Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Automatic Generation of Contact State Graphs between a Polygon and a Planar Kinematic Chain *

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 sometimes even necessary for a wide range of robotic tasks involving interactions between a robot and its environment or objects of manipulation. While most of the existing research is focused on contact states between two rigid bodies, this paper presents a practical approach to represent concisely and generate automatically graphs of contact states between a polygonal object and an articulated planar object, i.e., a planar kinematic chain. The approach effectively exploits topological and geometrical constraints associated with such contact states to ensure both correctness and efficiency, as demonstrated by the implementation and applied examples.**

***Index Terms*— contact state graphs, planar kinematic chain, polygon, automatic generation, topological and geometrical constraints, contact constraints, compliant motion**

I. Introduction

Many robotic tasks involve articulated robots in contact with the environment or objects and in compliant motion, ranging from grasping to whole arm manipulation to snake robot maneuvering inside a tight space for inspection, etc. It is often necessary to know not only the precise contact configuration of a robot but also the contact topology and geometry characterized by high-level, discrete, topological contact states. For example, when a robot arm is in a contact configuration, one needs to know which parts of the robot arm (e.g., upper or lower or both, inside or outside) are in what kind of contacts (e.g., point or line or area contacts), i.e., information captured in a contact state, in order to devise motions for whole-arm manipulation (e.g., [5], [8], [2]).

Contact states and adjacency information can be captured by a discrete *contact state graph*, where each node denotes a contact state and each arc links two adjacent contact states. Such a graph can be viewed as a partition of contact configurations on the surface of a C-obstacle. It offers a good characterization of a C-obstacle, which is difficult to compute in high-dimensional configuration space[1]. There is considerable research on representation and automatic generation of contact state graphs between two rigid bodies [7], [4], but not much attention is given to contact state space involving articulated objects, which introduce more degrees of freedom. Indeed, even for a planar kinematic chain interacting with a planar object, the corresponding C-obstacle can be of very high-dimensions if the chain has many degrees of freedom. In [3], a formalism was introduced to represent contact states involving an articulated polyhedral object based on the oriented matroid theory. However, it is too complex for implementation.

In this paper we present a practical approach to represent and generate automatically graphs of contact states between a polygonal object and a planar kinematic chain with revolute joints. A kinematic chain is the basic form of various components in an articulated robot (e.g., a finger, an arm, a leg, or a snake). Our approach effectively exploits topological and geometrical constraints associated with such contact states to ensure both correctness and efficiency, as demonstrated by the implementation and a test example. Note that we only consider geometrically feasible contact states without regarding the aspects of contact forces (such as force closure property).

The paper is organized as follows. Section II introduces the planar kinematic chain considered in this paper. Section III represents contact states between a polygonal object and a planar kinematic chain. In Section IV, neighboring contact states are defined, and in particular, locally-defined neighboring (LN) contact states, are analyzed. Section V describes our algorithm to generate automatically a graph of LN contact states from a seed contact state. The implementation and a testing example are introduced in Section VI, and Section VII concludes the paper.

II. Planar Kinematic Chain, Its Joint States, and Articulated Formation

For simplicity, we consider a planar kinematic chain with revolute joints as consisting of only link edges (of fixed lengths) and vertices, which include *joint vertices*, as shown in Figure 1.

The configuration of a planar kinematic chain with revolute joints can be defined as consisting of its base position and orientation and joint angles: $(x_b, y_b, \phi_b, \theta_1, ..., \theta_n)$, as shown in Figure 2.

We can qualitatively characterize the joint configurations $(\theta_1, ..., \theta_n)$ of a planar kinematic chain in the following

*This work is supported by the U.S. National Science Foundation under grant IIS-#0[illegible]

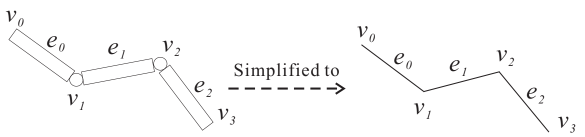

Fig. 1. An example planar kinematic chain, where v_1 and v_2 are joint vertices

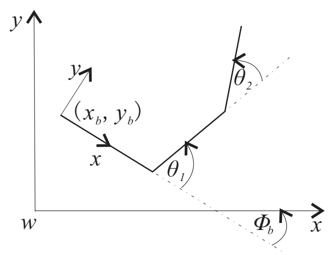

Fig. 2. The configuration of a planar kinematic chain

way. We use "+" and "-" to distinguish two sides of a planar kinematic chain. We further define the following types of topological *joint states* between two adjacent links:

- *acute state*: two adjacent links form an acute angle;
- *obtuse state*: two adjacent links form an obtuse angle;
- *line state*: two adjacent links form a straight line.

Figure 3 shows different types of joint states. Depending on which side the angle is formed between two adjacent links, we use "$++$" and "$+$" to denote a "+" side acute and obtuse state respectively and use "$--$" and "$-$" to denote a "-" side acute and obtuse state respectively. We use "0" to denote the line state.

III. Contact States between a Polygonal Object and a Planar Kinematic Chain

We define edges and vertices of a polygon or a planar kinematic chain as its *boundary elements*. We further define the vertices of an edge as the boundary elements of the edge. For simplicity, we use *e* and *v* to denote *edge* and *vertex* respectively.

Between a polygonal object A and a planar kinematic chain B, a joint vertex v of B is considered in a *concave* state if A is on the side where the links form v are in an acute or obtuse state; otherwise, it is *convex*. For example,

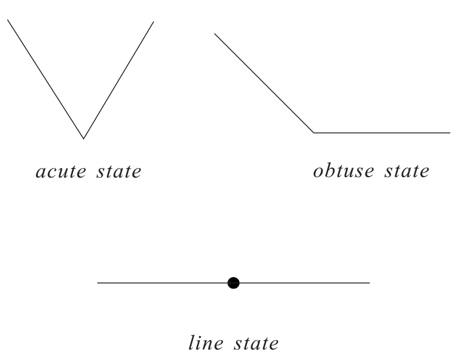

Fig. 3. Different topological joint states between two adjacent links of a planar kinematic chain

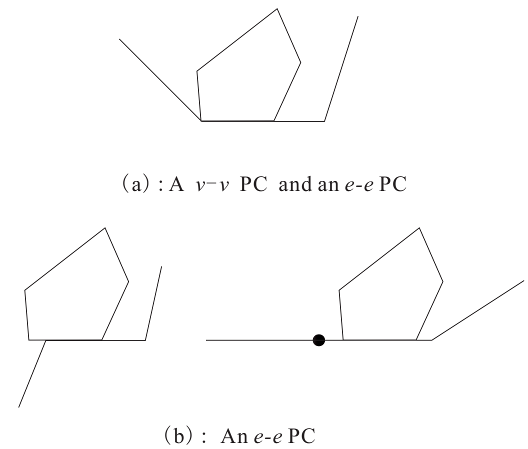

Fig. 4. Examples of different PCs between a polygon and a planar kinematic chain

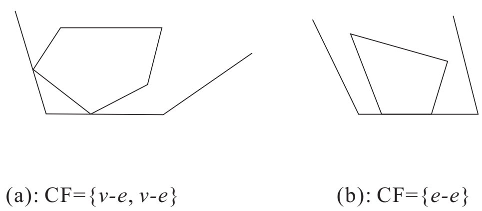

Fig. 5. Examples of contact formations

if v is in a "+" side acute state, and A is on the "+" side of B, then v is concave.

We now extend the notion of *principal contact* (PC) [6] to describe a primitive contact between a polygonal object A and a planar kinematic chain B: A principal contact is in terms of a pair of contacting boundary elements α_A of A and α_B of B such that not both elements are convex bounding elements of other contacting boundary elements.

There are three types of PCs: v-v, v-e/e-v, and e-e, where the first two types are *point contacts*, and the last type is a *line contact*.

Figure 4 shows two examples with different number of PCs. Figure 4a shows an example with two PCs, a point contact and a line contact. Figure 4b shows an example with only one PC of line contact.

We further extend the notion of *contact formation* [6], [7] to describe a general contact state between a polygonal object A and a planar kinematic chain B as a set of PCs formed: $CF = \{PC_1, PC_2, \cdots, PC_m\}$. Moreover, the *cardinality* of a CF describes the total number of PCs included in the CF, denoted as *card*(CF).

Figure 5 shows some examples of contact formations.

The *geometrical representation* of a CF denotes the set of configurations of the kinematic chain B relative to the contacting polygonal object A that satisfy all conditions of the PCs in the CF. Generally, such a set may consist of one or more connected regions of configurations, called *CF-connected* regions in the configuration space. Within a CF-connected region, there exists a motion constrained by the CF from any contact configuration to any other one, called a *CF-compliant* motion. In other words, there is no

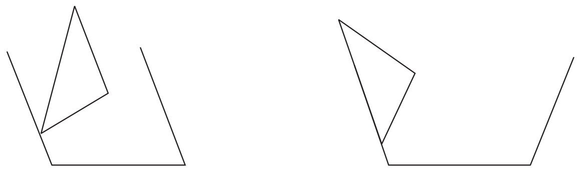

Fig. 6. Examples of LN CFs

CF_1={v-e} CF_2={v-e, v-e}

Fig. 7. An example of LN and GN CFs

need to change the CF in moving from one configuration to another within a CF-connected region. Thus, we define a *contact state*(CS) as a single CF-connected region of configurations, represented by the CF and a representative configuration in the region, denoted as a pair $<CF, C>$.

IV. Neighboring Contact States

Two contact states $<CF_i, C_i>$ and $<CF_j, C_j>$ are called *neighboring contact states* if there exists a relative and compliant motion of the contacting objects that changes CF_i to CF_j without going through another CF or complete loss of contact, and CF_i and CF_j are called *neighboring contact formations*. The motion is called a *neighboring transition motion*.

If two single-PC CFs, $CF_i = \{(PC_i, 1)\}$ and $CF_j = \{(PC_j, 1)\}$, where $PC_i \neq PC_j$, are neighboring CFs, then PC_i and PC_j are called *neighboring principal contacts*. Moreover, two neighboring PCs PC_i =α_{iA}-α_{iB} and PC_j = α_{jA}-α_{jB} satisfy one of the following conditions:

1) $\alpha_{iA} = \alpha_{jA}$ and α_{iB} is adjacent to α_{jB};
2) $\alpha_{iB} = \alpha_{jB}$ and α_{iA} is adjacent to α_{jA};

We can now further distinguish two kinds of neighboring contact states based on the topological information of the neighboring CFs. For two neighboring contact states $<CF_i, C_i>$ and $<CF_j, C_j>$, $<CF_j, C_j>$ is a locally-defined neighbor (LN) of $<CF_i, C_i>$ if the following conditions are satisfied:

- $card(CF_j) \leq card(CF_i)$, and
- every PC in CF_j either belongs to CF_i or is a neighboring PC of a PC in CF_i.

If $card(CF_j) < card(CF_i)$, then CF_i is a *globally-defined neighbor* (GN) of CF_j; $<CF_i, C_i>$ is a GN contact state of $<CF_j, C_j>$, and $<CF_j, C_j>$ is an LN contact state of $<CF_i, C_i>$.

The reason we differentiate LNs and GNs is that given a CF, the topological information of its LN CFs can be derived directly from its own topological information; that is, from the PCs in the CF, one can obtain the possible PCs of the LN CFs of the CF. This is a very useful property for automatic generation of articulate contact states.

Figure 6 shows an example of LN CFs. Figure 7 shows an example of LN and GN CFs, where CF_1 is a LN CF of CF_2, and its topological representation can be derived directly from CF_2; however, one cannot obtain the topological representation of CF_2, which is a GN CF of CF_1, directly from that of CF_1.

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

V. Generation of Contact State Space

Our approach to generate a contact state graph is similar to that used to generate a contact state graph between two rigid bodies [4] at the high-level: generate special subgraphs of the contact state graph $\mathcal{G}$ automatically and merge these subgraphs automatically.

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. In the loop for obtaining an LN contact state, there are two key steps:

(1) From a known contact state $<CF_i, C_i>$, hypothesize its LN CFs based on the topological information of CF_i, and

(2) determine if a hypothesized LN CF, CF_j, is valid or not by checking if there is a feasible neighboring transition motion from C_i under CF_i to some configuration C_j under CF_j that changes CF_i to CF_j. If such a feasible motion exist, then $<CF_j, C_j>$ is a valid LN contact state of $<CF_i, C_i>$.

It is in the above two key steps that our approach deals with challenges specific to contact states involving an articulated object (or in our case, a planar kinematic chain). We explain both steps in detail below.

A. Hypothesizing LN CFs

For a CF with a single principal contact $CF_i = \{PC_i\}$, to obtain a hypothesized LN CF of CF_i is to **change** PC_i to one of its neighboring PCs.

For a CF CF_i with multiple principal contacts, i.e., $card(CF_i) \geq 2$, an LN CF can be obtained by a combination of the following actions: **keep** some PCs, **change** some PCs to neighboring PCs, and **remove** some PCs. Thus, we also use these actions to hypothesize possible LN CFs of CF_i. Note that no **keep** or **remove** action can be applied simultaneously to all PCs in the CF.

However, if we blindly hypothesize LN CFs by taking a combination of **keep**, **change**, and **remove** PCs in CF_i, The number of hypothesized LN CFs will be too large the result of the combinatorial explosion, but most of these hypothesized ones are not possible topologically or geometrically.

Our strategy is therefore to try to hypothesize only topologically possible combinations as candidate LN CFs, and for these candidates, we next check if they are geometrically feasible by considering the relevant neighboring transition motions (see next subsection). For CFs between a polygonal object A and a planar kinematic chain B, many topologically infeasible combinations can be ruled out based on the following:

1) A vertex of one object cannot contact two different edges or vertices of the other object.
2) If A is convex, then it is impossible for an edge of B to contact two different edges of A or to contact two different vertices of A not sharing the same edge.
3) It is impossible for a CF to include such two pairs of v-v type PCs that the vertices on each object are bounds of one edge, but the two edges are not of the same length.
4) It is impossible for a CF to include two PCs defined by the two vertices of one edge e^A contacting an edge e^B or its bounding vertices at the same time, if e^A is longer than e^B.
5) It is impossible for a CF to include two PCs defined by the two vertices of one edge e^B contacting an edge e^A or its bounding vertices at the same time, if e^B is longer than e^A.
6) It is impossible for a CF to include the following three types of PCs: $v^A{}_1$-e_1^B, v_2^A-e_2^B, and e^A-e_3^B, where v_1^A and v_2^A bound e^A and e_3^B is adjacent to either e_1^B or e_2^B.
7) It is impossible for a CF to include the following three types of PCs: e_1^A-e_1^B, e_2^A-e_2^B, and v^A-v^B, where v^A is common to both e_1^A and e_2^A, e_1^B and e_2^B have no more than one link between them, but v^B is not shared by e_1^B and e_2^B.
8) It is impossible for a CF to include the following two types of PCs: v^A-e^B and e^A-v^B, where v^B is a vertex of e^B, but v^A is a vertex of e^A. If A is convex, then it is impossible to have those two types of PCs even if v^A is not a vertex of e^A.
9) It is impossible for a CF to include the following two types of PCs: e^A-e^B,v^A-v^B, where v^A is a vertex of e^A, v^B is not a vertex of e^B but is a vertex of an edge adjacent to e^B.
10) It is impossible for a CF to include the following two types of PCs: e_1^A-e^B, e_2^A-v^B, where e_1^A and e_2^A are adjacent, and v^B is a vertex of e^B.

B. Neighboring Transition Motions

Given a rigid polygonal object A and a planar kinematic chain B. To change the CF between them to a locally-defined neighboring (LN) CF, a neighboring transition motion of the kinematic chain B can be used. B can both translate and rotate without changing its joint configuration or have joint motions, all under the contact constraints. Figure 8 shows the neighboring transition motion of B changing a v^A-e^B PC to a v^A-v^B PC in a three-PC CF.

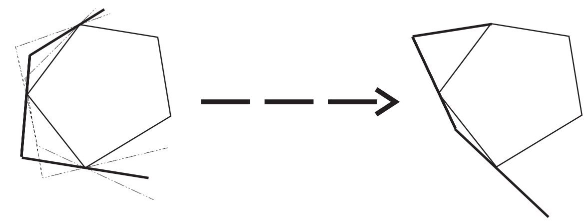

Fig. 8. An example of neighboring transition motion.

To check if a hypothesized LN CF CF_j from a CF CF_i is geometrically feasible is to see whether or not there exists a neighboring transition motion from CF_i to CF_j. Any neighboring transition motion may involve **remove**, **keep**, or **change** one or more PCs of CF_i, as introduced earlier.

To find a neighboring transition motion, our strategy is to first check if there are either v-v or e-e types of PCs to **keep** because only a few types of motions can keep such PCs:

- To keep a v-v type of PC, only **pure rotations** or **revolute joint motions** are possible.
- To keep a e-e type of PC, only **pure translations** or **pure translations** combined with **revolute joint motions** are possible.

We next see if one of the above motions can also accomplish the **change** or **remove** actions as required by the transition from CF_i to CF_j. If one motion can do so without causing unwanted contact or loss of contact, we then know that CF_j is a valid LN CF of CF_i.

If the transition from CF_i to CF_j does not require keeping any v-v or e-e type of PCs, our strategy is to consider a **change** motion of one PC PC_i in CF_i to a hypothesized PC PC_j in CF_j if a **change** action is required. Such a change motion can be a combination of base motions (i.e., translation and rotation) and various joint motions of the kinematic chain B, under the contact constraints on contacting links of B and also the geometric constraints on both the contacting and non-contacting links of B. Such a motion of a kinematic chain can be quite complex without an analytical expression (i.e., without analytical solutions of the constraint equations and inequalities, which are transcendental with respect to joint variables). Thus, we use a numerical approach to construct such a motion and to check its geometrical feasibility.

To be concrete, if a link e^B is involved in changing PC_i in CF_i to PC_j in CF_j, we first consider the required motion of e^B as classified in the following 8 categories:

1) To change a v^A-e^B type PC to an e^A-e^B type PC, where v^A is a vertex of e^A, the motion of e^B has to include a rotation about v^A of the angle between e^A and e^B.
2) To change an e^A-e^B type PC to a v^A-e^B type PC, where v^A is a vertex of e^A, if v^A is on e^B, the motion of e^B has to include a small rotation about v^A; otherwise, a translation of e^B along e^A is needed to make v^A on e^B before the motion with the small

rotation of e^B.

3) To change an e^A-v^B type PC to an e^A-e^B type PC, where v^B is a vertex of e^B, the motion of e^B has to include a joint motion or rotation about v^B of the angle between e^A and e^B.
4) To change an e^A-e^B type PC to an e^A-v^B type PC, where v^B is a vertex of e^B, the motion of e^B has to include a small joint motion or rotation about v^B if v^B is on e^A; otherwise, a translation of e^B along e^A is needed to make v^B on e^B before the motion with a small joint motion or rotation of e^B.
5) To change a v^A-v^B type PC to an e^A-v^B type PC, where v^A is a vertex of e^A, the motion of either link sharing v^B has to include a small translation along e^A.
6) To change an e^A-v^B type PC to a v^A-v^B type PC, where v^A is a vertex of e^A, the motion of either link sharing v^B has to include a translation along e^A to make v^B meet v^A.
7) To change a v^A-e^B type PC to a v^A-v^B type PC, where v^B is a vertex of link e^B, the motion of e^B has to include a translation to make v^B reach v^A while keeping e^B in contact with v^A.
8) To change a v^A-v^B type PC to a v^A-e^B type PC, where v^B is a vertex of e^B, the motion of e^B has to include a translation to move v^B away from v^A while keeping v^A contacting e^B.

Note that in the above, only the *required* motions are specified, and such motions of e^B are of one degree of freedom. However, a v-e/e-v PC allows two degrees of motion freedom for an involved edge. Therefore, a required compliant rotation (or translation) of e^B in a v-e/e-v PC can be accompanied by compliant translations (or rotations/joint motions). The extra degree of freedom can be used to add a "wiggling" component to the required motion in order to satisfy the geometric constraints and other contact constraints when the motion of the entire kinematic chain is considered. When there is an extra degree of freedom, if the required motion of e^B is a translation, then a wiggling rotational motion can be added, and if the required motion of e^B is a rotation, then a wiggling translational motion can be added if helpful.

To incorporate the corresponding motions of the other links of B when e^B moves, we digitize the motion of e^B into small steps, and for each small step motion of e^B, our algorithm then moves the other relevant links of B in corresponding small steps to satisfy both the geometric constraints of B and the contact constraints of the other PCs in CF_j. If each small step of every moving link is free of unintended contact/collision or unintended loss of contact, we say that a feasible neighboring transition motion exists (as the integration or sum of the small steps) between CF_i and CF_j. We then know that CF_j is a valid LN CF of CF_i.

As an example, let us consider how the neighboring transition motion indicated in Figure 8 is found. The object boundary elements and the starting CF of that example

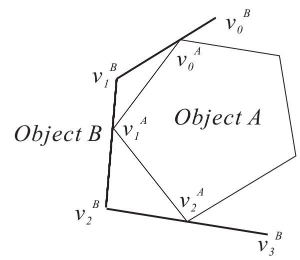

Fig. 9. Two objects A and B for the example in Figure 8

are drawn in Figure 9. Object A is assumed fixed, and the coordinates for v_0^A and v_1^A are (x_0^A, y_0^A) and (x_1^A, y_1^A) respectively. Let (x_i^B, y_i^B) denote the coordinate of B's vertex v_i^B, $i = 0, 1, 2, 3$. Let $\mathbf{d}(i, i{+}1)$ be the distance from v_i^B to v_{i+1}^B, i=0, 1, 2. Let e_i^B denote the i-th link of B, and let l_i be the length of e_i^B, i=0, 1, 2. The geometric constraints of B can be expressed as $\|\mathbf{d}(i,i{+}1)\| = l_i$, $i = 0, 1, 2$, and the bounds on the angle between two vectors $\mathbf{d}(i,i{+}1)$ and $\mathbf{d}(i{+}1, i{+}2)$, i=0, 1. Under the CF shown on the left of the figure 8, there are three v^A-e^B types of PCs, resulting in three contact constraint equations and additional inequalities.

To change the CF on the left to the CF on the right in Figure 8 requires the change of the PC v_0^A-e_0^B to the neighboring PC v_0^A-v_0^B. Thus, according to our algorithm, the motion of e_0^B requires a translation to make v_0^B reach v_0^A. By adding a wiggling rotational motion, the actual path of v_0^B is a waving one towards v_0^A. The wiggling motion is characterized by a bound on magnitude and a frequency: the magnitude is within the allowed interval for the rotational degree of freedom, and the frequency is related to the resolution of the digitization for the motion of v_0^B (i.e., the finer resolution, the higher frequency). For each small step motion of v_0^B, the corresponding small motion of v_1^B is decided based on the geometric constraints between v_0^B and v_1^B and the contact constraints involving v_0^A. Subsequently, the corresponding small motion of v_2^B is decided based on the geometric constraints between v_1^B and v_2^B and the contact constraints involving v_1^A, and the small motion of v_3^B can be decided in turn.

If, at any step, under the geometric constraints, some contact constraints cannot be satisfied and loss of contact or penetration occurs, the motion is considered not feasible. A new motion can be tried by adjusting the wiggling component (either its magnitudes or frequencies). Our analysis and tests show that if a few wiggly motions cannot be successful, the transition from CF_i to a hypothesized LN CF CF_j is either extremely difficult or impossible. Thus, CF_j is discarded as not a valid LN CF of CF_i.

Finally, if a neighboring transition motion only requires to **remove** some PCs, the motion can be infinitesimal. Again, our strategy first identifies a link e^B of B involved in a PC to be removed and an infinitesimal motion of e^B to remove the PC and then moves the other links while satisfying the geometric constraints and other contact constraints. If such a motion cannot be found without a

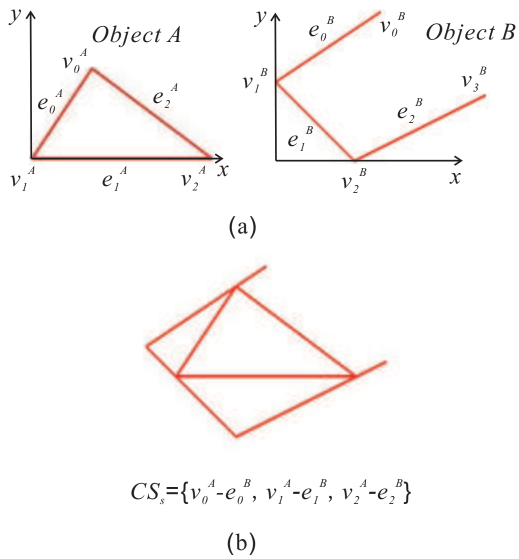

Fig. 10. An example

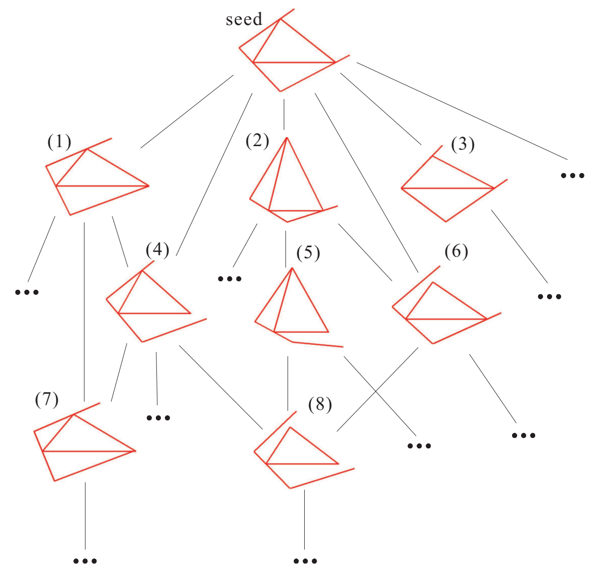

Fig. 11. Some contact states generated for the example

violation of constraints, the hypothesized LN CF is not valid.

VI. Implementation

We have implemented the general algorithm for automatic generation of an LN graph from a seed contact state between a polygonal object A and a planar kinematic chain B. The algorithm is implemented in Microsoft Visual C++ 6.0.

Figure 10 shows an example that our algorithm applied. In Figure 10a, object A is a triangle, and its boundary consists of three different vertices and edges. The three edges e_0^A, e_1^A, and e_2^A have lengths 3.6, 6.0, and 5.0 mm respectively. Object B has three links e_0^B, e_1^B, and e_2^B, with lengths 4.8, 4.24, and 5.59 mm respectively. In Figure 10b, a seed contact state is shown with three PCs. From this seed contact state, our algorithm has generated an LN graph with 86 contact states in 23 seconds. Figure 11 shows some contact states generated.

Note that in Figure 11, the contact state (7) appears to be the same as the contact state (1), but actually the two states are not the same. Both contact states involve two v-e types of PCs, but the contact state (1) has a v-v PC, whereas the contact state (7) has a e-v PC, in addition.

Note that for the implemented example, if topologically infeasible combinations are not ruled out in hypothesizing LN CFs, the total number of PC combinations (for n PCs, $1 \leq n \leq 3$) will be 26502. With many topologically infeasible combinations ruled out as described in Section V.A, the number of hypothesized CFs is drastically reduced to 2098. Among these hypothesized CFs, 86 of them are also geometrically feasible and therefore represent valid contact states as generated by our algorithm.

VII. Conclusions

In this paper, we have presented a practical new approach to generate automatically contact state graphs between a polygonal object and a planar kinematic chain. Automatic generation of contact state graphs is not only very desirable because it is tedious to do that manually but also necessary since many contact states and state transitions cannot be imagined or drawn easily. This is particularly true when a kinematic chain, not just a rigid body, is involved in a contact. The introduced algorithm is very efficient in exploiting kinematic and contact constraints to determine valid contact states and state transitions. It can be extended to dealing with more than one kinematic chain in contact. As a future step, more complex examples will be tested with our algorithm, and more complex objects, such as curved objects, will be considered. A long term objective is to investigate extension to spatial kinematic chains and objects.

References

[1] J.F. Canny, "The Complexity of Robot Motion Planning," *MIT Press*, 1988.
[2] P. Song, M. Yashima, V. Kumar, "Dynamic Simulation for Grasping and Whole Arm Manipulation," *Proc. IEEE Int. Conf. Robotics & Automation* (ICRA), pp. 1082 - 1087 vol.2 , 2000.
[3] E. Staffetti, W. Meeussen, J. Xiao, "A New Formalism to Characterize Contact States Involving Articulated Polyhedral Objects," *ICRA 2005*, pp. 3630-3637, 2005.
[4] P. Tang and J. Xiao, "Generation of Point-contact State Space between Strictly Curved Objects," *Proc. Robotics: Science and Systems Conference,* Philadelphia, August 16-19, 2006.
[5] T. Watanabe, K. Harada, T. Yoshikawa,Z.W. Jiang, "Towards Whole Arm Manipulation by Contact State Transition," *Proc. International Conference on Intelligent Robots and Systems* (IROS), Beijing, China, October 2006.
[6] J. Xiao, "Automatic Determination of Topological Contacts in the Presence of Sensing Uncertainties," *ICRA 1993*, pp. 65-70, Atlanta, May 1993.
[7] J. Xiao and X. Ji, "On Automatic Generation of High-level Contact State Space," *International Journal of Robotics Research*, (and its first multi-media extension issue http://www.ijrr.org/), 20(7):584-606, July 2001.
[8] M. Yashima, "On Planning for Whole Arm Manipulation with Switching Contact Modes," *IROS 2001*, Hawaii, USA, Oct. 29-Nov. 3, 2001.