Automatic Verification of Contact States Taking into Account Manipulator Constraints

Wim Meeussen*†, Jing Xiao*, Joris De Schutter†, Herman Bruyninckx† and Ernesto Staffetti*
* Computer Science Department
University of North Carolina, Charlotte, USA
† Department of Mechanical Engineering
Katholieke Universiteit Leuven, Belgium

***Abstract*— Compliant motion is required or desirable in many robotic tasks, especially assembly tasks. Both planning and execution of autonomous compliant motion requires the knowledge of contact states between parts in contact beforehand. Previous research has addressed automatic generation of contact states between rigid objects. However, not all such contact states can be possibly reached if a rigid object is attached to and moved by a manipulator due to the manipulator constraints. In this paper, we study the problem of finding feasible contact states between a polyhedral part A held by a manipulator with a fixed base and a fixed polyhedral part B. Given a contact state graph between an unattached part A and a fixed part B, our approach then attaches A to the manipulator model and checks the reachability of each contact state and the connection between two neighboring contact states by applying a virtual compliant controller to the manipulator to test possible compliant motions of A. Implementation results validate the effectiveness of our method.**

I. Introduction

Many robotic applications cannot avoid the "contact sports" – compliant motions between a robot or the object held by a robot and the environment. Compliant motion planning and control requires the knowledge of contact topology and geometry, as characterized by certain discrete contact states and state transitions [6], [7], [10], [11]. Such knowledge is often manually extracted and fed into a system as input, which can be extremely tedious, incomplete, and error prone for even tasks of simple geometry [11] and is practically infeasible for complex tasks due to the huge number of complex contact states. To address the problem, Xiao and Ji [14] developed a systematic approach to generate automatically the contact state space between two arbitrary polyhedral objects in terms of a contact state graph. Each node in the graph denotes a contact state, described by a *contact formation* (CF) [12] and a representative configuration of the CF, and an arc between two nodes indicates feasible neighboring state transitions. More recently, an alternative method is also reported for generating a similar contact state graph automatically [8].

However, in many robotic tasks, it is a robotic manipulator that moves an object and creates contacts between the object and the environment. Therefore, whether a contact state can be formed and whether a contact state transition is possible is subject to the constraints of the manipulator. It is necessary to take into account such a manipulator in obtaining a contact state graph so that a compliant motion plan generated based on such a graph can be actually executed by the manipulator. In this paper, we study the problem of finding feasible contact states between a polyhedral part A held by a manipulator and a fixed polyhedral environment B. Given a contact state graph between an unattached part A and a fixed environment B as generated by the method of [14], our approach then attaches A to a manipulator model and checks the reachability of each contact state and the connection between two neighboring contact states by applying a virtual compliant controller to the manipulator to create possible compliant motions of A.

The rest of the paper is organized as follows. In Section II, we briefly review the concepts of principal contacts and contact formations to represent contact states and the generation of such a contact state graph as presented in [14]. In Section III, we present an overview of our approach to re-examine the feasibility of contact states and state transitions by taking into account the manipulator constraints.

In Section IV, we explain the virtual compliant controller and how we can use it to generate a compliant motion in detail. In Section V, we explain how the virtual controller is used to generate the desired compliant relaxation motions. We explain how we check the feasibility of compliant relaxation motions in Section VI and conclude the paper in Section VII.

II. Review: Contact States, State Graph and Generation

The notion of *Principal Contacts* (PCs) was introduced [12] to describe a contact primitive between two surface elements of two polyhedral objects in contact, where a surface element can be a face, an edge, or a vertex. The *boundary elements* of a face are the edges and vertices bounding it, and the boundary elements of an edge are the vertices bounding it. Formally, a PC denotes the contact between a pair of surface elements which are not boundary elements of other contacting surface elements. As an example, each case in figure 1 is described by a unique, non-degenerate PC. Each non-degenerate PC is associated with a *contact plane*, defined by a contacting face or the two contacting edges at an e-e-cross PC. A general contact state between two objects can be characterized topologically by the set of PCs formed, called a *contact formation* (CF). A CF is geometrically described by the set of relative contact configurations that satisfy the contact constraints of all the PCs in the CF, called *CF-compliant configurations*. Any motion

formed by a sequence of CF-compliant configurations is called a *CF-compliant motion*. A contact state can be defined in terms of a CF and a CF-compliant configuration. This high-level, discrete contact state representation is proven to be sufficient to capture topological, geometrical, and physical properties of contacts as well as efficient to use [4].

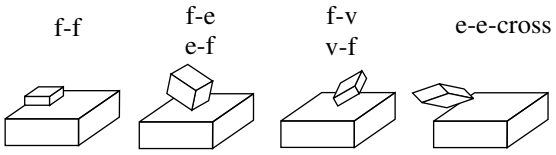


Fig. 1. Non-degenerate Principal Contacts (PCs)

Xiao and Ji developed a divide-and-merge approach [13], [14] to generate the contact state space between two polyhedral objects as a contact state graph G. Specifically, the approach takes advantage of the fact that G can be divided into special subgraphs called the *goal-contact relaxation* (GCR) graphs, where each GCR graph is defined by a locally most constrained contact state, called the seed, and its less-constrained neighboring contact states, which is easier to generate because of several nice properties. The main properties include:

- Given a valid contact formation, CF_i , all of its less constrained neighboring CFs can be hypothesized topologically from the PCs in CF_i.
- A hypothesized less constrained neighboring contact formation, CF_j, is valid, if and only if there exists a compliant motion to relax certain constraints of CF_i to obtain CF_j that does not result in any other CF. Such a compliant motion is called a *neighboring contact relaxation motion*.
- Neighboring relaxation can be generally achieved by a CF_i-compliant motion, followed by an instantaneous compliant motion for state transition and a CF_j-compliant motion, and most neighboring relaxation can be achieved by instantaneous compliant transition motion alone.

The approach was implemented with algorithms to generate a complete GCR graph automatically and to merge multiple GCR graphs automatically into a single contact state graph [14]. It is shown to be both effective and efficient in generating contact state graphs of hundreds and thousands of different contact states and their connections within seconds.

III. State Graph Revisited under Manipulator Constraints

Given a moveable part A and a fixed environment B, the contact state graph G generated by the approach in [14] essentially assumes that A can move by itself. However, in an actual robotic task, it is usually a robot manipulator that moves A. Therefore we need to check if a contact state in G (denoted by a node) and a state transition in G (denoted by an arc) that are feasible for A are still feasible taking into account the manipulator constraints.

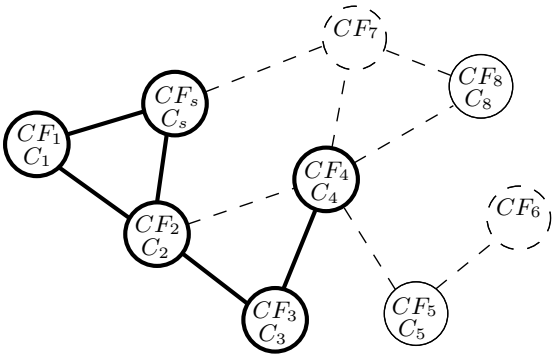


Fig. 2. Revised contact state graph. Feasible nodes and arcs in are solid lines, infeasible nodes and arcs are in dashed lines. The resulting subgraph is shown in bold.

We combine the model of A and that of a manipulator with a fixed base to form the model that represents A being held by the manipulator end-effector. We call the combined model the *held* A. The configuration of the held A should be understood as the joint space configuration of the manipulator holding A. Now we can define a *CF-compliant configuration of the held* A as a (joint space) configuration of the held A where all links except the single part A are collision-free and the configuration of the single part A is CF-compliant. Moreover, a *CF-compliant motion of the held* A is defined as consisting of only CF-compliant configurations of the held A. A *feasible state transition motion of the held* A between two neighboring contact formations CF_i and CF_j in G is defined as consisting of a CF_i-compliant motion and a CF_j-compliant motion of the held A.

First, we need to check, for each contact state in G, whether a CF-compliant configuration of the held A can be found. Let s denote a contact state in G with a contact formation CF_s and a CF_s-compliant configuration $C^A_{s,0}$ of the single part A. If there exists an inverse kinematic solution C_s of C^A_s such that C_s is a CF_s-compliant configuration of the held A, then the state s is re-expressed as $<CF_s, C_s>$ and is labeled as *feasible*. Otherwise, another CF-compliant configuration $C^A_{s,1}$ is sampled [3], [4] with its position component within the working envelope of the manipulator, and inverse kinematics is performed again. This process can continue until either (1) a CF_s-compliant configuration C_s of the held A is found and s is labeled feasible, or (2) no CF_s-compliant configuration of the held A can be found after a certain number of samples. In the latter case, the contact formation CF_s is considered difficult or impossible to reach, and the state s is labeled *infeasible*.

Next, we perform a graph traversal of G, verifying if an arc in G, representing a compliant transition between two neighboring contact states s_i and s_j in G, is still possible with the held A, if s_i and s_j are both feasible for the held A. The graph traversal starts from a feasible contact state s in G for the held A (see figure 2). The algorithm can be written using a recursive function Feasibility-Check:

initialize:
$state \Leftarrow s$
function: Feasibility-Check $(state)$
for each arc a_i between $state$ and a feasible neighbor $state_i$ **do**
  if a_i is not labeled feasible AND there is a feasible state transition motion of the held A from $state$ to reach $state_i$ **then**
    a_i is labeled feasible;
    Feasibility-Check $(state_i)$;
  end if
end for

The result will be a subgraph of G showing feasible contact states and feasible state transitions for the held A, as shown in figure 2. The essence of *Feasibility-Check* is to see if a neighboring state transition motion between two feasible states is possible for the held A. The checking is done by a virtual compliant controller through a compliant relaxation motion from the more constrained to a less constrained contact state, as described in the following sections.

IV. Virtual Compliant Controller

In our approach, we use a virtual controller to generate a path of feasible configurations for the held A that describes a compliant relaxation motion. The motion is generated step by step in small movements for the held A, steering it towards its goal. This time-step based approach avoids the difficulty of generating configuration space obstacles [1]. Also, we avoid the problem of *manipulator jumps*, i.e. when using inverse kinematics, two nearby Cartesian configurations of the moved object A could give two significantly different manipulator configurations.

A. Local Task Specification

The velocity of the held A is not directly specified in the joint space of the manipulator, but instead we use multiple local specifications in the Cartesian space. For this purpose, we further decompose each PC in a CF into *elementary contacts* (ECs): vertex-face, face-vertex and edge-edge contacts, which are associated with a *contact point* and a *contact normal*, as shown in figure 3.

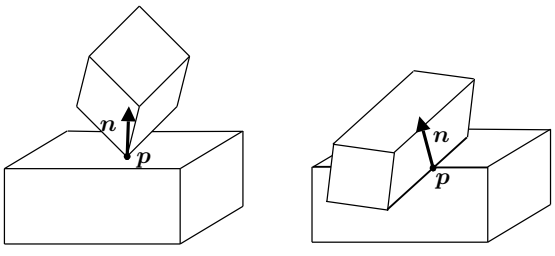


Fig. 3. Elementary contacts (ECs) are associated with a contact point p and a contact normal n

Now we can specify the complex joint space motion of the held A using simple local specifications at ECs between the held A and the environment B. At each EC we specify a local force between the held A and the environment by virtually attaching a linear spring between the surface elements of the EC. The spring is positioned along the contact normal of the EC through the contact point of the EC, as shown in figure 4. The force applied by the spring on the surface elements depends on the stiffness of the spring, its rest length and the distance between the surface elements. The next section describes how we use the local specifications at ECs to specify the compliant motion of the held A.

TABLE I
Spring constants (stiffness K and rest length x_0) for different functions of ECs

	K	x_0
Constraining EC	high	zero
Relaxation EC	medium	medium
Unintended EC	medium	high
Goal EC	low	zero

B. Compliant Motion Specification

To specify the complex joint space motion of the held A, using local specifications at ECs, we distinguish different functions for ECs, such as ensuring compliant motion, avoiding obstacles or goal attraction. Depending on the function of an EC we use a different local specification by different virtual springs. Table I gives an overview of the springs used for each different function of an EC.

A CF-compliant motion can be realized by maintaining the ECs that are required in the CF. We call these ECs the *constraining ECs*. To maintain a constraining EC, we attach a spring between the surface elements of the EC, to ensure that the contacting elements of the EC remain in contact.

An instantaneous relaxation motion from $CF_i \rightarrow CF_j$ can be realized by keeping the constraining ECs of CF_j and breaking the ECs in CF_i that are not allowed in CF_j. We call the ECs that need to be broken to realize the relaxation motion the *relaxation ECs*. To break a relaxation EC we position a compressed spring between the contacting surface elements of the EC, to push them apart.

To perform local obstacle avoidance, we avoid all ECs between the held A and the environment that are not needed for the contact formation. We call these ECs the *unintended ECs*. To prevent a collision at an unintended EC, we position a compressed spring between the surface elements of the EC, to push them apart.

The attraction between the held A and a desired goal configuration is based on the ECs that are required in the goal configuration. We call these ECs the *goal ECs*. At each goal EC we position a spring that pulls the surface elements towards the goal configuration.

To avoid a joint from reaching the end of its range, we position a torsion spring at the joint, to push the joint away from its end position.

Figure 4 shows a configuration of the held A with three ECs. At each EC the contact normal n_i and the spring between the

surface elements at the contact are shown.

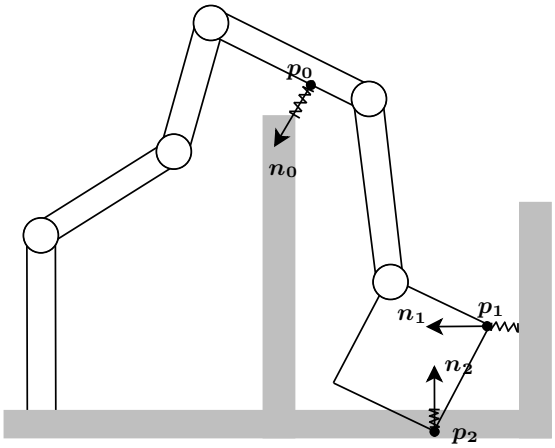


Fig. 4. Contact normals: $\boldsymbol{n_0}$ for local obstacle avoidance, $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ for maintaining contact.

C. Velocity of the held A

Now we describe how the instantaneous motion of the held A, i.e. the motion at each time step, represented by the joint motion of the manipulator, is obtained from the given local specifications at the ECs between the held A and the environment B. The motion specification of the held A consists of a set of springs at ECs. Using the spring stiffness, its rest length and the distance at the EC, we obtain the force applied by the spring on the surface elements of the EC. The joint space motion of the held A is then obtained by simulating the dynamics of the serial chain, using the forward dynamics of a serial chain.

D. Implementation

We have implemented this virtual compliant controller for an industrial manipulator with six degrees of freedom, the Kuka 361. Figure 5 shows the manipulator model we use.

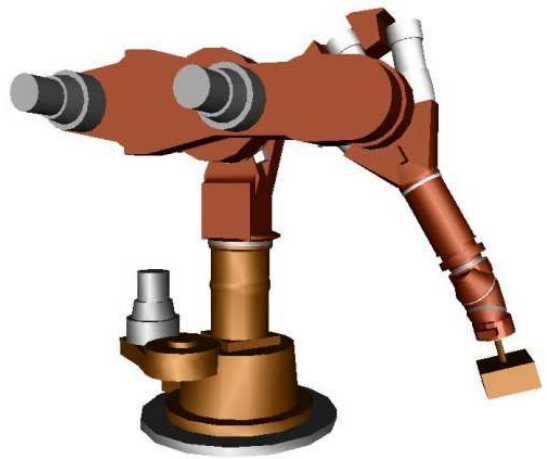


Fig. 5. The Kuka 361 industrial manipulator

In each time step of the virtual controller we search all ECs that exist in the configuration of the held A at that time, attach virtual springs to the surface elements of each EC, and simulate the motion of the held A during the time step caused by the forces of the springs. For finding the ECs we use the collision detection algorithm by Gilbert et. al [2] to find the closest features between two polyhedrals, extended with Zhang's algorithm [15] to find all PCs between two polyhedrals. The motion of the held A is simulated based on an approximation of the single rigid body dynamics of object A:

$$\sum_i \boldsymbol{w_i} = \boldsymbol{M} \cdot \dot{\boldsymbol{t}} + \boldsymbol{D} \cdot \boldsymbol{t} \tag{1}$$

with $\boldsymbol{w_i}$ the wrench at the reference frame of A generated by the force applied by the spring at EC_i, $\boldsymbol{M} = (\boldsymbol{m}, \boldsymbol{I})$ the mass matrix, $\boldsymbol{D}$ the viscous damping, and $\boldsymbol{t}$ the twist of the single rigid body A. Given the twist of A at the previous time step, we then calculate the twist of A at the next time step using equation 1. This new twist is then converted to the joint velocities of the held A using the inverse instantaneous kinematics for a kinematic chain.

V. Compliant relaxation motion

Given two feasible neighboring contact states $<CF_i, C_i>$ and $<CF_j, C_j>$ with CF_j a less constrained neighbor of CF_i, we search a compliant relaxation motion from C_i to C_j. This compliant relaxation motion consists of three separate motions. First, a CF_i-compliant motion from C_i to an intermediate CF_i-compliant configuration C_i^r that contains all ECs that are required by CF_j. We call this intermediate configuration, which allows an instantaneous relaxation motion to CF_j, the *relaxation configuration*. Then, an instantaneous relaxation motion from C_i^r in CF_i to C_j^r in CF_j, and finally a CF_j-compliant motion from C_j^r to C_j.

A. CF_i-compliant motion

The first CF_i-compliant motion starts from a given CF_i-compliant configuration C_i, towards a CF_i-compliant relaxation configuration C_i^r. Using the virtual controller, we generate this CF_i-compliant motion by keeping the ECs that are required by CF_i, making the new ECs that are required by the less constrained neighbor CF_j, and avoiding collisions at the same time. The new ECs that are required by CF_j act like an attractor towards the relaxation configuration C_i^r.

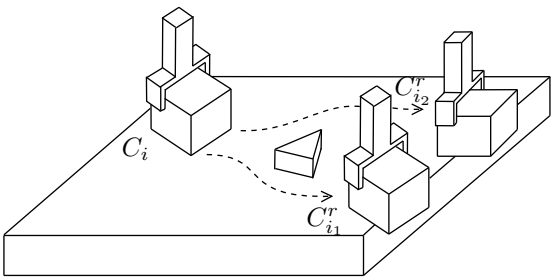
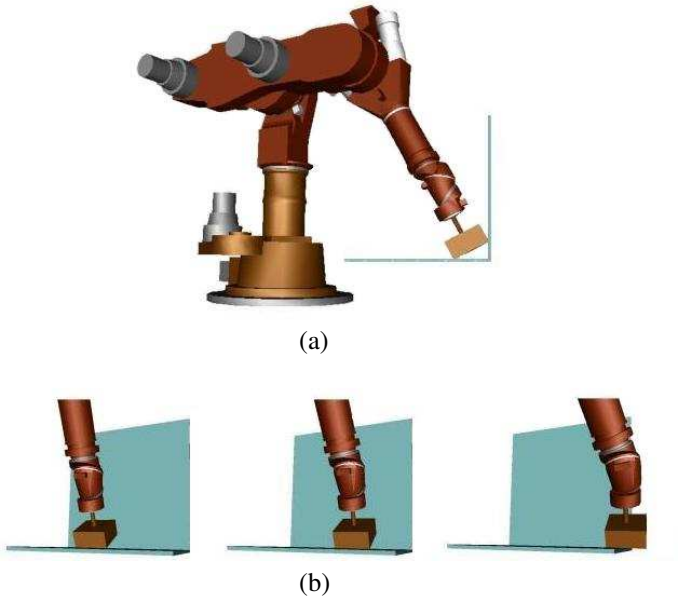


Fig. 6. Possible CF-compliant motions from a configuration C_i to a relaxation configuration C_i^r.

Figure 6 shows two possible compliant motions within a face-face CF, from a start configuration C_i towards one of the relaxation configurations C_i^r. In this example, the relaxation configuration C_i^r can be any configuration, within the face-face CF, that allows an instantaneous relaxation motion towards the desired edge-face CF. The configuration C_i has four vertex-face ECs and each valid C_i^r has two edge-edge ECs, as required by the edge-face CF, and in addition to that C_i^r has one, two or three vertex-face ECs, as required by the face-face CF.

Figure 7 shows an example of a feasible CF-compliant motion, generated by the virtual controller on the Kuka 361.



(a)

(b)

Fig. 7. CF-compliant motion, where the CF has two edge-face PC's. (a) side view and (b) front view

B. Instantaneous relaxation motion

The instantaneous relaxation motion is a compliant motion from a given CF_i-compliant configuration C_i^r to a less constrained CF_j-compliant neighboring configuration C_j^r. This motion is generated by the virtual controller by keeping all ECs in CF_i that are also present in CF_j and breaking all other ECs.

Figure 8 shows an example of a relaxation motion generated by the virtual controller.

C. CF_j-compliant motion

The CF_j-compliant motion starts from the CF_j-compliant configuration C_j^r, towards a given CF_j-compliant configuration C_j. Note that for this CF_j-compliant motion, the start and end configurations are given, unlike for the CF_i-compliant motion, where more than one possible end configuration C_i^r exists. Using the virtual controller we generate a CF_j-compliant motion from C_j^r to C_j by keeping the ECs that are required by CF_j and positioning a torsion spring at each joint

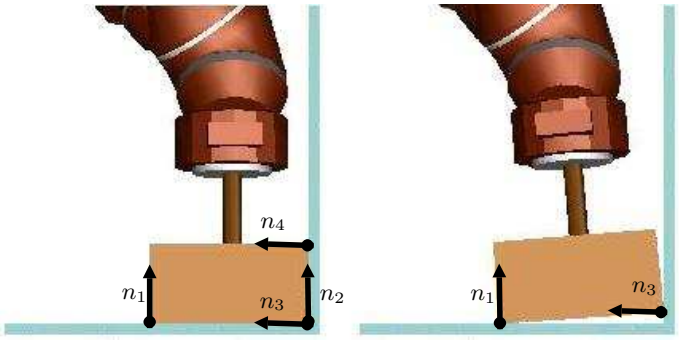


Fig. 8. Compliant relaxation motion, with $\{n_1, \ldots, n_4\}$ the contact normals

to move it towards the position that is required by C_j. Again, other collisions are avoided during the motion.

VI. Feasibility checking

In this paragraph we describe the method to verify if the motions generated by the virtual controller define a feasible compliant relaxation motion.

By using the approach based on a set of local specifications for each configuration of the held A, we in fact build a potential field for the held A, with a global minimum at the goal configuration. Unfortunately, most potential field constructions suffer from the well documented local minima problem [5], [9]. A "trap situation" in a local minimum occurs when, due to a balance between all local specifications, the held A does not move.

When the held A gets trapped in a local minimum, we use the following approach. First we shut down the goal attractor. This disturbs the balance between the local specifications and makes the held A move away form the local minimum. To prevent the held A from getting trapped again after the goal attractor is re-activated, we then apply a random force on the held A during a random time interval, to help steering it to a different configuration.

We repeat this process until (1) the held A escapes from the local minimum and moves again towards the global minimum, or (2) no way out of the local minimum can be found after a certain number of attempts. In the former case, we consider that the generated compliant relaxation motion is so far feasible for the held A; if the held A reaches a goal configuration, we say the compliant path is feasible. In the latter case, we consider the motion infeasible for the held A. Instead of using a simple goal attractor for the held A, we plan to develop a full-fledge compliant motion planner for the held A to generate an optimized compliant path. The emphasis in this paper, however, is not on finding any optimized compliant path, but on finding a feasible compliant path to verify that the compliant relaxation motion between two neighboring contact states can be achieved.

VII. Conclusions

This paper addresses how to revise a valid contact state graph between two polyhedral objects A and B by adding

the constraint that A can only be moved by a manipulator. An effective approach is introduced to check the reachability of contact states and the connections between neighboring states for A held by a manipulator via a virtual compliant controller. A compliant motion path generated based on the revised contact state graph can actually be executed by a manipulator.

Our main short term research goal is to complete and extend the implementation of the approach for generating the revised contact state graph. Next we will study compliant motion planning strategies for the kinematic chain of the object A held by a manipulator, based on the contact state graph. We will consider different optimization criteria in planning and interface the output plan with a low-level controller to execute the planned motion on a real robot manipulator.

ACKNOWLEDGMENT

All authors gratefully acknowledge the financial support by the U.S. National Science Foundation under grant IIS-0328782 and K.U.Leuven's Concerted Research Action GOA/99/04.

REFERENCES

[1] B. Donald. On motion planning with six degrees of freedom: solving the intersection problems in configuration space. In *Proceedings of the 1985 IEEE International conference on Robotics and Automation*, pages 536–541, March 1985.

[2] E. Gilbert, D. Johnson, and S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, April 1988.

[3] X. Ji and J. Xiao. Planning motions compliant to complex contact states. *The international Journal of Robotics Research*, 20(6):446–465, June 2001.

[4] X. Ji and J. Xiao. Random sampling of contact configurations in two-pc contact formations. In K. Lynch B.R. Donald and D. Rus, editors, *New Directions in Algorithmic and Computational Robotics*, Boston, 2001. Kluwer.

[5] B. H. Krog. A generalized potential field approach to obstacle avoidance control. In *The next five years and beyond*, Bethlehem, PA, USA, August 1984. International Symposium of Robotics Research.

[6] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Polyhedral contact formation modeling and identification for autonomous compliant motion. *IEEE Transactions on Robotics and Automation*, 19(1):26–41, February 2003.

[7] B. J. McCarragher and H. Asada. The discrete event modeling and trajectory planning of robotic assembly tasks. *ASME Journal ofDynamic Systems, Measurement, and Control*, 117(3), 1995.

[8] F. Pan and J. Schimmels. Efficient contact state graph generation for assembly applications. In *IEEE Int. Conf. Robotics and Automation*, pages 2592–2598, Taipei, Taiwan, September 2003.

[9] E. Rimon and D. E. Kodischek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[10] J. Schimmels and M. Peshkin. Admittance matrix design for force guided assembly. *IEEE Transactions on Robotics and Automation*, 8(2):213–227, August 1992.

[11] R. H. Sturges and S. Laowattana. Fine motion planning through constraint network analysis. In *IEEE International Symposium on Assembly and Task Planning*, pages 160–170, Pittsburgh, PA, August 1995.

[12] J. Xiao. Automatic determination of topological contacts in the presence of sensing uncertainties. In *Proceedings of the 1993 IEEE Int. Conf. Robotics and Automation*, pages 65–70, Atlanta, GA, USA, May 1993.

[13] J. Xiao and X. Ji. A divide-and-merge approach to automatic generation of contact states and contact motion planning. In *Proceedings of the 2000 IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, USA, April 2000.

[14] J. Xiao and X. Ji. On automatic generation of high-level contact state space. *International Journal of Robotics Research*, 20(7):584–606, July 2001.

[15] L. Zhang and J. Xiao. Derivation of Contact States from Geometric Models of Objects. In *Proceedings of the IEEE Int. Conf. Assembly and Task Planning*, pages 375–380, Pittsburgh, PA, August 1995.