Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Automatic Generation of High-level Contact State Space between Planar Curved Objects *

Peng Tang
IMI Lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
ptang@uncc.edu

Jing Xiao
IMI Lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
xiao@uncc.edu

***Abstract*— Information of high-level, topological contact states is useful and even necessary for a wide range of applications, including many robotics applications. While there is considerable research related to topological contact states between two polyhedral objects, little is studied about how to characterize, represent, and automatically generate topological contact states between curved objects. In this paper we extend the representation of topological contact states between polyhedral objects to general planar curved objects in terms of contacting curve elements, obtained from curvature monotonic segmentation [6]. We further introduce an approach to generate automatically graphs of such contact states between two planar curved objects, which represent not only valid contact states but also adjacency relations among those contact states. Implementation results of the related algorithms demonstrate the effectiveness of our approach. The approach can be naturally extended to generation of contact states between 3-D curved objects.**

***Index Terms*— contact states, planar curved objects, automatic generation, compliant motion**

I. Introduction

When a contact occurs between two objects, it is often desirable to know not only the precise contact configuration but also the high-level, discrete contact state that is more descriptive of the topological and physical contact characteristics. Information of such high-level contact states is useful and even necessary for a wide range of applications, from real-world robotic tasks involving compliant motion [4], [8], to dynamic simulation [9] and haptic interaction in a virtual world [7].

For contacting polyhedral objects, it is common to describe a contact state as a set of contact primitives. Each contact primitive defines a single connected region of contact and can be naturally characterized by a pair of contacting surface elements in terms of faces, edges, and vertices, because such surface elements are convex for polyhedral objects and there can only be one connected region of contact between two such elements. Different contact state representations essentially differ only in how contact primitives are determined with those surface elements; some define a contact primitive as a point contact in terms of a vertex and a face in contact [5], [2], while others allow a contact primitive to be a line or planar contact region as well [1], [10].

In order to describe contact states between two non-polyhedral objects in a way analogous to that between polyhedral objects, however, it is necessary to further partition the boundary of each non-polyhedral object to obtain surface elements of useful properties in addition to the partition by the natural edges and vertices that indicate derivative discontinuous points [6]. Specifically, given a general curved object, its boundary consists of smooth surfaces (which includes flat surfaces as a special case), smooth curves (which includes straight-line segments as a special case), and vertices, which are first-derivative discontinuous points separating two smooth curves. Since a smooth surface or curve can be non-convex generally, a contact between two such surfaces or curves may consist of more than one disjoint region of contact. Thus, such smooth surfaces and curves cannot be used directly to define contact primitives. In [6], it shows that by doing a curvature monotonic segmentation to the boundary of a curved object, useful topological surface elements can be obtained, which are in terms of smooth surface patches or curve segments with monotonically changing or constant curvatures, separated by curvature extremal points or inflection points of a surface, in addition to vertices. A contact primitive can again be defined as a contact between two such elements, which forms a single connected region of contact. A topological contact state between two curved objects can thus be represented again as a set of contact primitives.

In this paper we extend the representation of topological contact states between polyhedral objects to general planar curved objects in terms of contacting curve elements, which are obtained from curvature monotonic segmentation [6]. We further introduce an approach to generate automatically graphs of such contact states between two planar curved objects, which represent not only valid contact states but also adjacency relations among those contact states. In Section II, we review the notions of *principal contacts* as contact primitives [10], [6] to define contact states for planar curved objects and to characterize the neighboring relations between contact states. In Section III, we describe our approach and algorithms to generate contact state graphs automatically, and in Section IV, we present some

*This work is supported by the U.S. National Science Foundation under grant IIS-#0328782.

implementation results. We conclude the paper in Section V.

II. Contact States between Planar Curved Objects

A. Topological Curve Elements

Given a planar curved object, its boundary consists of smooth curves and vertices, which are first-derivative discontinuous points separating two smooth curves. By performing a curvature monotonic segmentation [6], each smooth curve can be further partitioned into curve segments of monotonic curvature, which we call *edges*, by inflection or extreme points on the smooth curve, which we call *pseudo-vertices*. Clearly an edge is between either vertices or pseudo-vertices or one vertex and one pseudo-vertex. We call these edges, vertices, and pseudo-vertices the *topological curve elements* of a planar curved object.

We further define the following *element containment relation* among those elements: for a planar curved object, a pseudo-vertex *contains* the edges that are adjacent to it, an edge *contains* the vertices that are adjacent to it.

Note that the above containment relation is similar to that defined among surface elements, i.e., faces, edges, and vertices, of a polyhedral object [11]: a face contains its bounding edges, and an edge contains its bounding vertices. The usefulness of the containment relation will be evident in the following subsections.

B. Principal Contacts and Contact Formations for Planar Curved Objects

A *principal contact* (PC) between two contacting polyhedra is defined as the contact between a pair of contacting surface elements (i.e., faces, edges, and vertices) that are not contained by other contacting surface elements [10], [11]. This ensures that PCs are the highest-level contact primitives to describe a contact state most concisely [11]. For planar curved objects, we can extend the notion of PCs in terms of the topological curve elements introduced above: a PC between two planar curved objects is the contact between a pair of topological curve elements that are not contained by other contacting topological curve elements of the two objects. Denote a PC between two planar curved objects A and B as a_A-b_B, where a_A is the contacting curve element of A, and b_B is the contacting curve element of B. Let v, pv, and e denote a vertex, a pseudo vertex, and an edge respectively. Figure 1 shows the different types of PCs between two planar curved objects.

Now a general contact state between two planar curved objects can be defined as a set of PCs formed, called a *contact formation* (CF), similar to that defined for polyhedral objects. The *cardinality of a CF*, denoted as $card$(CF), is the number of PCs in the CF.

The *geometrical representation* of a PC denotes the set of (relative) contact configurations between the two contacting objects that satisfy the topological definition of the PC. The *geometrical representation* of a CF denotes the set of contact configurations that satisfy the contact conditions of all the PCs in the CF.

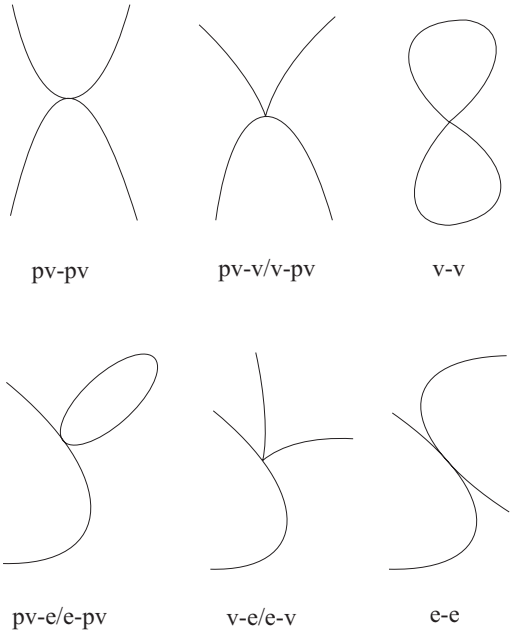


Fig. 1. Different types of principal contacts between two planar curved objects

C. Contact States and Connectivity

With the concepts of PCs and CFs extended to planar curved objects, we can now define contact state space between such objects in a way similar to that between polyhedral objects [11].

Generally, the set of contact configurations in the geometrical representation of a contact formation may consist of one or more connected regions of contact configurations, called *CF-connected* regions. Within a CF-connected region, there exists a motion constrained by the CF from any contact configuration to any other one, called *CF-compliant* motion. In other words, there is no need to change the CF in moving from one configuration to another within a CF-connected region. Thus, we can define a *contact state* between two curved objects as a single CF-connected region, represented by the CF and a representative configuration in the region, denoted as a pair $<CF, C>$.

In this paper, we only focus on cases where the geometrical representation of a CF is a single CF-connected region.

Now we consider connectivity between contact states of different CFs. If from the state $<CF_i, C_i>$, there exists a CF_i-compliant motion succeeded by a transition to a configuration C_j of CF_j, then $<CF_i, C_i>$ and $<CF_j, C_j>$ are *generally-defined neighboring contact states*, and CF_i and CF_j are called *generally-defined neighboring contact formations*. The above compliant motion making the transition from $<CF_i, C_i>$ to $<CF_j, C_j>$ is called a *neighboring transition motion*.

The above configuration-based definition of neighboring CFs can be mapped to necessary topological conditions in terms of the *containment relations of PCs*. Given two different PCs between A and B: $PC_i = i_A$-i_B and $PC_j = j_A$-

j_B, we say PC_i *contains* PC_j if and only if one of the following conditions holds:

1) i_A contains j_A, and i_B contains j_B;
2) i_A is j_A, and i_B contains j_B;
3) i_B is j_B, and i_A contains j_A.

Neighboring PCs can now be defined. Two PCs are *neighboring PCs* if one contains another.

Note that based on the above definition, a PC is contained by a more constrained neighboring PC, where the contacting objects have fewer degrees of freedom. Recall that we define that a pseudo-vertex *contains* the adjacent edges in the element containment relation in Section II.A, because a PC involving a pseudo-vertex is more constrained than a neighboring PC that does not involve a pseudo-vertex.

Next we can define the necessary conditions for two kinds of neighboring relations among CFs. Given two generally-defined neighboring CFs, CF_i and CF_j, CF_j is a *locally-defined neighbor* (LN) of CF_i if

1) $card(CF_j) \leq card(CF_i)$, and
2) one of the following two conditions holds (not both):
   - 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.
   - For every PC in CF_j, it either belongs to CF_i or contains a unique PC in CF_i.

Moreover, if $card(CF_j) > card(CF_i)$, then CF_j is a *globally-defined neighbor* (GN) of CF_i.

The reason that we differentiate neighboring CFs into LNs and GNs is that given a CF, the topological information of its LNs can be derived directly from its topological definition, that is, from the PCs in the CF, one can obtain the possible PCs of the LNs of the CF. This is a very useful property for automatic generation of contact states (see next section). Figure 2 shows an example, where a's and b's label the pseudo or real vertices of objects A and B respectively, and an edge is labeled by its two pseudo or real vertices. CF_1 is a LN of CF_2, and its topological representation can be obtained from CF_2; however, one cannot obtain the topological representation of CF_2, which is a GN of CF_1, directly from the topological representation of CF_1. This is because CF_2 involves a PC not described by the neighboring elements of the contacting curve elements of CF_1.

Given two generally-defined neighboring contact states, $<CF_j, C_j>$ and $<CF_i, C_i>$, if CF_j is a LN of CF_i, then $<CF_j, C_j>$ is a *locally-defined neighboring contact state* or LN contact state of $<CF_i, C_i>$. If CF_j is a GN of CF_i, then $<CF_j, C_j>$ is a *globally-defined neighboring contact state* or GN contact state of $<CF_i, C_i>$.

The contact state space (of the contacting objects) can be defined as a contact state graph $\mathcal{G}$, where each node denotes a valid contact state $<CF, C>$, and each link connects two neighboring contact states.

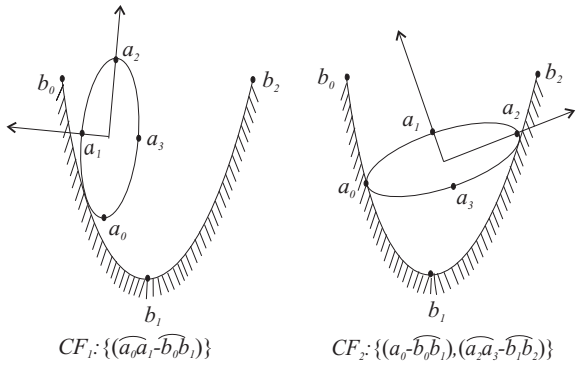


Fig. 2. An example of LN and GN CFs

III. Generation of Contact State Graphs

Our approach to generate the contact state graph between two planar curved objects is to generate special subgraphs of the contact state graph $\mathcal{G}$ automatically and merge such subgraphs automatically to form $\mathcal{G}$.

Each special subgraph we generate is an undirected graph consisting of a seed contact state $<CF_s, C_s>$, its LN contact states, their subsequent LN contact states, and so on, which we call a *LN graph* of $<CF_s, C_s>$. Starting from the seed contact state $<CF_s, C_s>$, the LN graph can be grown by repeatedly obtaining LN contact states until all the LN contact states have been generated in a breadth-first search as follows:

1) Initialize the LN graph LNG with a single node containing $<CF_s, C_s>$. Initialize the queue NEW with a single node $<CF_s, C_s>$.
2) Remove the front node $<CF_i, C_i>$ from the queue NEW.
3) Hypothesize topologically all possible LN CFs of CF_i.
4) For each possible LN CF, CF_j, of CF_i, do:
   - If LNG has a node of $<CF_j, C_j>$ and there is no link between $<CF_j, C_j>$ and $<CF_i, C_i>$, then if there exists a feasible neighboring transition motion from C_i in CF_i to C_j in CF_j, then build a link between the node of $<CF_j, C_j>$ and the node of $<CF_i, C_i>$ in LNG.
   - If LNG does not have a node of $<CF_j, C_j>$, then if there exists a feasible neighboring transition motion from C_i in CF_i to a configuration C_j in CF_j, then create a node for $<CF_j, C_j>$, link it to the node of $<CF_i, C_i>$ in LNG, and append it to the end of queue NEW.
5) If the queue NEW is not empty, go to step 2.
6) Output LNG.

Clearly we prefer that a seed CF, CF_s, of a LN graph has a maximized cardinality (i.e. the number of PCs) so that the LN graph can be maximized. Among all neighboring CFs between two objects, those with cardinalities being local maxima could be chosen as seed CFs. There are very few such CFs in many practical situations, and they can be singled out relatively easily to be used as seed CFs.

After all LN graphs are generated, they can be merged automatically by taking the union of the nodes in all the LN graphs and their neighboring links (connections). The same merge algorithm used in [3] can be used here. For nodes in different LN graphs but sharing the same CF, they are collapsed into one node with a single contact configuration kept, and corresponding redundant links connecting two neighboring states are also collapsed into one.

In the following subsections, we explain our algorithm in more detail.

A. Hypothesizing Locally-defined Neighboring Contact Formations

Given a valid CF = $\{\text{PC}_i\}_{i=1}^{N}$, where $N \geq 1$, its possible LN CFs can be hypothesized according to the definitions in Section II by applying one of the following sets of actions:

- Action set 1: for each PC_i, either **remove** PC_i, or **keep** PC_i, or **change** PC_i to a PC it contains.
- Action set 2: for each PC_i, either **remove** PC_i, or **keep** PC_i, or **change** PC_i to a PC that contains it.

Note that the two sets of actions enforce that only one type of change can be applied simultaneously to more than one PC in the CF. Note also that in both set of actions, no **remove** nor **keep** can be applied simultaneously to all PCs in the CF to result in an empty set or the CF itself[1].

B. Neighboring Transition Motions

There are three possible types of a neighboring transition motion between two neighboring contact states of two smoothly curved objects A and B: **slidingA**, **slidingB**, or a **combination** of **slidingA** and **slidingB**. None of these motions are of pure translation or rotation. Figure 3 shows examples of these different types of motions of a planar curved object A with respect to a fixed planar curved object B: note that during **slidingA** the contact point of A does not change but the contact point of B changes, whereas during **slidingB**, the contact point of A changes but the contact point of B does not change, and during a **combination** motion, the contact points of A and B both change; note that if the contact points of A and B change in equal displacements, the motion is in fact a rolling motion; otherwise, the motion is a combined sliding and rolling motion.

Of course, for two general planar curved objects that also have non-smooth elements, such as vertices, or certain "polygonal" elements, such as straight-line edges, pure rotation and pure translation are possible as neighboring transition motions involving those elements, just as in the case of polygonal objects [3]. Here we focus only on the three types of motions – **slidingA**, **slidingB**, and **combination** of **slidingA** and **slidingB**, which are unique to curved objects.

Each type of motion can be generally viewed as an integral of instantaneous pure (normally 2-D) translation ds combined with an instantaneous pure (1-D) rotation $d\theta$,

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

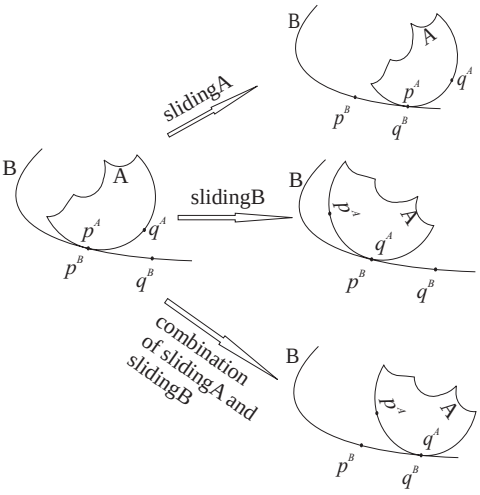


Fig. 3. Three types of neighboring transition motions

and the results can be implemented by a summation of small motions, and each small motion is implemented as a small translation combined with a small rotation $\Delta s + \Delta\theta$.

Specifically, considering A and B initially in contact at points p^A of A and p^B of B, a small slidingA, slidingB, or combination of slidingA and slidingB can be implemented as follows:

- Small **slidingA** of A to reach a nearby point p_1^B of B: Δs is the distance from p^B to p_1^B, $\Delta\theta$ is the angle between the tangent of A at p^A and the tangent of B at p_1^B, and the rotation is about p^A.
- Small **slidingB** of A to change the contact point of A to a nearby point p_1^A of A: Δs is the distance from p^A to p_1^A, $\Delta\theta$ is the angle between the tangent of A at p_1^A and the tangent of B at p^B, and the rotation is about p^B.
- Small **combination** motion of A so that a nearby p_1^A of A contacts a nearby p_1^B of B: a **slidingA** of A to reach p_1^B of B followed by a **slidingB** of A to change the contact point of A to p_1^A with the contact point of B being p_1^B.

C. Checking the feasibility of Neighboring Transition Motions

Given a valid CF CF_i and a hypothesized possible LN CF CF_j of CF_i between two planar curved objects A and B, to check whether CF_j is a valid LN CF of CF_i or not means to determine whether there exists a feasible neighboring transition motion from CF_i to CF_j. There are just a finite number of possible compliant motions of A to achieve the neighboring transition starting from a contact configuration C_i in CF_i to reach certain contact configuration C_j in CF_j. Moreover, as mentioned in Section III.A, any neighboring transition may involve three types of actions or their combinations: **remove**, **keep**, or **change** one or more PCs of CF_i. Specifically, it is easy to see that

- if a neighboring transition involves **change** actions, it may or may not involve **keep** and/or **remove** actions;

- a neighboring transition may involve both **keep** and **remove** actions without **change** actions.

We now can present the following general strategy to construct a neighboring transition motion from CF_i to CF_j and check if it is feasible, i.e., to check if the compliant motion is implementable as a CF_i compliant motion followed by a CF_j compliant motion without causing other collisions along the way:

1. If the neighboring transition needs to change (at least) one PC, then construct a compliant motion of A in small steps (see Section III.B) to realize the **change** action. If the motion is possible in all small steps without causing additional collisions and is able to **keep** or **remove** some other PCs that the transition may also require, the motion is considered feasible, and the hypothesized CF_j is considered subsequently a valid LN CF of CF_i.

2. Otherwise, if the neighboring transition does not need to change any PC but needs to keep (at least) one PC, then construct a compliant motion of A in small steps to maintain the PC. If the motion is possible in all small steps without causing additional collisions and is able to **remove** some other PC(s) that the transition also require, the motion is considered feasible, and the hypothesized CF_j is considered subsequently a valid LN CF of CF_i.

If no possible motion is feasible in any case, CF_j is discarded as invalid.

The types of possible compliant motions to **change** a PC, PC_i, to a neighboring PC, PC_j, depends on the types of PC_i and PC_j. Since the change of PC_i to PC_j means the change of a contacting curve element (of either A or B) to an adjacent contacting curve element (of that object), the direction of motion is determined by the relative placement of the new contacting element with respect to the original contacting element as either clockwise or counter-clockwise. Moreover,

- if $PC_i = i_A$-i_B and $PC_j = j_A$-i_B, i.e., from PC_i to PC_j, the contacting curve element of A changes to an adjacent element, and the contacting curve element of B does not change, then a **slidingB** of A is needed;
- if $PC_i = i_A$-i_B and $PC_j = i_A$-j_B, i.e., from PC_i to PC_j, the contacting curve element of B changes to an adjacent element, and the contacting curve element of A does not change, then a **slidingA** of A is needed;
- if $PC_i = i_A$-i_B and $PC_j = j_A$-j_B, i.e., from PC_i to PC_j, both contacting curve elements of A and B change to respective adjacent elements, then a **combination** motion of A is needed.

The type of possible compliant motions to **keep** a PC also depends on the type of the PC:

- to keep a v-v, pv-v, or v-pv type of PC, the only motion is a pure rotation of A[2];
- to keep a v-e type of PC, the motion can be either a **slidingA** of A or a pure rotation of A about v;
- to keep a pv-e type of PC, the only motion is a **slidingA** of A;
- to keep an e-v type of PC, the motion can be either a **slidingB** of A or a pure rotation of A about v;
- to keep an e-pv type of PC, the only motion is a **slidingB** of A;
- to keep an e-e type of PC, the motion can be one of the three types of motions of A: **slidingA**, **slidingB**, or a **combination**.

The above motions can be in either direction (i.e., either clockwise or counter-clockwise).

IV. Implementation

We have implemented the general algorithms as described in Section III for automatic generation of a LN graph of a seed contact state on a Pentium 4, 2.8 GHz machine with 1024 MB RAM. The algorithm can handle arbitrary planar curved objects A and B. Figure 4 shows an example that our algorithm applied, where A is an ellipse and B is an object with a parabolic curve. a's and b's label the pseudo vertices of A and the pseudo and real vertices on B's parabolic curve. An edge of A or B is labeled by its two pseudo or real vertices. A seed contact state CS_s is shown in Figure 4b in the CF of type $\{pv\text{-}e, e\text{-}e\}$.

From CS_s, our algorithm has generated its LN graph consisting of 37 valid nodes automatically[3] in about 2 seconds. The number of valid nodes shows that not all hypothesized LN contact states are valid. Figure 5 displays the valid LN contact states of CS_s generated as well as several subsequent LN contact states. Note that one of the hypothesized LN CF of CF_s, CF_0 = $\{(a_0\text{-}\widehat{b_0b_1}), (a_2\text{-}\widehat{b_1b_2})\}$, of type $\{pv\text{-}e, pv\text{-}e\}$, is detected by the algorithm as not a valid LN CF since there is no feasible neighboring transition motion from CF_s to CF_0. Our algorithm for feasibility checking is able to detect such a case even though it may not seem obvious intuitively. Indeed, the computation of intersections between an ellipse and a parabolic curve confirms that this result is correct. Figure 6 shows more examples of hypothesized contact states that are not valid.

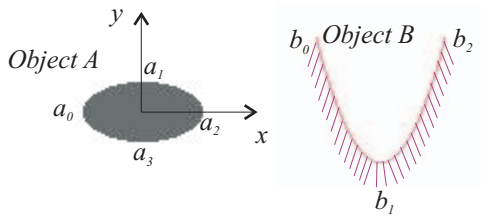
On the other hand, there is still a considerable number of nodes in the LN graph of CS_s, reflecting the fact that contact states are different if their CFs involve contacting curve elements of different labels. This is a general treatment suitable for contact states between arbitrary planar curved objects, although for the example shown, because of the symmetry of the particular A and B, several contact states look alike.

V. Conclusions

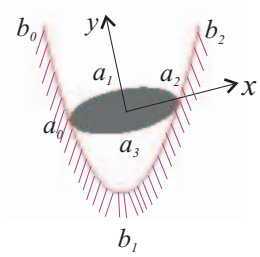
We have introduced a systematic approach to generate automatically the contact state graphs between two arbitrary planar curved objects based on a general representation of contact states between such objects that extends

[2]Note that to keep a pv-pv type of PC, no motion is allowed.

[3]Note that in this example we only considered the parabolic curve of object B rather than the whole boundary of B, which, if considered, will result in more valid nodes.



(a)



$CS_s = \{(a_0\text{-}\widehat{b_0 b_1}),\ (\widehat{a_2 a_3}\text{-}\widehat{b_1 b_2})\}$

(b)

Fig. 4. An implemented example

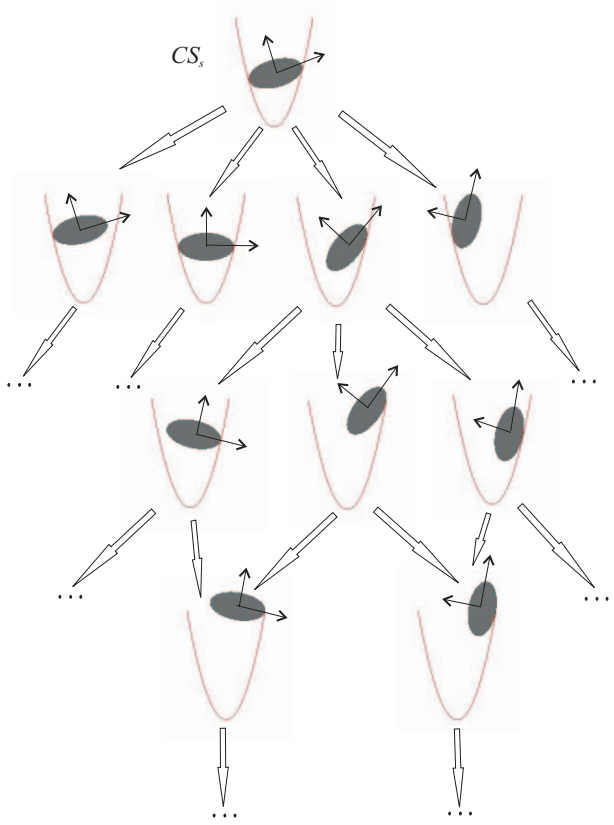
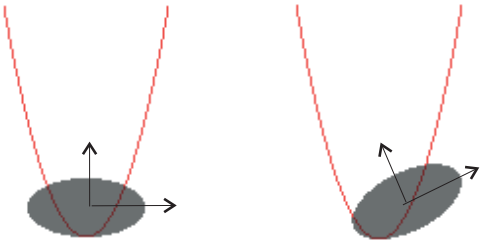


Fig. 5. Some LN contact states generated



(a) (b)

Fig. 6. Some invalid contact states

the notions of principal contacts and contact formations originally introduced for polyhedral objects. The approach is implemented, and the execution of our algorithm shows that automatic generation of contact state graphs between two planar curved objects is even more desirable than that in the case of polyhedral objects not only because of the large number of valid contact states but also because that it is often not obvious to the naked eyes whether a contact state is actually possible or not.

We plan to continue this research by improving and testing the algorithm on more sophisticated cases, including cases where a CF may have multiple connected regions of contact points, and by extending the approach significantly to handle contact states between 3-D curved objects.

ACKNOWLEDGEMENT

The authors would like to thank Qi Luo for his help.

REFERENCES

[1] R. Desai and J. Xiao and R. Volz, "Contact Formations and Design Constraints: A New Basis for the Automatic Generation of Robot Programs," *NATO Advanced Research Workshop: CAD Based Programming for Sensor Based Robots*, B. Ravani, Ed., pp. 361-395, July 1988.

[2] B. Donald, "On Motion Planning with Six Degrees of Freedoms: Solving the Intersection Problems in Configuration Space," *IEEE Int. Conf. Robotics & Automation (ICRA)*, 1985.

[3] X. Ji and J. Xiao, "Automatic Generation of High-Level Contact State Space," *ICRA*, pp. 238-244, May 1999.

[4] T. Lefebvre, "Contact Modeling, Parameter Identification and Task Planning for Autonomous Compliant Motion Using Elementary Contacts," Ph.D. Thesis, Katholieke Universiteit Leuven, Leuven, Belgium, May 2003.

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

[6] Q. Luo, E. Staffetti, J. Xiao, "Representation of Contact States between Free-Form Objects," *ICRA*, pp. 3589-3595, April 2004.

[7] Q. Luo and J. Xiao, "Physically Accurate Haptic Rendering with Dynamic Effects," *IEEE Computer Graphics and Applications, Special Issue - Touch-Enabled Interfaces*, Nov/Dec. 2004.

[8] F. Pan, J.M. Schimmels, "Efficient Contact State Graph Generation for Assembly Applications," *ICRA*, pp. 2591-2598, Sept. 2003.

[9] D. Ruspini, O. Khatib, "Collision/Contact Models for Dynamic Simulation and Haptic Interaction," *Proc. 9th Int. Symp. Robotics Research*, pp. 185-194, Oct. 1999.

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

[11] J. Xiao and X. Ji, "On Automatic Generation of High-level Contact State Space," *Int. Journal of Robotics Res.*, 20(7):584-606, July 2001.