Derivation of Contact States from Geometric Models of Objects *

Lixin Zhang and Jing Xiao
Computer Science Department
University of North Carolina - Charlotte
Charlotte, NC 28223, USA

Abstract

An efficient algorithm is presented to identify the topological contacts between two contacting polytopes from their geometric models. The algorithm extends the result of the polytope distance algorithm by Gilbert et. al *[3] by exploring local topological information. Its worst-case time complexity is $O(N + n_{e/v}^2)$, where N is the total number of vertices of two contacting polytopes and $n_{e/v}$ is the average number of edges at one vertex. The algorithm is fully implemented and used in a computer simulation system for a contact-based fine motion planning scheme [7]. With its great efficiency and effectiveness, the algorithm can be used in any computer modeling and simulation environment which require reasoning about topological contacts.*

1 Introduction

In many simulation problems of robotic tasks, particularly simulation of manipulation tasks and rigid-body dynamics (e.g., [7, 5, 2]), it is often necessary to have a mechanism which tells whether and how two "virtual" objects, characterized by their geometric models, are in contact. In addition, certain offline motion planning algorithms also require the derivation of topological contact information from two object models. For instance, the path-finding algorithm recently introduced in [6] uses the gradient of the growth distance between two polytopes as the heuristic for searching valid paths. The computation of such gradient requires the knowledge of the topological contact state between two contacting grown objects, given the geometric object models.

Deriving contact information from the geometric models of two "virtual" objects in a computer simulation environment, poses a unique problem. As shown in Fig.1, due to discrete motion (figure (a)) or accumulation of digital computation errors (figure (b)), two objects which are supposed to be in contact may often penetrate into each other or not share any points. To deal with this problem, we must first modify the conventional definition of contact (i.e., two objects share boundary points) by including some tolerance.

Definition 1 *In computer modeling or simulation environments, two objects A and B are* in contact *iff their distance d is within the interval $[-\epsilon, \epsilon]$, where ϵ is a very small positive number, and a negative d indicates that A and B interpenetrate and $|d|$ is the shortest distance to separate them.*

This is a very general definition. However, the value of ϵ can be determined based on the precision requirement of a specific application.

Real-world knowledge can be used to detect if there is a contact in a simulation environment. Since no solid object can penetrate into another solid object, if a "virtual" object is detected to be intersecting another one in a simulation (Fig. 1a), we know that a contact has occurred between the two objects. Similarly, during the simulation of a compliant motion, a "virtual" object may be moved (due to computation error) out of contact (Fig. 1b) from the constraining surface, but we know that the object should really be in contact with the surface. Note that in those cases, a necessary step is to adjust the relative location of the "virtual" objects so that the above definition of contact is satisfied.

There is little research on deriving topological contact states from geometric models. Bouma *et. al* [2] presented an algorithm using set-theoretic intersection between contacting polyhedra to find the contact state. The precondition for performing set-theoretic intersection is that the contacting polyhedra should penetrate into each other within some tolerance. In other words, they in fact modified the Definition 1 by requiring that the distance between objects be in the range $[-\epsilon, -\epsilon_h]$, where $0 < \epsilon_h < \epsilon$. Their algorithm

*This research is supported by a National Science Foundation grant under grant number IRI-9210423.

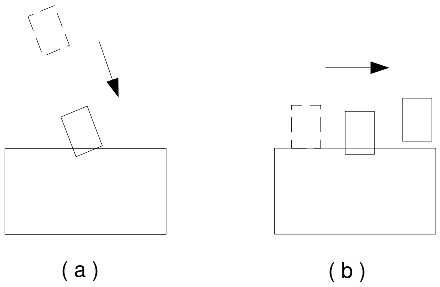

Figure 1: Contact detection in computer simulations.

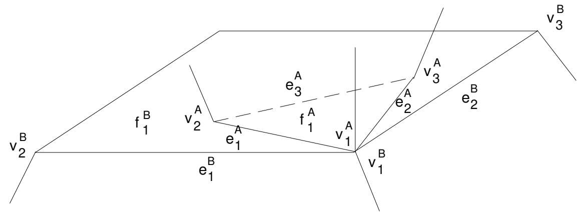

Figure 2: An example of elemental contacts (ECs)

seemed slow due to computation of set-theoretic intersection and complex analysis of the intersection regions obtained.

In this paper, we describe a different algorithm. Our algorithm is based on the algorithm for computing the distance between two polytopes developed by Gilbert *et. al* [3]. Thus, it also requires a modification of Definition 1 but in the opposite direction. For convenience, we present our modification as the following new definition.

Definition 2 An alternative definition of "contact": *In computer modeling or simulation, two objects are* in contact *iff their distance is within an interval* $[\epsilon_l, \epsilon]$, *where* ϵ_l *and* ϵ *are very small positive numbers.*

Note that with this modification, for two *intersecting* objects satisfying Definition 1, a slight adjustment of object locations are required to make them satisfy this modified definition. Based on the output of Gilbert's polytope distance algorithm, our algorithm builds up a local contact hierarchy graph and searches the graph to determine the topological contact state between the two polytopes. Our algorithm is extremely efficient because of the fast polytope distance algorithm and its highly localized output. Currently, our implementation of the algorithm applies to convex polyhedra only. But since non-convex polyhedra can be decomposed into convex ones, it is possible to extend the algorithm to non-convex polyhedra.

The paper is organized as the following. In Section 2, we introduce the contact models for convex polytopes. In Section 3, we present how to find an existing elemental contact (EC) from the result of the polytope distance algorithm. In Section 4, we describe how to search a contact hierarchy graph starting from this EC for the correct topological contact state. In Section 5, we analyze the time complexity of the algorithm. In Section 6, we discuss the implementation of this algorithm. We conclude this paper in Section 7.

2 Contact Models for Convex Polytopes

In this section, we introduce our contact models for convex polytopes. We first define the meanings of topological elements and then introduce the concepts of *elemental contact* (EC), *containing relationship* between two ECs, and *principal contact* (PC).

We define the faces and edges of polytopes as open point sets, i.e., a face is a finite surface without its boundary; an edge is an open line segment between its bounding vertices. The contact between two topological elements is defined as following,

Definition 3 *Given two convex polytopes A and B in contact following Definition 2, the topological elements* u_i^A *of A and* u_j^B *of B are* in contact *iff there exist points* p^A *and* p^B *which satisfy*

$$p^A \in u_i^A,\ p^B \in u_j^B,\ \|p^A - p^B\| \in [\epsilon_l, \epsilon]$$

p^A *and* p^B *are* overlapping *points.*

In [1], Desai first introduced the notion of elemental contacts. For convenience, we formally describe the concept again in the context of our definition of 'contact' above.

Definition 4 *For two contacting convex polytopes, an* elemental contact *(EC) is the contact between a pair of their topological elements. In addition, the face-face EC, edge-face EC, face-edge EC and edge-edge-touch EC require that the two contacting topological elements be parallel* [1]*; the edge-edge-cross EC requires that the two contacting edges be non-parallel.*

An EC between topological elements u_i^A and u_j^B is denoted as u_i^A-u_j^B EC. An edge-edge-touch EC between the edges e_i^A and e_j^B is denoted as e_i^A-e_j^B-touch EC.

[1] In this paper, the concept of *parallel* should also be understood as within some tolerance.

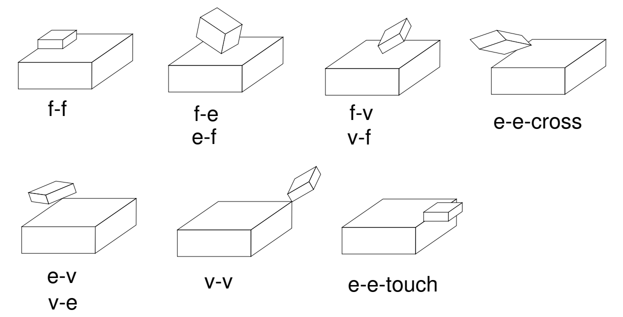

Figure 3: Principal contacts (PCs) between two convex polytopes

An edge-edge-cross EC between the edges e_i^A and e_j^B is denoted as e_i^A-e_j^B-cross EC.

We now define the *(directly) containing* relationship between two ECs.

Definition 5 *For two convex polytopes A and B in contact, u_i^A-u_j^B EC* contains *u_m^A-u_n^B EC iff one of the following three conditions is satisfied.*

1. *u_m^A is on the boundary of u_i^A and u_n^B is on the boundary of u_j^B.*
2. *u_m^A is on the boundary of u_i^A and u_n^B is u_j^B.*
3. *u_m^A is u_i^A and u_n^B is on the boundary of u_j^B.*

Definition 6 *For two contacting convex polytopes, EC_i* directly contains *EC_j iff*

- *EC_i contains EC_j, and*
- *if another EC_k also contains EC_j, EC_k is not contained by EC_i.*

As shown in Fig.2, for example, e_1^A-f_1^B EC directly contains v_2^A-f_1^B EC, and f_1^A-f_1^B EC directly contains e_2^A-f_1^B EC. However, f_1^A-f_1^B EC contains v_1^A-v_1^B EC but not directly contains it.

There are many ECs between two contacting *convex* polytopes, but only one of them captures the nature of the contact state. For instance, in Fig.2, it is the f_1^A-f_1^B EC which captures the fact that a face is contacting another face. Thus, we define the concept of *principal contact*[2] below, in terms following the previous definitions.

Definition 7 *For two contacting convex polytopes, the* principal contact *(PC) is an EC that contains all the other ECs.*

[2] The concept was first introduced in [9].

There are 7 types of PCs as shown in Fig.3. Note that there exists only one PC between two contacting convex polytopes at one time. Our algorithm is to find the PC between two contacting convex polytopes.

3 Extension of the Polytope Distance Algorithm

In this section, we first extend the polytope distance algorithm developed by Gilbert *et. al* [3] by Theorem 1 to derive the topological information about the two nearest points of the two contacting convex polytopes. Then, we describe how to find an EC from such topological information.

Theorem 1 *For two contacting convex polytopes A and B, computing their distance by the polytope distance algorithm in [3] gives the expressions of the two nearest points $p^A \in A$ and $p^B \in B$,*

$$p^A = \sum_{i=1}^{m} \lambda_i v_i^A, \quad \sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i > 0, \quad 1 \le m \le 3 \qquad (1)$$

$$p^B = \sum_{j=1}^{n} \lambda_j v_j^B, \quad \sum_{j=1}^{n} \lambda_j = 1, \quad \lambda_j > 0, \quad 1 \le n \le 3 \qquad (2)$$

where $v_i^A, i = 1, \cdots, m$ are different vertices of A and $v_j^B, j = 1, \cdots, n$ are different vertices of B. If $m = 3$, $v_i^A, i = 1, 2, 3$ are vertices bounding one face of A. If $m = 2$, $v_i^A, i = 1, 2$ are either vertices bounding one edge of A or vertices bounding a diagonal of one face of A. Similar discussions also apply to the vertices of B.

Proof: According to [3], Computing the distance between A and B by the polytope distance algorithm also gives the following expressions of their nearest points,

$$p^A = \sum_{k=1}^{l} \lambda_k v_k^A, \quad p^B = \sum_{k=1}^{l} \lambda_k v_k^B \qquad (3)$$

$$\sum_{k=1}^{l} \lambda_k = 1, \quad \lambda_k > 0, \quad 1 \le l \le 4 \qquad (4)$$

where v_k^A and v_k^B are vertices of A and B respectively. Since the distance between A and B is more than ϵ_l (> 0)[3], according to [3], l should not be more than 3. In Equation (3), v_k^A, $k = 1, \cdots, l$, may not all be different

[3] This is why in Definition 2, we demand a positive lower bound of the distance between two contacting objects.

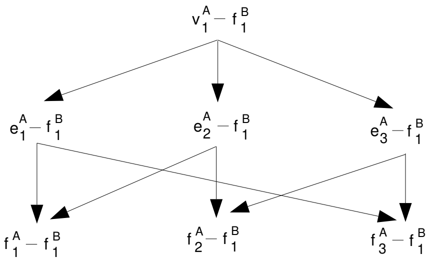

Figure 4: The CHG starting from v_1^A-f_1^B EC$_o$

vertices of A, and likewise, v_k^B, $k = 1, \cdots, l$, may not all be different vertices of B. Combining the terms of the same vertices in Equation (3) gives Equations (1) and (2). v_i^A and v_j^B should satisfy the conditions specified in this theorem. ▮

An EC between of the contacting convex polytopes A and B can be found from the Equations (1) and (2). , for example, if $m = n = 3$, we know that p^A and p^B are inner points of two faces ($\lambda_i > 0$, $\lambda_j > 0$). Let f_i^A and f_j^B denote these two faces. f_i^A and f_j^B should be parallel (If they are not, the closest points p^A or p^B can not be the inner points of faces). In addition, the distance between p^A and p^B, which is also the distance between A and B, is within the range $[\epsilon_l, \epsilon]$. Thus, according to Definition 4, we find an f_i^A-f_j^B EC. Similarly, for each other combination of values of m and n, we can also find a corresponding EC. Note that when $m = 2$ and the two vertices in Equation (1) bounds a diagonal of one face, p^A is an inner point of a face instead of that of an edge.

We use EC$_o$ to denote the EC found according to above analysis. The EC$_o$ may not be *the* PC. For example, in Fig. 2, there is a face-face PC between A and B. The polytope distance algorithm [3] may indicate that the vertices v_1^A and v_1^B are the nearest. Thus, by Theorem 1, we will find a v_1^A-v_1^B EC$_o$.

In the next section, we discuss how to find the PC from the EC$_o$.

4 The Contact Hierarchy Graphs

According to Definition 7, for two contacting convex polytopes, the PC is an EC, and it contains all the other ECs. To find the PC, we first hypothesize all the possible ECs that contain EC$_o$. Clearly, the PC should be either EC$_o$ or one of the hypothesized

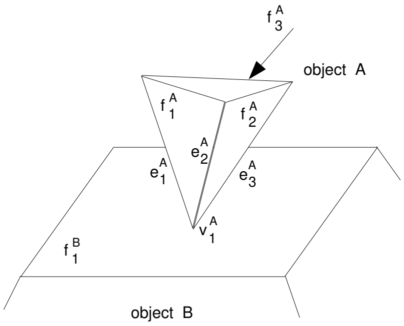

Figure 5: The topological elements in $\mathcal{T}^A$ and $\mathcal{T}^B$ for v_1^A-f_1^B EC$_o$.

ECs (HECs)[4]. Our strategy of finding the PC is to use a *contact hierarchy graph* to organize the HECs that contain EC$_o$ and search the PC in the graph based on the characteristics of the graph.

First of all, we describe how to find the set $\mathcal{H}$ of all possible HECs containing an EC$_o$. Let EC$_o$ be u_o^A-u_o^B. Let the set $\mathcal{T}^A$ ($\mathcal{T}^B$) consists of u_o^A (u_o^B) and all the topological elements of object A (B) which contains u_o^A (u_o^B) on their boundaries. Based on Definition 5 and the following facts:

- an edge-edge-touch EC can contain vertex-edge, edge-vertex or vertex-vertex ECs, but not an edge-edge-cross EC, and
- a face-edge EC or an edge-face EC cannot contain an edge-edge-touch EC,

we can use the following strategy to obtain $\mathcal{H}$.

Strategy 1 *Construct $\mathcal{H}$.*

1. $\mathcal{S} = \mathcal{T}^A \times \mathcal{T}^B$, $\mathcal{H} = \emptyset$.
2. *Take an element* $(u_i^A,\ u_j^B)$ *out of* $\mathcal{S}$.
   (a) *If* $(u_i^A,\ u_j^B)$ *is* $(e_i^A,\ e_j^B)$ *then*
   If EC$_o$ is an edge-edge-cross EC,
   form an e_i^A-e_j^B-cross EC$_i$.
   Otherwise, form an e_i^A-e_j^B-touch EC$_i$.
   (b) *If* $(u_i^A,\ u_j^B)$ *is* $(f_i^A,\ e_j^B)$ *or* $(e_i^A,\ f_j^B)$ *and EC$_o$ is edge-edge-touch EC, goto Step 3.*
   (c) *Otherwise, form a u_i^A-u_j^B EC$_i$.*

   Add EC$_i$ to $\mathcal{H}$.
3. *If $\mathcal{S} = \emptyset$, report $\mathcal{H}$, stop. Otherwise, goto step 2.*

[4]Note that an HEC may not be an EC, because the pair of topological elements of an HEC may not be actually in contact.

Now we define a contact hierarchy graph to organize the HECs in $\mathcal{H}$. Note that from now on, we also extend the concept of the *(directly) containing* relationship (Definition 7) to HECs.

Definition 8 *A* Contact Hierarchy Graph *(CHG) is a directed graph. It has a single source node denoting EC_o. A node of CHG denotes an HEC in $\mathcal{H}$. An arc of CHG denotes the direct containing relationship between HECs, i.e., an arc from HEC_i to HEC_j denotes that HEC_j directly contains HEC_i. A CHG has no cycles.*

Note that since there are seven types of ECs (or HECs), there are seven types of CHGs, each of which contains a finite number of nodes. For each type of EC_o and the corresponding $\mathcal{H}$, we simply build the CHG manually. As an example, Fig.4 shows the CHG starting from a v_1^A-f_1^B EC_o, where the labels of the topological elements are shown in Fig.5. The other types of CHGs can be found in [10].

We now discuss the characteristics of CHGs by answering the following questions in a few theorems before describing how to search for the PC in a CHG.

1. Is it expensive to test if an HEC is an EC?

Theorem 2 *It needs constant time to test if an HEC is an EC.*

Proof: Inspecting all types of CHGs gives this theorem (See [10]). We use the CHG shown in Fig.4 as an example. Given a v_1^A-f_1^B EC_o, to test if an edge-face HEC or a face-face HEC is an EC, according to Definition 4, we only need to test if the contacting topological elements are parallel. This needs constant time. ▮

2. Which HEC in a CHG is the PC?

Theorem 3 *A node of the CHG is the PC, iff it is an EC and no node in a path starting from this node is an EC.*

Proof: This theorem can be proven according to the Definition 7 and the fact that the CHG is used to represent the containing relationship between HECs. The detailed proof is given in [10].▮

3. Do CHGs have some characteristics, based on which we can develop an efficient search algorithm?

Theorem 4 *In a CHG, there exists at least one path between an EC and the PC satisfying that all nodes on this path are ECs.*

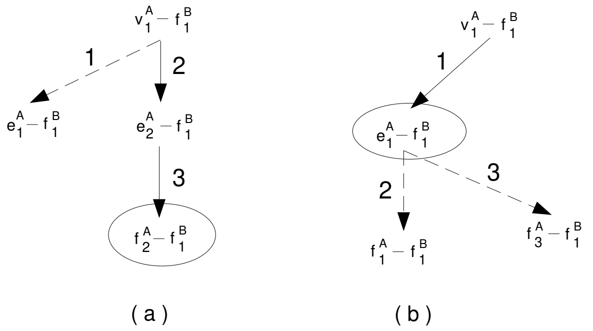

Figure 6: Some search trees created

Proof: The proof is given in [10].▮

Corollary 1 *A node of a CHG is the PC, iff it is an EC and no node directly connected by an arc starting from this node is an EC.*

Proof: This corollary can be obtained from Theorem 3 and 4. The detailed proof is given in [10]. ▮.

According to the above theorems, we can design a strategy to search a CHG for the PC. Let EC_d denote the current EC found by the search in the CHG. HEC_i, $i = 1, \cdots, n$ denote the nodes connected by an arc starting from EC_d. The algorithm is listed below.

Strategy 2 *Search a CHG for the PC*

1. *$EC_d = EC_o$.*
2. *Find HEC_i, $i = 1, \cdots, n$.*
3. *For i = 1 To n Do*
   If HEC_i is an EC, set $EC_d = EC_i$, goto step 2.
4. *Report EC_d as the PC, stop.*

Fig. 6 shows two typical search trees created by Strategy 2 on the CHG shown in Fig. 4. In Fig. 6, the HEC pointed by a dashed arc is not an EC. The HEC pointed by a solid arc is an EC. The PC is circled in the figure. The order of the search is indexed by the numbers at the edges.

5 Time Complexity

From the analyses in Sections 3 and 4, the algorithm for identifying the PC between contacting convex polytopes A and B consists of the following three steps.

1. Compute the distance between A and B by the polytope distance algorithm [3].

2. Find EC_o by the method described in Section 3.

3. Search for the PC in the CHG starting from the EC_o by the method described in Section 4.

Now we analyze the time complexity of each step of the algorithm. Let N be the total number of vertices of A and B, N_1 be the maximum number of faces and edges of A and B, Also, for simplicity, we assume that A and B have same number of edges intersecting at each vertex. Let $n_{e/v}$ denote this number. According to [3], the time complexity of the first step is $O(N)$. In the second step, it needs $O(N_1)$ time to find the contacting topological elements by searching the boundary representations of A and B. According to [4], $N_1 < 3N$. Thus the time complexity of the second step is also $O(N)$. In the third step, the most complex CHG is the CHG starting from a vertex-vertex EC, which has $4n_{e/v}^2$ nodes (See [10]). By the search strategy presented in Section 4, in the worse case, it needs $O(n_{e/v}^2)$ time to find the PC. Summarizing the above analyses, we see that the worst case time complexity of our algorithm is $O(N + n_{e/v}^2)$.

6 Implementation

The algorithm is implemented as a part of the simulation system for a contact based fine motion planning scheme [7]. In this system, an edge of a polytope has a length typically in the range of [10 mm, 200 mm]. The ϵ_l and ϵ are chosen to be 0.9×10^{-7} mm and 1.0×10^{-7} mm, respectively. We have done a lot of experiments using this system. In all cases, the algorithm can identify the contact state quickly and successfully.

7 Conclusions

We have introduced an efficient algorithm for deriving the topological contact state from the geometric models of two contacting objects in a simulation environment, taking into account the effect of computation errors. The algorithm applies to convex polytopes but can be extended to nonconvex polytopes in a straightforward way. This algorithm can be used in robot simulation tasks and offline motion planning systems which need to deal with "virtual" objects in contact (as introduced in Section 1).

It should be emphasized, however, that the object models are assumed to be precise in the problem addressed in this paper. Thus, the current algorithm cannot be readily used in situations where the objects have location uncertainties and/or modeling uncertainties. Research on contact recognition in the presence of such uncertainties is surveyed in [8].

References

[1] Desai, R., *On Fine Motion in Mechanical Assembly in Presence of Uncertainty*, Ph.D thesis, University of Michigan, 1989.

[2] Bouma, W. and Vaněček Jr., G., "Modeling Contacts in a Physically Based Simulation," Proc. 2nd Symp. Solid Modeling and Applications, pp. 409-418, Montreal, Canada, May 1993.

[3] Gilbert, E., Johnson, D. and Keerthi, S., " A Fast Procedure for Computing the Distance between Complex Objects in Three-dimensional Space," *IEEE J. of R & A,* Vol. 4, No. 2, pp. 193-203, April 1988.

[4] Grönbaum, B., *Convex Polytopes*, John Wiley & Sons,1967.

[5] Hirukawa, H., Papegay, Y. and Matsui, T., "A Motion Planning Algorithm for Convex Polyhedra in Contact under Translation and Rotation," Proc. IEEE ICRA, pp. 3020-3027, May 1994.

[6] Ong, C. J., *Penetration Distance and Their Applications to Path Planning,* Ph.D thesis, the University of Michigan, 1993.

[7] Xiao, J., and Zhang, L., "A Geometric Simulator SimRep for Testing the Replanning Approach toward Assembly Motions in the Presence of Uncertainties," to appear in the IEEE Int. Symp. Assembly and Task Planning, Pittsburgh, Aug. 1995.

[8] Xiao, J., and Zhang, L., "An Efficient Algorithm (FAPRIC) for Finding the Principal Contacts Possibly Established due to Uncertainties," Proc. IEEE ICRA, Nagoya, Japan, May 1995.

[9] Xiao, J., "Automatic Determination of Topological Contacts in the Presence of Sensing Uncertainties," Proc. IEEE ICRA, pp. 65-70, Atlanta, May 1993.

[10] Zhang, L. and Xiao, J., "Derivation of Contact States from Geometric Models of Objects," Tech. Report, Dept. of Computer Science, UNCC, 1995.