Automatic Generation of High-level Contact State Space*

Xuerong Ji and Jing Xiao
Computer Science Department
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
xji@uncc.edu and xiao@uncc.edu

Abstract

Planning in contact state space is very important for many robotics tasks. This paper introduces a general and novel approach for automatic creation of high-level, discrete contact state space between two objects, called contact formation (CF) graphs. A complete CF graph is the result of merging several special subgraphs, called the goal-contact relaxation (GCR) graphs. We have implemented our algorithm for arbitrary contacting polygons with results presented in this paper and is extending the implementation for arbitrary contacting polyhedra. The time complexity of our algorithm is bounded by $O(N^2)$, where N characterizes the complexity of each object.

1 Introduction

Contact motions, or compliant motions (as they are constrained by contact), are basic motions for manipulation and are necessary for assembly to overcome uncertainties [2, 11, 12, 13, 14, 16, 17]. A contact motion plan can be generally viewed as consisting of a sequence of contact state transitions and compliant motion strategies to realize the transitions. Hence, how to model and represent the knowledge of contacts is a key problem.

In many planning systems which require the knowledge of a collection of contact states, the information is usually fed manually into the system as input. In other words, all contact states and the relations among contact states are enumerated[1] and presented to the system manually. This, however, is awfully burdensome and is practically infeasible for complex tasks where complicated non-convex objects can create a huge variety of different contact states [16]. Recently an algorithm was reported by Hirukawa *et al.* [6, 7] to create certain contact state graphs automatically, but only the result for convex polyhedra was reported.

We need to point out that there is considerable work in the literature computing configuration space (C-space) obtacles, but most are just for 2-D objects [1, 9, 14, 15]. Only a few researchers attempted to compute the C-space obstacles for 3-D polyhedra [5, 10]. Although the surface of C-space obstacles constitute contact configurations, 6-dimensional C-space computation for general 3-D polyhedra remains a formidable open problem. In this paper, we introduce a novel, *alternative* approach aiming at deriving high-level graphs of discrete contact states from exploiting both topological and geometrical knowledge of contacts in the physical space. Note that our approach is general to both 2-D and 3-D objects. Although the results reported in this paper are for 2-D polygons, the implementation of our approach for 3-D polyhedra will soon be reported. Note also that a high-level graph of discrete contact states is not only needed itself for many purposes (see previous paragraphs), but can also be used to simplify computation of C-space paths of (continuous) contact configurations: a high-level sequence of contact state transitions can first be planned with such a contact state graph, and then the computation of the low-level path of contact configurations is reduced to computing the path of a transition between two adjacent contact states in a *lower-dimensional* configuration space (constrained by the *known* contact states).

The paper is outlined below. In Section 2, we review the notion of *contact formation* (CF) in terms of *principal contacts* (PC) [18] as a contact state representation. We also examine the neighboring relation among CFs and represent the contact state space as a CF graph. In Section 3, we introduce a general approach to automatically creating the so-called *goal-contact relaxation graphs* (GCR graphs) [20], which are subgraphs of a CF graph, and merging the GCR

*This research is supported by the National Science Foundation under grant IIS-9700412.

[1] for discrete contact states

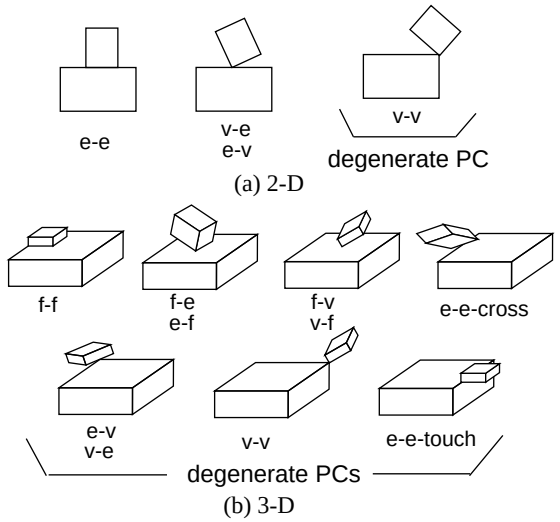

Figure 1: Principal contacts (PCs)

graphs to form the CF graph. In Section 4, we describe the implementation of our approach for generating CF graphs between arbitrary polygons and present the results. In Section 5, we discuss further implementation and extension before concluding the paper.

2 Contact State Space

2.1 Principal Contacts and Contact Formations

The concept of *principal contact* (PC) was first introduced in [18]. Denoting the *boundary elements* of a face as the edges and vertices bounding it, and the boundary elements of an edge as the vertices bounding the edge, a PC can be defined as the following:

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

With the notion of PC, an arbitrary contact state between arbitrary polyhedra is described as the set of PCs formed, which we still call a *contact formation* (CF). There are four and ten types of PCs between arbitrary polygons and arbitrary polyhedra in 2-D and 3-D respectively, as shown in Figure 1.

Note that PCs are higher-level primitives than other topological primitives introduced in the literature [4, 5, 10] and are thus more robust to uncertainties [21].

2.2 Connectivity of Contact Formations

The question about connectivity of contact formation (CF) is whether, for each CF, the contact configurations belonging to it form a connected region, as CFs partition the space of contact configurations. In most practical cases the answer is yes (including all cases in Section 4). *We restrict our consideration to cases where each CF describes a single connected region of contact configurations in the rest of this paper.*

A general definition of neighboring CFs was proposed by Desai [3] in terms of relative contact motions. Specifically, for objects A and B in contact, if there exists a contact motion which brings A and B in CF_i to CF_j such that during the motion, A and B are always in contact and are not in any CF other than CF_i to CF_j, then CF_i and CF_j are *neighboring contact formations*. Desai's definition took into account the important fact proven by Hopcroft and Wilfong [8] that if there was a way to move two objects from one contact configuration to another, then there was a way to do so with the objects remaining in contact throughout the motion.

As CFs describe discrete contact states topologically, we can map the general motion-based definition of neighboring CFs to a topological one in terms of how PCs (or the topological surface elements of the PCs) are related. We first define a containment relation among PCs.

Definition 2: Let $\mathrm{PC}_i = (a^A, b^B)$ and $\mathrm{PC}_j = (c^A, d^B)$ be two PCs between two polyhedra A and B. PC_i *contains* PC_j iff one of the following 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, or
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 among CFs.

Definition 3: Let CF_i and CF_j be two contact formations between polyhedra A and B such that $\mathrm{CF}_i \neq \mathrm{CF}_j$. CF_i *contains* CF_j iff

- $card(\mathrm{CF}_i) \geq card(\mathrm{CF}_j)$, where $card(*)$ returns the cardinality;
- for every PC in CF_j, it either belongs to CF_i or 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.

Corollary: For two CFs, CF_i and CF_j, if $\mathrm{CF}_j \subset \mathrm{CF}_i$ then CF_i contains CF_j.

Now we can define neighboring relations among PCs and among CFs in terms of the corresponding

containment relations.

Definition 4: Two PCs, PC_i and PC_j, are *neighboring PCs*, iff either one contains the other. If PC_i contains PC_j, then PC_j is a *less-constrained neighbor* of PC_i, and PC_i is a *more-constrained neighbor* of PC_j.

Definition 5: Two CFs, CF_i and CF_j ($CF_i \neq CF_j$), are *neighboring CFs*, iff either one contains the other. If CF_i contains CF_j, then CF_j is a *less-constrained neighbor* of CF_i, and CF_i is a *more-constrained neighbor* of CF_j.

We can represent the entire contact state space (of the contacting objects) as a contact formation graph where each node denotes a CF, and each arc connects two neighboring CFs.

3 Generation of CF graphs

Consider two objects in contact, with a moving object A and a static object B. To construct the contact formation graph $\mathcal{G}_{CF}$ automatically requires the handling of two issues: (1) how to generate *valid* CFs, or how to tell if a set of PCs form a geometrically valid CF, and (2) how to connect these CFs in the graph. While the second issue can be handled by applying the neighboring relations defined in Section 2, the first one is more difficult in general because the domain of valid CFs depends on the geometric shapes of the contacting objects[2].

3.1 Approach

Our approach is to divide the contact formation graph $\mathcal{G}_{CF}$ into certain subgraphs, generate the subgraphs, and then merge them to form $\mathcal{G}_{CF}$.

The kind of subgraph we generate consists of a seed CF, CF_g, and all the less-constrained CFs in its neighborhood, which we call a *Goal-Contact Relaxation* (GCR) graph of CF_g [20]. Starting from such a goal CF, CF_g, the GCR graph can be grown by repeatedly "relaxing" contact constraints to obtain less-constrained neighboring (LCN) CFs. The process terminates when there is no new LCN CF to be added to the graph[3]. As for the goal CFs, they are the locally most-constrained CFs in $\mathcal{G}_{CF}$, many of which indicate goal or intermediate goal CFs of an assembly. Given the goal CFs, $\mathcal{G}_{CF}$ can be formed either partially or completely by merging the corresponding GCR graphs.

We prefer to form $\mathcal{G}_{CF}$ from GCR graphs because a GCR graph is much easier to generate automatically than an arbitrary subgraph of $\mathcal{G}_{CF}$, 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, and (2) the transition motion from a CF to an LCN is often simpler than that to a more-constrained neighboring CF. Our algorithm for constructing a GCR graph can be outlined below:

Add CF_g to FIFO queue *open*;
WHILE *open* is not empty DO
BEGIN
1) $CF_c \leftarrow$ CF removed from *open*;
2) hypothesize LCN CFs of CF_c (Section 3.2);
3) find feasible LCN CFs (Sections 3.3 & 3.4);
4) for each feasible LCN CF found, link it to CF_c and add it to *open*.
END

After GCR graphs are generated, our merge algorithm performs merging by taking the union of the nodes in all the GCR graphs and their neighboring arcs (connections). Note that, as we associate each CF node with a contact configuration belonging to it for feasibility check (Sections 3.3 and 3.4) and for display (Section 4), there can be two or more nodes of the same CF but with different associated configurations during merging. In such cases, only one node is kept, others will be discarded.

3.2 Hypothesizing LCN CFs

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 3–5 (Section 2.2):

remove PC_i,
change PC_i to an LCN PC,
keep PC_i,

provided that either **remove** or **keep** is *not* simultaneously applied to all PCs in the CF to result in an empty set or the CF itself[4].

[2] Only in the case of two convex objects, it is trivial to decide a valid CF: every valid CF consists of a single PC, and every possible PC is a valid CF.

[3] Clearly, such process will always terminate.

[4] This obviously implies that for a single-PC CF, only **change** can be applied.

3.3 Hypothesizing Neighboring Relaxation Motions

To determine if a hypothesized LCN CF is feasible is to check if there is a feasible *neighoring transformation motion* to that LCN CF.

A neighboring transformation motion consists of up to two parts:

- *reconfiguration*[5] which changes contact configurations within the same CF,
- *transition* which changes a contact configuration in a CF to one in the neighboring CF and is an infinitesimal motion.

In the case when a CF Q is an LCN of another CF P, a neighboring transformation motion between P and Q is further called a *neighboring relaxation motion*.

Neighboring relaxation motion has the following important characteristics which make its feasibility check much easier than that of a general neighboring transformation motion:

- in many cases, there is no need for reconfiguration, i.e., there is a neighboring relaxation motion consisting of only an infinitesimal transition motion;
- in the other cases, only a finite translation is needed for reconfiguration.

For 2-D objects (i.e., contacting polygons), there are a finite number of possible relaxation motions, in terms of a finite number of axes and directions of motions, to change any CF to one of its LCN CFs. The possible (or hypothesized) motions can be specified topologically by the edge elements involved in the change, and every finite translation for reconfiguration is a straight-line translation. For example, in Figure 2, to change an edge-edge PC (e_1^A, e_1^B) to a vertex-edge PC that it contains, (v_1^A, e_1^B), there are only two possible neighboring relaxation motions: either (i) just an *infinitesimal rotation* (IR) about v_1^A or (ii) a *finite translation* (FT) along e_1^B succeeded by an IR about v_1^A.

For 3-D objects (i.e, contacting polyhedra), however, certain change of CF to its LCN can be achieved by an infinite number of motions, in terms of an infinite number of axes/directions, and for reconfiguration, the finite translation is not limited to a straight-line. In cases where there can be an infinite number of possible neighboring relaxation motions, it is necessary to discretize the motions by sampling the motion axes. In cases where reconfiguration is needed, finding a feasible finite translation can be treated as a local path planning problem.

[5] This term was first introduced by Desai in [3].

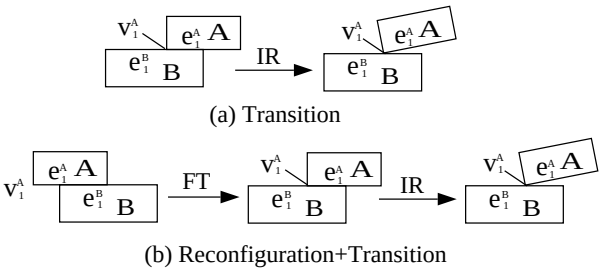

Figure 2: Neighboring relaxation motions

3.4 Checking Motion Feasibility

Once a neighboring relaxation motion is hypothesized, we need to check its feasibility. As explained in Section 3.3, we only need to check the feasibility of an infinitesimal transition motion and that of a finite translation.

To check the feasibility of an infinitesimal transition motion is relatively easy. Let $\mathbf{n}$ denote the normal of a *contact plane*, defined by a contacting face or two crossing edges (or of a *contact line*, defined by a contact edge, for 2-D objects) pointing towards the static object B. Given the type, axis, and direction of an infinitesimal motion, the feasibility of the motion is checked as folllows.

Infinitesimal Translation (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 (or contact lines) into B: $\mathbf{v} \cdot \mathbf{n} \leq 0$ means no penetration through the contact plane with normal $\mathbf{n}$.

Infinitesimal Rotation (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 (or contact line) with normal $\mathbf{n}$, i.e., $\mathbf{v} \cdot \mathbf{n} \leq 0$.

Infinitesimal Combined Translation and Rotation (IC): An IC motion can be equivalent to either (i) an IR followed by a guarded translation (GT) or (ii) an IT followed by a guarded rotation (GR) [6], see Figure 3. It is feasible if either (i) or (ii) is feasible. The amount of GT or GR is determined by the δ amount of the IR or IT prior to it respectively[7].

To check the feasibility of a **finite translation** (FT) which follows a straight line is also relatively easy. First, calculate the direction and the amount of motion in terms of a distance vector $\mathbf{d}$. Second, find all the faces and edges of A and those of B which

[6] A *guarded* motion is terminated by a collision/contact.

[7] Theoretically δ can be arbitrarily small, but it is finite in implementation.

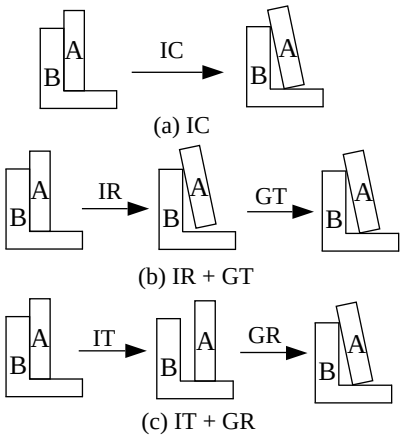

Figure 3: Equivalent motions of IC type of motions

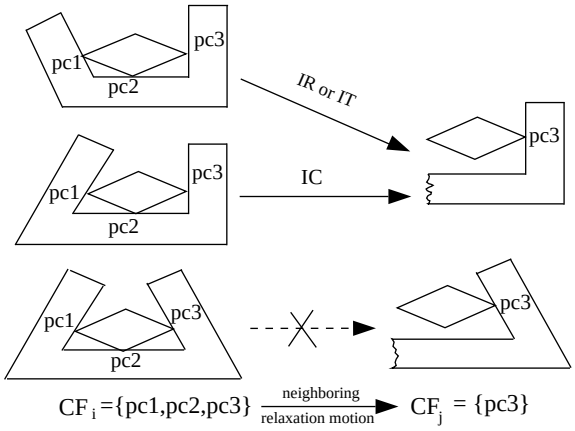

Figure 4: The feasibility of a possible neighboring relaxation motion depends on contact geometry.

roughly face each other as A moves towards B along $\mathbf{d}$ and the bounding vertices of these faces and edges. Last, check whether there is an intersection between such a vertex of one object with the other object as A translates $\mathbf{d}$. If there is intersection, the FT is infeasible; otherwise, it is feasible.

Figure 4 shows a 2-D example to demonstrate the need for feasibility check of motions. From the topological expressions of CF_i and CF_j, in the upper two cases it is feasible to transmit the CF from CF_i to CF_j by some infinitesimal transformations, but in the bottom case, there is no infinitesimal transformation to do that. The actual feasibilities of motions depend on the actural *geometry* of objects.

3.5 Complexity Analysis

Suppose two polyhedra (or polygons for 2-D) A and B have the same number of faces (or edges for 2-D polygons), represented by N, for simplicity. According to the Euler formula, the total number of edges and the total number of vertices are $3N-6$ and $2N-4$ respectively for polyhedra. Thus, the average number of edges/vertices per face for a polyhedra is bounded by the constant 6 (and the number of vertices per edge for a polygon is 2). Further, we will only consider CFs with ≤ 3 PCs since in an n-PC CF ($n > 3$), many PCs are not independant so that the CF is often found equivalent to a certain CF with ≤ 3 PCs [19].

Now we describe the asymptotic complexity of our algorithm. Due to limited space, we will only present the results of our analysis. First, it can be shown that the maximum number of GCR graphs, i.e., the maximum number of locally most constrained (or "seed") CFs, S, is bounded by $O(N^2)$. Second, it can be shown that the number of nodes in one GCR graph is much smaller than the constant 6^6 for 3-D polyhedra (or 2^6 for 2-D polygon). Hence, the total number of nodes W of the entire contact state graph is much smaller than 6^6S (or 2^6S for 2-D polygon), and is bounded by $O(N^2)$.

To generate each CF node requires either just a constant time based on local contact information or a constant number of feasibility check of FT motions, where each FT feasibility check requires time $O(N)$. It can be shown that the total number of CF nodes in the entire contact state graph which require FT feasibility check, W_1, is bounded by $O(N)$. Hence, the total time to generate those CF nodes is bounded by $O(N^2)$. In addition, the total number of nodes without requiring FT feasibility check is $W - W_1$, which is bounded $O(N^2)$. Therefore, the total time T to generate the entire contact state graph using our algorithm is bounded by $O(N^2)$.

4 Implementation and Results

We have implemented the approach of constructing and merging GCR graphs for arbitrary CFs with $\leq$ 3 PCs between arbitrary polygons. The program is written in C using Xwindow graphics library on Sun Ultra 10 workstation. Note that to make the graph look less crowded, we have made a modification on the definition of neighbors, i.e., *case 1 is removed from Definition 2*. With such modification, the number of neighbors for each CF is reduced, and so is the number of arcs in a GCR graph. However, the modification does not affect the number of CFs (i.e., nodes) in a GCR graph.

Figure 5 shows the goal CFs of several pairs of polygons which contain 2 or 3 PCs and the number of

Seed CF		# of nodes in GCR	Seed CF		# of nodes in GCR
1PC		9	3PC		10
		6			16
2PC		13			8
		11			8
		10			15
		10			13
		11			25

Figure 5: Some seed CFs and number of nodes in the GCR graphs generated from them.

nodes in the GCR graphs generated from these seed (goal) CFs.

Figure 6 shows one GCR graph of a peg-in-hole assembly (the node with '$*$' is the seed node), and Figure 7 shows the merged result of several GCR graphs for the peg-in-hole assembly, all produced by our GCR graph generation and merging algorithms.

We have run our algorithm for constructing GCR graphs on over 40 different examples involving polygon pairs of vastly different shapes (some of them are shown in Figure 5). The running time has always been in the order of several milliseconds on a SUN Ultra 10 workstation. For example, the GCR graph in Figure 6 was generated in only 1.6 milliseconds.

5 Discussion and Conclusion

We have introduced a general approach for automatic construction of contact state space, in terms of CF graphs of the objects in contact. Effective implementation has been completed (a) for automatically generating GCR graphs of arbitrary CFs formed by ≤ 3 PCs between two arbitrary polygons and (b) for merging GCR graphs.

We are extending the implementation to generating GCR graphs of CFs between 3-D polyhedra. It is worth emphasizing that our approach is general for

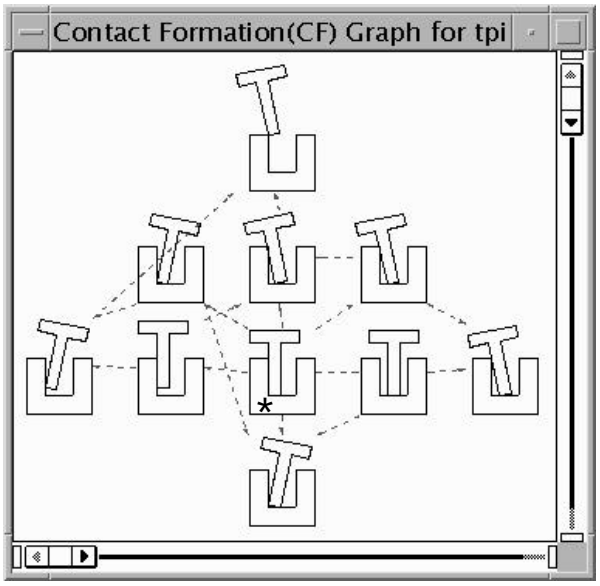

Figure 6: One GCR graph of the peg-in-hole assembly: the node with '$*$' is the seed node.

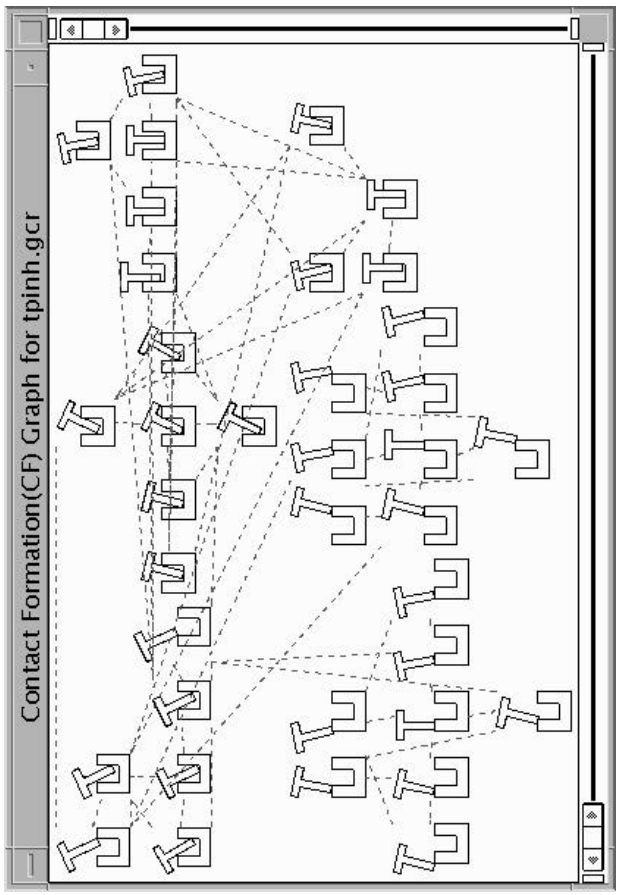

Figure 7: Merged result of several GCR graphs of the peg-in-hole assembly.

both 2-D and 3-D cases except for the detailed selection of possible neighboring relaxation motions, as explained in Section 3.3. Now we are incorporating sampling and local path planning in our algorithm in order to handle 3-D cases, and we expect to achieve preliminary results soon.

References

[1] Randy C. Brost, "Computing Metric and Topological Properties of Configuration- Space Obstacles", *IEEE Int. Conf. Robotics & Automation*, pp. 170-176, 1989.

[2] G. Dakin, and R. Popplestone, "Simplified Fine-Motion Planning in Generalized Contact Space", *IEEE Int. Symp. on Intell. Control*, pp. 281-287, 1992.

[3] R. Desai, *On Fine Motion in Mechanical Assembly in Presence of Uncertainty*, Ph.D. thesis, Dept. of ME, Univ. of Michigan, 1989.

[4] R. Desai, J. Xiao and R. Volz, "Contact Formations and Design Constraints: A New Basis for the Automatic Generation of Robot Programs", *NATO ARW: CAD Based Prog. for Sensor Based Robots*, pp. 361-395, 1988.

[5] B. R. Donald, "On Motion Planning with Six Degree of Freedoms: Solving the Intersection Problems in Configuration Space", *IEEE Int. Conf. Robotics & Automation*, pp. 536-541, 1985.

[6] H. Hirukawa, Y. Papegay, and T. Matsui, "A Motion Planning Algorithm for Convex Polyhedra in Contact under Translation and Rotation", *IEEE Int. Conf. Robotics & Automation*, pp. 3020-3027, May 1994.

[7] H. Hirukawa, "On Motion Planning of Polyhedra in Contact", *Proc. of the Works. on Algor. Found. of Robotics*, 1996.

[8] J. Hopcroft and G. Wilfong, "Motion of Objects in Contact", *Int. J. Robotics Res.*, 4(4):32-46, 1986.

[9] L. Joskowicz, R. H. Taylor, "Interference-Free Insertion of a Solid Body Into a Cavity: An Algorithm and a Medical Application", *Int. J. Robotics Res.*, 15(3):211-229, June 1996.

[10] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach", *IEEE Trans. Comput.*, C-32(2):108-120, 1983.

[11] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic Synthesis of Fine-motion Strategies for Robot", *Int. J. Robotics Res.*, 3(1):3–24, Spring 1984.

[12] B. J. McCarragher, and H. Asada, "A Discrete Event Approach to the Control of Robotic Assembly Tasks", *IEEE Int. Conf. Robotics & Automation*, pp. 331-336, 1993.

[13] B. J. McCarragher, "Task Primitives for the Discrete Event Modeling and Control of 6-DOF Assembly Tasks", *IEEE Trans. Robotics and Automation*, 12(2):280-289, April 1996.

[14] J. Rosell, L. Basañez, and R. Suárez, "Determining Compliant Motions for Planar Assembly Tasks in the Presence of Friction", *Proc. of the 1997 IEEE/RRSJ Int. Conf. on Intell. Robots & Sys.*, pp. 946-951.

[15] E. Sacks, C. Bajaj, "Sliced Configuration Spaces for Curved Plannar Bodies", *Int. J. Robotics Res.*, 17(6):639-651, June 1998.

[16] R. H. Sturges, and S. Laowattana, "Fine Motion Planning through Constraint Network Analysis", *IEEE Int. Sym. Assembly and Task Planning*, pp. 160-170, Aug. 1995.

[17] J. Xiao, "Replanning with compliant rotations in the presence of uncertainties", *IEEE Int. Sym. Intell. Cont.*, pp. 102-107, Aug. 1992.

[18] J. Xiao, "Automatic Determination of Topological Contacts in the Presence of Sensing Uncertainties", *IEEE Int. Conf. Robotics & Automation*, pp. 65-70, May 1993.

[19] J. Xiao and L. Zhang, "Contact Constraint Analysis and Determination of Geometrically Valid Contact Formations from Possible Contact Primitives", *IEEE Trans. Robotics and Automation*, 13(3):456-466, June 1997.

[20] J. Xiao, "Goal-contact Relaxation Graphs for Contact-based Fine Motion Planning", *1997 IEEE Int. Sym. Assembly and Task Planning*, pp. 25-30, Aug. 1997.

[21] J. Xiao and L. Liu, "Contact States: Representation and Recognizability in the Presence of Uncertainties", *IEEE/RSJ Int. Conf. Intell. Robots and Sys.*, Oct. 1998.