. Lynch, and D. Rus. Boston: A. K. Peters.
- Ji, X., and Xiao, J. 2001b. Planning motion compliant to complex contact states. *International Journal of Robotics Research*, forthcoming.
- Johnston, D., and Xiao, J. 2000. On relating the disconnectedness of a topological contact state to the geometric properties of its constituent objects. *Proceedings of the IEEE International Conference on Robotics and Automation*, April, pp. 2284–2289.
- Joskowicz, L., and Taylor, R. H. 1996. Interference-free insertion of a solid body into a cavity: An algorithm and a medical application. *International Journal of Robotics Research* 15(3):211–229.
- Kang, S. C., et al. 1997. A compliant motion control for insertion of complex shaped objects using contact. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 841–846.
- Kavraki, L. E., and Latombe, J. C. 1998. Probabilistic roadmaps for robot path planning. In *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, ed. Gupta, K., and del Pobil, A. P., New York: John Wiley & Sons, pp. 33–53.
- Kavraki, L. E., Svestka, P., Latombe, L. C., and Overmars, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12:566–580.
- Latombe, J. C. 1991. *Robot Motion Planning*. Norwell, MA: Kluwer Academic.
- Lin, M. C., Manocha, D., Cohen, J., and Gottschalk, S. 1996. Collision detection: Algorithms and applications. *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, March.
- Lozano-Pérez, T. 1983. Spatial planning: A configuration space approach. *IEEE Transactions on Computing C-32*:108–120.
- Lozano-Pérez, T., Mason, M. T., and Taylor, R. H. 1984. Automatic synthesis of fine-motion strategies for robot. *International Journal of Robotics Research* 3(1):3–24.
- Mason, M. T. 1982. Compliant motion. In *Robot Motion: Planning and Control*, ed. M. Brady et al., 305–322. Cambridge, MA: MIT Press.
- McCarragher, B. J. 1996. Task primitives for the discrete event modeling and control of 6-DOF assembly tasks. *IEEE Transactions on Robotics and Automation* 12(2):280–289.
- Rosell, J., Basañez, L., and Suárez, R. 1997. Determining compliant motions for planar assembly tasks in the presence of friction. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 946–951.
- Sacks, E., and Bajaj, C. 1998. Sliced configuration spaces for curved planar bodies. *International Journal of Robotics Research* 17(6):639–651.
- Shekhar, S., and Khatib, O. 1987. Force strategies in real-time fine motion assembly. *Proceedings of the ASME Winter Annual Meeting*, Boston, MA.
- Sturges, R. H., and Laowattana, S. 1995. Fine motion planning through constraint network analysis. *IEEE International Conference on Assembly and Task Planning*, Pittsburgh, PA, August, pp. 160–170.
- Whitney, D. E. 1985. Historical perspective and state of the art in robot force control. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 262–268.
- Xiao, J. 1993. Automatic determination of topological contacts in the presence of sensing uncertainties. *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, May, pp. 65–70.



- Xiao, J. 1997. Goal-contact relaxation graphs for contact-based fine motion planning. *1997 IEEE International Conference on Assembly and Task Planning*, Marina Del Rey, CA, August, pp. 25–30.
- Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. 1997. Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation* 1:18–28.
- Xiao, J., and Volz, R. 1989. On replanning for assembly tasks using robots in the presence of uncertainties. *Proceedings of the IEEE International Conference on Robotics and Automation*, May, pp. 638–645.
- Xiao, J., and Zhang, L. 1997. Contact constraint analysis and determination of geometrically valid contact formations from possible contact primitives. *IEEE Transactions on Robotics and Automation* 13(3):456–466.

### Index to Multimedia Extensions

Extension	Media Type	Description
<a href="#">1</a>	Data	Sample input file to <b>GCR-gen</b>
<a href="#">2</a>	Data	Sample output of <b>GCR-gen</b>
<a href="#">3</a>	Images	Display of the sample GCR graph
<a href="#">4</a>	Image	Example of the merged result of two GCR graphs
<a href="#">5</a>	Code	<b>GCR-gen</b> and <b>GCR-merge</b> programs
<a href="#">6</a>	Code	<b>Show-GCR</b> program (executables)
<a href="#">7</a>	Video	Animation of the example in Figure 8
<a href="#">8</a>	Videos	Animations of the Star example

NOTE: GCR = goal-contact relaxation. The multimedia extensions can be found online by following the hyperlinks from [www.ijrr.org](http://www.ijrr.org).