The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

# Generating a Contact State Graph of Polyhedral Objects for Robotic Application

Sung Jo Kwak, Seong Youb Chung, and Tsutomu Hasegawa, *Member, IEEE*

***Abstract*—Traditional methods require large computation to model a contact state graph of polyhedral objects for robotic application, and moreover they are heuristic. In this paper, we propose a framework to generate the contact state graph automatically. All faces of the polyhedral objects are triangulated. A sub-contact is defined as single contact between two polyhedral objects and a contact state is presented with a set of the sub-contacts. There are two sub-contacts, a vertex-triangle contact and an edge-edge contact. According to convexity or concavity of the edges composing the triangle, the vertex-triangle contact and the edge-edge contact are classified into 10 types and 7 types, respectively. A contact state graph is made by evolutionary transitions of the sub-contacts. This procedure is accomplished only using the topology of the sub-contacts, which is possible in real-time. The proposed framework is evaluated by an example of square peg-in-hole assembly.**

## I. Introduction

A sequence or a path is required to automatically operate robots for an assembly task in workspace. In general, the geometric information of objects is used to make an available path in off-line, and the path is realized by sensing and controlling the position of the robot. However, in on-line operation, uncertainties such as tolerance, collision errors, or sensor noise make it difficult to follow the defined path, because they can cause the positioning error of the robots or the objects. So, tactile sensors or vision sensors are broadly used to overcome the uncertainties [1], [2]. The compliance control methods using mechanical devices or force-moment sensors have been proposed and succeeded in unidirectional peg-in-hole insertion and in polyhedral environment [3], [4]. However, these methods have such limitations as it should put the peg near the entrance of the hole and should apply difference control policies in accordance with the shapes of the holes. Hence they are inadequate to deal with the polyhedral objects which have complex-shaped faces.

A few researches have proposed contact states to find admissible path for robotic assembly under the uncertainties. The contact state represents a contact condition of objects during assembly and the contact state graph is built up by giving transitional relations among the contact states. Some researchers developed the methods to generate the contact state graph using the elements of objects such as vertex, edge, and face. Hirai [5] considered the contact states as combination of plausible primitive contacts, and verified those states by applying randomly sampled points to the contact space. Chung and Lee have proposed an automatic method to generate the contact state fast on level of topology [6]. Xiao and Xi generated the contact state graph by using the principal contacts and infinitesimal motion between polyhedral objects [7]. Staffetti *et al*. [8] specified the contact states in polyhedral objects and polyhedral environment. In recent, Tang and Xiao expanded it to generate contact states between 3D-curved objects [9]. Although they proposed an automatic approach for contact modeling, they require much computation because of exhaustive search on configuration space. Moreover, it is difficult to make a generalized rule for automatic generation of contact states, because they do not regard faces of a polyhedral object as polygons. Although we have information of vertices, edges, and faces, their methods are inefficient because the generation rules are case by case according to the shape of a face.

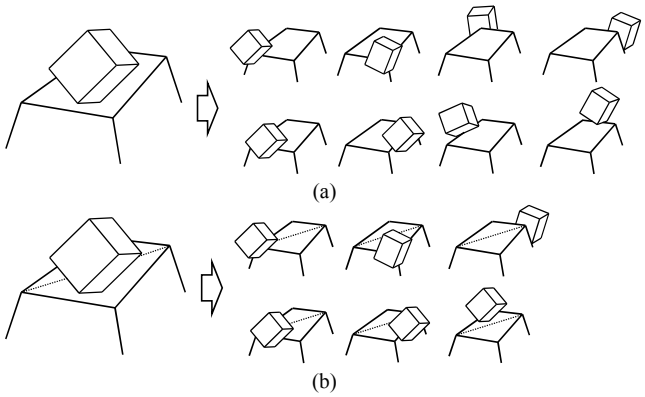


Fig. 1. Contact states generated with (a) a vertex-face contact and (b) a vertex-triangle contact.

In this paper, we propose a new framework to automatically generate a contact state graph of polyhedral objects. First we triangulate all faces of a polyhedral object. If all faces are triangles with three vertices and three edges, we can apply a single rule to find neighboring contact states. A sub-contact is newly defined as a vertex-triangle contact. Then a contact state is analytically represented by using two sub-contacts of the vertex-triangle contact and the edge-edge contact. This method can distinguish contact situation more clearly than the vertex-face contact. For example, in case of the vertex-face contact of Fig. 1(a), we have to investigate

S. J. Kwak and T. Hasegawa are with Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan (e-mail: sungjo@irvs.is.kyushu-u.ac.jp, hasegawa@ait.kyushu-u.ac.jp).

S. Y. Chung is with Department of Mechanical Engineering, Chungju National University, Chungju, Republic of Korea (e-mail: sychung@cjnu.ac.kr).

about all vertices, edges, and faces neighboring it to generate contact states. On the other hand, the search range is restricted to only three vertices, three edges, and three adjacent triangles for the vertex-triangle contact of Fig. 1(b). Therefore, in view of the contact state generation, the vertex-triangle contact can reduce the search range of contact states.

This paper is structured as follows. Section II-A describes the definition of the sub-contacts and triangulation for the vertex-triangle contact is described in Section II-B. Adjacent sub-contacts from the current sub-contact are defined according to the type of adjacent triangles in Section II-C. Using the adjacent sub-contacts, the framework for automatic generation of the contact state graph is presented in Section III. It is evaluated with simulations on a square peg-in-hole assembly of Section IV. We conclude in Section V with some final remarks.

## II. Sub-contacts for Polyhedral Object

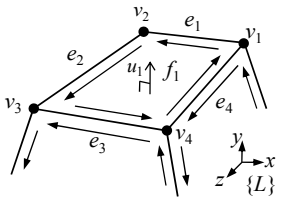
### A. Definition of Polyhedral Object and Sub-Contacts

The topological information is useful to describe adjacent relations between vertices (*v*s), edges (*e*s), and faces (*f*s) of a polyhedral object. The elements of a polyhedral object are presented with positions of vertices, two vertices of edges, and edges of faces as shown in Fig. 2. For instance, the $x_1$, $y_1$, and $z_1$ of Fig. 2 is the position of $v_1$ in the local coordinate ($L$) of the polyhedral object. The $e_1$ consist of two vertices of $v_1$ and $v_2$, and the four edges of $e_1$, $e_2$, $e_3$, and $e_4$ are arranged in counter-clockwise order on the normal vector ($u_1$) to make the $f_1$. The adjacent relation between elements implies the connection by an edge in case of two vertices, by a vertex in case of two edges, and by an edge in case of two faces. The adjacent relations are used to generate another contact states.

In our framework, all faces of the polyhedral objects are triangulated to employ the proposed sub-contacts (*SC*s). The sub-contacts are defined as a vertex-triangle (*v*-*t* or *t*-*v*) contact and an edge-edge (*e*-*e*) contact between two objects in Fig. 3. The sub-contacts are denoted as ($v_i$, $t_j$) or ($t_i$, $v_j$) for the vertex-triangle contact and ($e_i$, $e_j$) for the edge-edge contact. Although the contacts of *v*-*v*, *v*-*e*, *e*-*t* and *t*-*t* can be presented theoretically, we do not regard them as the sub-contacts. It is because the *v*-*v* contact and the *v*-*e* contact are impossible in robotic operation and the *e*-*t* contact and *t*-*t* contact are multi-contact situations which can be described by two *v*-*t* contacts and three *v*-*t* contacts, respectively.

### B. Triangulation of Faces

We propose a hierarchical structure of vertices and edges to triangulate faces. In other words, all vertices and all edges are stacked in the hierarchical structure according to their adjacent relations. The structure of adjacent triangles is constructed using the adjacent relations in the hierarchical structure. This hierarchy is effective to decrease the search range of contact states, because it consists of elements to be passed from an initial contact to a target contact.




| Element | Components |
|---|---|
| $v_1$ | $\{x_1, y_1, z_1\}$ |
| ⋮ | ⋮ |
| $e_1$ | $\{v_1, v_2\}$ |
| ⋮ | ⋮ |
| $f_1$ | $\{e_1, e_2, e_3, e_4\}$ |
| ⋮ | ⋮ |

Fig. 2. Topologies and geometries of a polyhedral object.

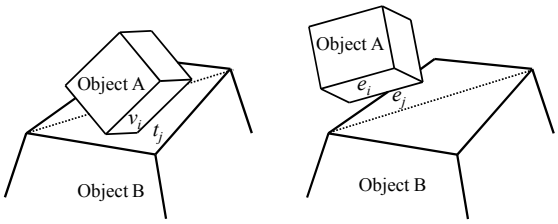



Fig. 3. Definition of sub-contacts: (a) the vertex-triangle contact and (b) the edge-edge contact.

If all faces of a polyhedral object consist of convex polygons in Fig. 4(a), the adjacent vertices (*a*s) of $v_i$ are defined as the vertices which are linked by edges from the $v_i$. If a face is a concave polygon of Fig. 4(b) or a hollow polygon of Fig. 4(c), the adjacent vertices are obtained after the face is divided into pieces of convex polygons using virtual edges [10]. The hierarchical structure of adjacent vertices and edges is constructed in Fig. 5. A source vertex is ranked on the top of the hierarchical structure. We select the source vertex among vertices of an initial contact, because a contact state graph is evolved from the initial contact to a target contact. The adjacent vertices of the source vertex are listed on second level and again their adjacent vertices are ordered on third level. New level is made until the hierarchical structure includes all vertices. The $n_v$ in Fig. 5 indicates the level of vertices in the hierarchical structure and the $n_e$ indicates the level of edges at $n_v$ and between $n_v$ and $n_v$+1.

If a face is not a triangle itself, basically triangulation of faces is accomplished by drawing diagonals between vertices on each face. Fig. 6(a) shows diagonals drawn arbitrarily on a face. Let the arrows represent directions of contact transition. The $e_2$ in Fig. 6(a) is more adjacent than $e_1$ from ($v_1$, $t_1$) but it is not true. So we develop an algorithm using the hierarchical structure of adjacent vertices to solve this problem. Fig. 6(b) shows the result obtained by the following procedure. 1) An arbitrary line is drawn from an *a* of the $n_v$ to another *a* of the $n_v$+1 in Fig. 5. 2) If the arbitrary line is not an edge and its two vertices belong to a same face, the line becomes a diagonal and is stored in the list of edges and diagonals (*g*s). In the same way, 3) if an *a* of $n_v$ makes two edges between $n_v$ to $n_v$+1, an arbitrary line is drawn between two vertices of the two edges. 4) If the line also is not an edge and the two vertices belong to a same face, it also becomes a diagonal. This procedure is repeated until the last $n_v$ of the hierarchical structure.

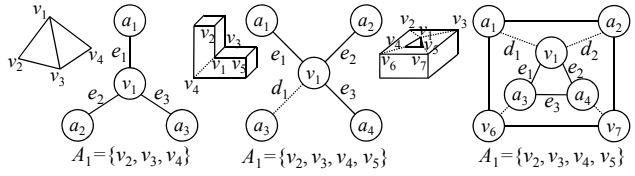



Fig. 4. Adjacent vertices on (a) convex faces, (b) a concave face, and (c) a hollow face.

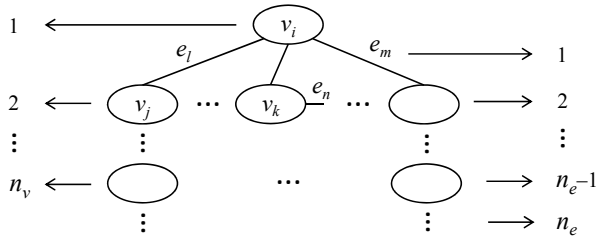


Fig. 5. Hierarchical structure of adjacent vertices and edges.

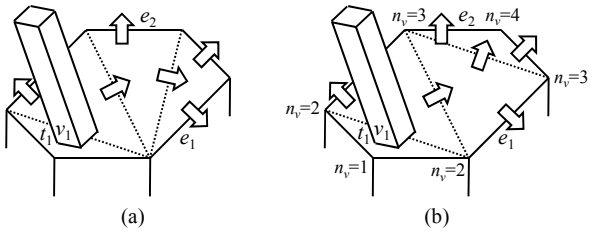



Fig. 6. Diagonals drawn (a) arbitrarily and (b) by the proposed procedure on a face.

After diagonals are drawn on all faces, we can make triangles using the edges and the diagonals in the hierarchical structure. The triangulation is performed by the following procedure. 1) Two *g*s at a $n_e$ are chosen in order of *g*s. 2) It is investigated whether *a*s of the two g*s* are a same vertex and the other vertices make another *g*. 3) If it is true, the two *g*s and another *g* compose a triangle. This procedure is also repeated until it reaches the last $n_e$. Finally, the structure of adjacent triangles (*t*s) is constructed as shown in Fig. 7.

### C. Adjacent Sub-contacts

Adjacent sub-contacts are defined as sub-contacts which can be transferred from a sub-contact while maintaining its contact. A triangle has three adjacent triangles. Each adjacent triangle can be one among three features defined according to its spatial relationship in Fig. 8, where two triangles of the feature A, the feature B, and the feature C have shares a convex edge, a diagonal, and a concave edge to each other, respectively. In this paper, we assume that the convex edges and the concave edges are known.

The adjacent sub-contacts from a *v*-*t* contact or an *e*-*e* contact are generated differently according to the types of adjacent triangles. First the number of types for adjacent triangles of the *v*-*t* contact is determined by (1).

$$\text{The number of types for } v\text{-}t = {}_3H_3 - 1 + 1 = 10. \quad (1)$$

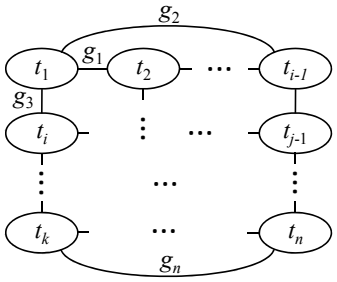


Fig. 7. Structure of adjacent triangles.

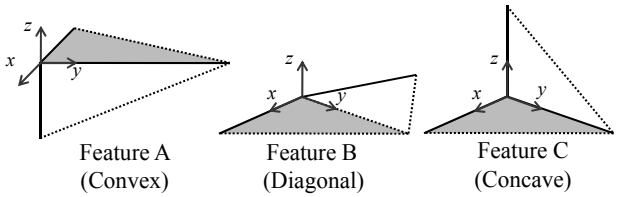



Fig. 8. Features of adjacent triangle.

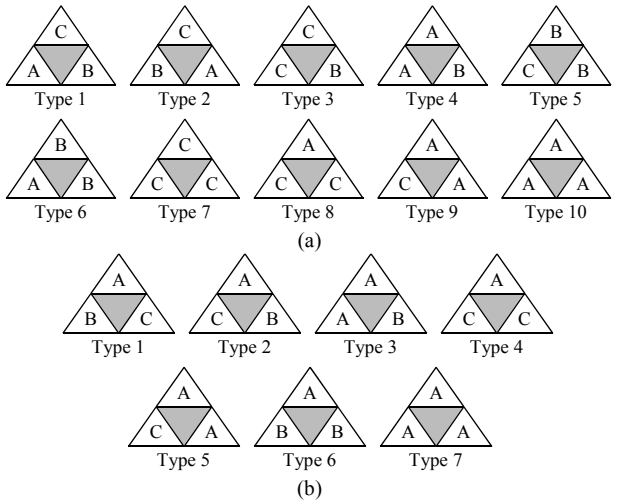



Fig. 9. Types of adjacent triangles (a) for the *v*-*t* contact and (b) for the *e*-*e* contact.

where ${}_3H_3$ is the number of combinations with repetition of A, B, and C. The {B, B, B} is excluded from the types of the *v*-*t* contact, because it does not appear in case of convex polygons and a concave polygon and a hollow polygon can be divided into some convex polygons [10]. Since the {A, B, C} and {A, C, B} make different types in space, both of them are presented. Then the 10 types of Fig. 9(a) are obtained for the *v*-*t* contact.

On the other hand, a convex relation is only meaningful for the *e*-*e* contact. Hence the number of types for the *e*-*e* contact is computed as the following equation.

$$\text{The number of types for } e\text{-}e = {}_3H_2 + 1 = 7. \quad (2)$$

where ${}_3H_2$ is the number of combinations with repetition of adjacent triangles except the feature A. The {A, B, C} and {A, C, B} are also different shapes in space as mentioned above. Hence the adjacent triangles for the *e*-*e* contact have 7 types of Fig. 9(b).

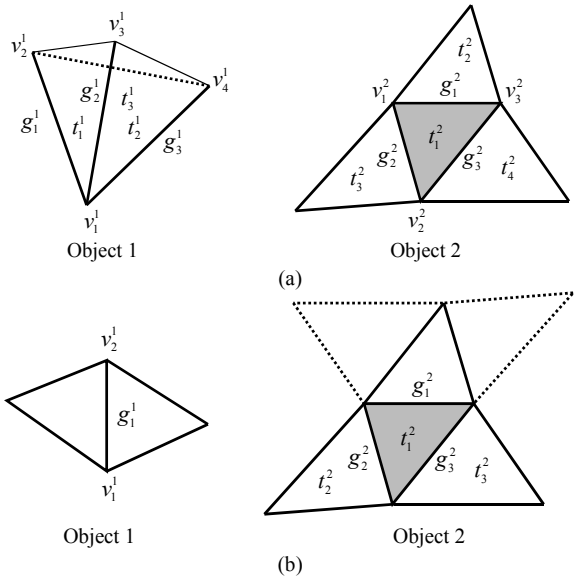



Fig. 10. Adjacent elements to generate adjacent sub-contacts (a) of a *v-t* contact and (b) of an *e-e* contact.

A *v-t* contact and an *e-e* contact can transfer to another sub-contact of *v-t* contact, *t-v* contact, or *e-e* contact, namely, adjacent sub-contacts. The adjacent sub-contacts can be indentified using adjacent vertices, edges, and triangles, namely, adjacent elements. The adjacent sub-contacts can be induced with the adjacent elements of Fig. 10(a) for *v-t* contact and those of Fig. 10(b) for *e-e* contact, where *v*, *g*, and *t* indicate topologies of vertices, edges or diagonals, and triangles. The adjacent sub-contacts of the *v-t* contact and the *e-e* contact are obtained with Table I and Table II. The first column of the tables is the types of adjacent triangles of Fig. 9(a) and Fig. 9(b). The second column is its adjacent sub-contacts. Then a *v-t* contact and an *e-e* contact using the tables can transfer to the adjacent sub-contacts of the second column according to the type of adjacent triangles.

## III. Contact State Graph

### A. Generation of Contact State Graph

A contact state can be represented with a set of sub-contacts. For example, the contact state of Fig. 11(a) is represented as $\{SC_1=(v_1, t_1), SC_2=(v_2, t_2)\}$. Also the contact state of Fig. 11(b) is described as $\{SC_1=(v_1, t_1), SC_2=(e_1, e_2)\}$. To develop a contact state graph, first the adjacent elements of sub-contacts composing the current contact state are selected in the structures of Fig. 5 and Fig. 7. The adjacent sub- contacts of each sub-contact are obtained using Table I and II. A new contact state is made as a set of the current sub-contact and the adjacent sub-contacts. However, it is not guaranteed that all contact states are possible in case of multiple-point contact situation, because the contact states are generated only with topologies of objects. So we propose four rules of (3) to guarantee feasible contact states. The *DS* in (3) is a set of distances between vertices, edges, and triangles of two sub-contacts representing a contact state. For example,

TABLE I

Adjacent Sub-contacts of $(v_1^1, t_1^2)$

| Type | Adjacent sub-contacts |
|---|---|
| 1 | $(v_1^1, t_2^2), (v_1^1, t_4^2), (g_1^1, g_2^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_4^2), (t_1^1, v_2^2), (g_2^1, g_2^2), (v_3^1, t_1^2), (v_3^1, t_2^2), (v_3^1, t_4^2), (t_2^1, v_2^2), (g_3^1, g_2^2), (v_4^1, t_1^2), (v_4^1, t_2^2), (v_4^1, t_4^2), (t_3^1, v_2^2)$ |
| 2 | $(v_1^1, t_2^2), (v_1^1, t_3^2), (g_1^1, g_3^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_3^2), (t_1^1, v_2^2), (g_2^1, g_3^2), (v_3^1, t_1^2), (v_3^1, t_2^2), (v_3^1, t_3^2), (t_2^1, v_2^2), (g_3^1, g_3^2), (v_4^1, t_1^2), (v_4^1, t_2^2), (v_4^1, t_3^2), (t_3^1, v_2^2)$ |
| 3 | $(v_1^1, t_2^2), (v_1^1, t_3^2), (v_1^1, t_4^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_3^2), (v_2^1, t_4^2), (v_3^1, t_1^2), (v_3^1, t_2^2), (v_3^1, t_3^2), (v_3^1, t_4^2), (v_4^1, t_1^2), (v_4^1, t_2^2), (v_4^1, t_3^2), (v_4^1, t_4^2)$ |
| 4 | $(v_1^1, t_4^2), (g_1^1, g_1^2), (g_1^1, g_2^2), (v_2^1, t_1^2), (v_2^1, t_4^2), (g_2^1, g_1^2), (g_2^1, g_2^2), (v_3^1, t_1^2), (v_3^1, t_4^2), (t_1^1, v_1^2), (t_1^1, v_2^2), (t_1^1, v_3^2), (g_3^1, g_1^2), (g_3^1, g_2^2), (v_4^1, t_1^2), (v_4^1, t_4^2), (t_3^1, v_1^2), (t_3^1, v_2^2), (t_3^1, v_3^2)$ |
| 5 | $(v_1^1, t_2^2), (v_1^1, t_3^2), (v_1^1, t_4^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_3^2), (v_2^1, t_4^2), (v_3^1, t_1^2), (v_3^1, t_2^2), (v_3^1, t_3^2), (v_3^1, t_4^2), (t_1^1, v_3^2), (v_4^1, t_1^2), (v_4^1, t_2^2), (v_4^1, t_3^2), (v_4^1, t_4^2), (t_3^1, v_3^2)$ |
| 6 | $(v_1^1, t_2^2), (v_1^1, t_4^2), (g_1^1, g_2^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_4^2), (t_1^1, v_1^2), (t_1^1, v_2^2), (t_1^1, v_3^2), (g_2^1, g_2^2), (v_3^1, t_1^2), (v_3^1, t_2^2), (v_3^1, t_4^2), (t_2^1, v_1^2), (t_2^1, v_2^2), (t_2^1, v_3^2), (g_3^1, g_2^2), (v_4^1, t_1^2), (v_4^1, t_2^2), (v_4^1, t_4^2), (t_3^1, v_1^2), (t_3^1, v_2^2), (t_3^1, v_3^2)$ |
| 7 | $(v_1^1, t_2^2), (v_1^1, t_3^2), (v_1^1, t_4^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_3^2), (v_2^1, t_4^2), (v_3^1, t_1^2), (v_3^1, t_2^2), (v_3^1, t_3^2), (v_3^1, t_4^2), (v_4^1, t_1^2), (v_4^1, t_2^2), (v_4^1, t_3^2), (v_4^1, t_4^2)$ |
| 8 | $(v_1^1, t_3^2), (v_1^1, t_4^2), (g_1^1, g_1^2), (v_2^1, t_1^2), (v_2^1, t_3^2), (v_2^1, t_4^2), (g_2^1, g_1^2), (v_3^1, t_1^2), (v_3^1, t_3^2), (v_3^1, t_4^2), (g_3^1, g_1^2), (v_4^1, t_1^2), (v_4^1, t_3^2), (v_4^1, t_4^2)$ |
| 9 | $(v_1^1, t_3^2), (g_1^1, g_1^2), (g_1^1, g_3^2), (v_2^1, t_1^2), (v_2^1, t_3^2), (g_2^1, g_1^2), (g_2^1, g_3^2), (v_3^1, t_1^2), (v_3^1, t_3^2), (t_1^1, v_3^2), (g_3^1, g_1^2), (g_3^1, g_3^2), (v_4^1, t_1^2), (v_4^1, t_3^2), (t_3^1, v_3^2)$ |
| 10 | $(g_1^1, g_1^2), (g_1^1, g_2^2), (g_1^1, g_3^2), (v_2^1, t_1^2), (g_2^1, g_1^2), (g_2^1, g_2^2), (g_2^1, g_3^2), (v_3^1, t_1^2), (t_1^1, v_1^2), (t_1^1, v_2^2), (t_1^1, v_3^2), (g_3^1, g_1^2), (g_3^1, g_2^2), (g_3^1, g_3^2), (v_4^1, t_1^2), (t_3^1, v_1^2), (t_3^1, v_2^2), (t_3^1, v_3^2)$ |

TABLE II

Adjacent Sub-contacts of $(g_1^1, g_1^2)$

| Type | Adjacent sub-contacts |
|---|---|
| 1 | $(v_1^1, t_1^2), (v_1^1, t_3^2), (v_2^1, t_1^2), (v_2^1, t_3^2)$ |
| 2 | $(v_1^1, t_1^2), (v_1^1, t_2^2), (v_2^1, t_1^2), (v_2^1, t_2^2)$ |
| 3 | $(v_1^1, t_1^2), (v_2^1, t_1^2), (g_1^1, g_2^2)$ |
| 4 | $(v_1^1, t_1^2), (v_1^1, t_2^2), (v_1^1, t_3^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (v_2^1, t_3^2)$ |
| 5 | $(v_1^1, t_1^2), (v_1^1, t_2^2), (v_2^1, t_1^2), (v_2^1, t_2^2), (g_1^1, g_2^2)$ |
| 6 | $(v_1^1, t_2^2), (v_2^1, t_1^2)$ |
| 7 | $(v_1^1, t_1^2), (v_2^1, t_1^2), (g_1^1, g_2^2), (g_1^1, g_3^2)$ |

$DS(v_i, t_j)$, $DS(t_i, t_j)$, and $DS(e_i, e_j)$ indicate the set of distances between $v_i$ and three vertices of $t_j$, the set of distances between three vertices of $t_i$ and three vertices of $t_j$, and the set of distances between two vertices of $e_i$ and two vertices of $e_j$, respectively. A *DS* has a maximum distance and a minimum distance. If sub-contacts of a contact state satisfy (3), we assume that the contact state is feasible. However, these rules are only necessary conditions for feasible contact states. They do not remove all infeasible contact states.

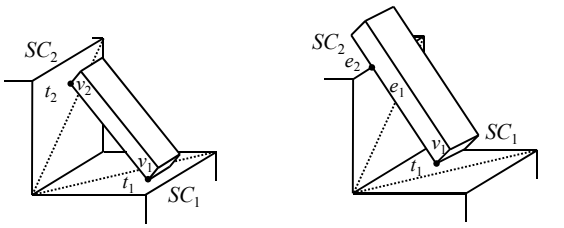



Fig. 11. Example of contact states of (a) {*v*-*t*, *v*-*t*} and (b) {*v*-*t*, *e*-*e*}.

Whenever a feasible contact state is generated, the contact state and its transitions are registered into the contact state graph. Repetition of this procedure makes the contact state graph to reach the target contact state ($CS_{target}$) from initial contact state ($CS_{initial}$), which implies that at least an assembly sequence to the $CS_{target}$ is guaranteed. Fig. 12 shows a contact state graph. Each node represents a contact state, and unidirectional arcs are given between contact states downwardly. The weight ($w$) of the arcs is the distance between the centroids of sub-contacts composing two contact states. For example, the *v*-*t* contact and the *e*-*e* contact in Fig. 13 have the distance between the centroid of the *t* and the center of the *e*. The $w$ is determined as the maximum among the distances between the sub-contacts.

$$
\begin{aligned}
&\text{(i) } \{(v_i^1, t_p^2), (v_j^1, t_q^2)\}: \\
&\min DS(t_p^2, t_q^2) < DS(v_i^1, v_j^1) < \max DS(t_p^2, t_q^2). \\
&\text{(ii) } \{(v_i^1, t_p^2), (t_j^1, v_q^2)\}: \\
&\{\min DS(v_i^1, t_j^1) < \max DS(v_q^2, t_p^2) < \max DS(v_i^1, t_j^1)\} \text{ or} \\
&\{\min DS(v_q^2, t_p^2) < \max DS(v_i^1, t_j^1) < \max DS(v_q^2, t_p^2)\}. \\
&\text{(iii) } \{(v_i^1, t_p^2), (e_j^1, e_q^2)\}: \\
&\{\min DS(t_p^2, e_q^2) < \max DS(v_i^1, e_j^1)\} \text{ and} \\
&\{\min DS(v_i^1, e_j^1) < \max DS(t_p^2, e_q^2)\}. \\
&\text{(iv) } \{(e_i^1, e_p^2), (e_j^1, e_q^2)\}: \\
&\{\min DS(e_i^1, e_j^1) < \max DS(e_p^2, e_q^2)\} \text{ and} \\
&\{\min DS(e_p^2, e_q^2) < \max DS(e_i^1, e_j^1)\}.
\end{aligned}
\tag{3}
$$

### B. Sequence Planning

The contact state graph can have a lot of assembly sequences from $CS_{initial}$ to $CS_{target}$. The sequence of minimum nodes is often regarded as an optimal assembly sequence and it can be obtained by shortest path algorithm [11]. However, it cannot be optimal if there is long distance between contact states. In this paper, the assembly sequence is planned by considering both of shortest nodes and shortest distance. The $w$ of the contact state graph is a measure to estimate the traveling distance of a robot which grasps an object. Hence geometric constraints are not considered to find an optimal sequence. We propose a cost function ($C$) which considers both of the number of nodes and the distance of arcs in (4).

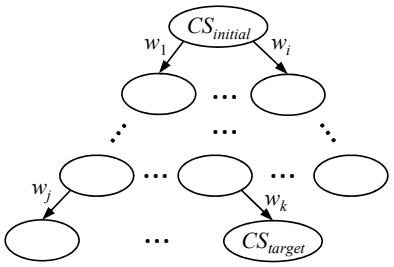



Fig. 12. Contact state graph.

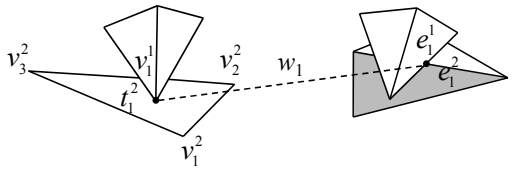



Fig. 13. Distance ($w$) between elements of two sub-contacts.

The cost function is computed in the same ratio of the number of nodes and the distance of arcs. The assembly sequence with minimized $C$ is optimal.

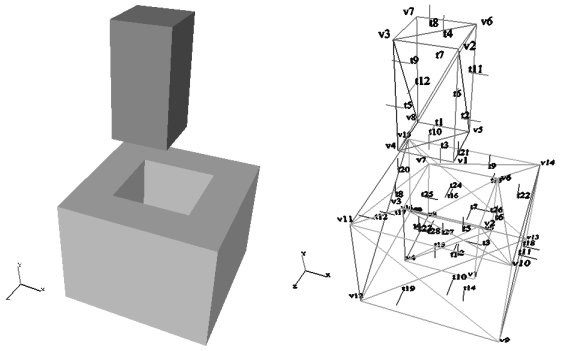
$$
C = \frac{\text{number of sequence}}{\text{total number of nodes}} + \frac{\text{distance of sequence}}{\text{total distance of arcs}}. \tag{4}
$$

### C. Computational Complexity

A polyhedral object with $m$ vertices can be triangulated in $O(m\log m)$ time [10]. Let the number of sub-contacts is $n$, then the computational complexity is $O(n)$ for one step to generate the contact states. The contact states generation is repeated in $n$ times for the worst case until it reaches $CS_{target}$. Therefore, the worst-case running time of the proposed method is $O(n^2)$. It is the same as the methods proposed by Hirai [5] and Xiao [7]. For Hirai's method, the additional computation of $l$ times is required to give the transitional connections between contact states. So its computational complexity is actually $O(n^2 l)$ considering the additional $l$ times computation. Xiao applies infinitesimal motions to find new contact states. If it needs $k$ steps to reach a new contact state, the computational complexity is $O(n^2 k)$. Therefore, the proposed method is more efficient than the other methods.

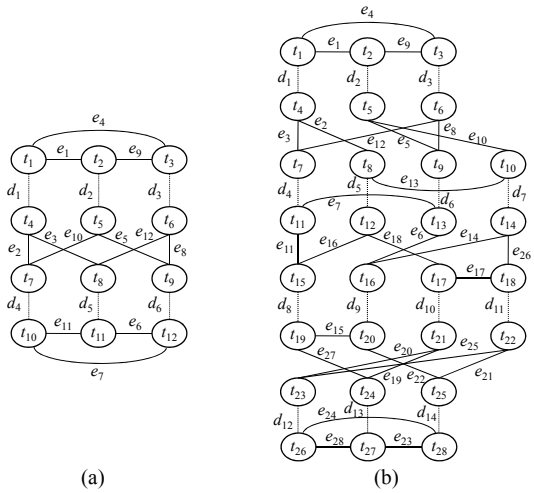
## IV. Case Study

A square peg-in-hole assembly of Fig. 14(a) is studied to verify the proposed method. First we define the topologies and geometries of the objects like the table of Fig. 2. The hierarchical structures of adjacent vertices and edges are constructed for peg and hole. Then all faces of each polyhedral object are triangulated according to the procedure described in Section II-B as shown in Fig. 14(b). Fig. 15 shows the structures of adjacent triangles. If $CS_{initial}=\{(v_1, t_{19})\}$ is given, the adjacent elements of $v_1$ are selected as $\{v_2, v_4, v_5\}$, $\{e_1, e_4, e_9\}$, and $\{t_1, t_2, t_3\}$ from Fig. 15(a). Also the adjacent elements of $t_{19}$ are selected as $\{v_7, v_{11}, v_{15}\}$, $\{d_8, e_{15}, e_{27}\}$, and $\{t_{15}, t_{20}, t_{24}\}$ from Fig. 15(b). Since the type of



(a) (b)

Fig. 14. Two polyhedral objects for peg-in-hole insertion: (a) before triangulation and (b) after triangulation.



(a) (b)

Fig. 15. Structure of adjacent triangles: (a) peg object and (b) hole object.

adjacent triangles $\{t_{15}, t_{20}, t_{24}\}$ is {B, A, B}, we use Type-6 of Table I to obtain adjacent sub-contacts. Contact states are generated by combination of $CS_{initial}$ and the adjacent sub-contacts. The feasibility of each contact state is checked by (3) before new contact states are registered on the contact state graph. The contact state graph is developed until a contact state reaches at $CS_{target}=\{(v_4, t_7), (v_1, t_6)\}$ in Fig. 16. Finally, the optimal sequence is obtained with $\{(v_1, t_{19})\}$, $\{(v_1, t_{15})\}$, $\{(e_4, e_{11})\}$, $\{(v_4, t_{11})\}$, $\{(v_4, t_7)\}$, and $\{(v_4, t_7), (v_1, t_6)\}$ is obtained by applying the cost function (4).

## V. Conclusion

This paper proposes a framework to generate contact states automatically. The vertex-triangle contact instead of the vertex-face contact is newly defined as a sub-contact of the contact state. The vertex-triangle contact can differentiate contact states more clearly than the vertex-face contact since even concave or hollow faces can be handled with triangles which are convex. A new hierarchy is invented and actively used in generation of contact states, where it consists of vertices, edges or diagonals, and triangles. The adjacent relations can be identified by using the hierarchy. The hierarchy is also used to effectively decrease the search range to find adjacent sub-contacts. The contact state graph is

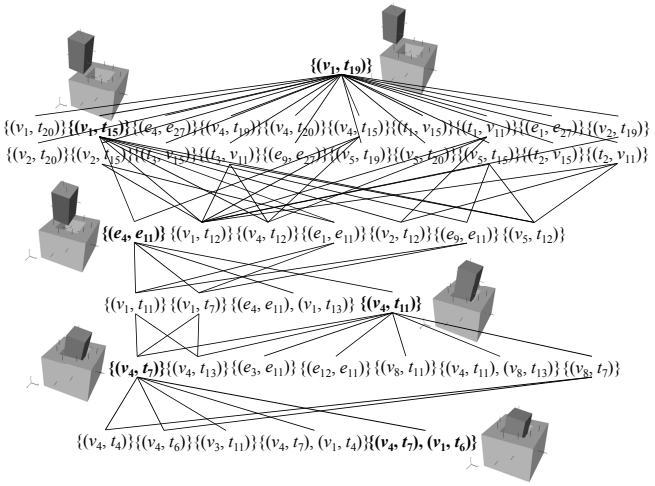



Fig. 16. The optimal sequence from $CS_{initial}=\{(v_1, t_{19})\}$ to $CS_{target}=\{(v_4, t_7), (v_1, t_6)\}$.

evolved from an initial contact state to a target contact state. If a whole contact state graph is a contact state graph consisting of all possible contact states and their transitions, the contact state graph from the proposed method is a subset of the whole one. However the proposed framework guarantees at least a realizable sequence for the assembly task. In near future, we expect to implement the proposed framework for practical assembly tasks of polyhedral objects.

## References


[1] P. Sikka and B. McCarragher, "Monitoring Contact Using Clustering and Discriminant Functions," in *Proc. of the IEEE Int. Conf. Robotics and Automation*, April 1996, Minneapolis, USA, pp. 1351–1356.

[2] A. Fox and S. Hutchinson, "Exploiting Visual Constraints in the Synthesis of Uncertainty-Tolerant Motion Plans," *IEEE Trans. Robotics and Automation*, vol. 11, no. 1, pp. 56–71, 1995.

[3] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Control of Two Industrial Robots for Parts Mating," in *Proc. of the IEEE Int. Conf. on Control Applications*, Sep. 1998, Trieste, Italy, pp. 562–566.

[4] H. Qiao and S. K. Tso, "Three-step Precise Robotic Peg-hole Insertion Operation with Symmetric Regular Polyhedral Objects," *International Journal of Production Research*, vol. 37, no. 15, pp. 3541–3563, 1999.

[5] S. Hirai, *Analysis and planning of manipulation using the theory of polyhedral convex cones*, Ph.D. dissertation, Kyoto University, Japan, 1991.

[6] S. Y. Chung and D. Y. Lee, "An augmented Petri net for modelling and control of assembly tasks with uncertainties," *Int. J. Computer Integrated Manufacturing*, vol. 18, no. 2–3, pp. 170–178, March–May 2005.

[7] J. Xiao and X. Ji, "Automatic generation of high-level contact state space," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 584–606, 2001.

[8] E. Staffetti, L. Ros, and F. Thomas, "A Simple Characterization of the Infinitesimal Motions Separating General Polyhedra in Contact," in *Proc. of the IEEE Int. Conf. Robotics and Automation*, May 1999, Detroit, USA, pp. 571–577.

[9] P. Tang and J. Xiao, "Automatic Generation of High-level Contact State Space between 3D Curved Objects," *The International Journal of Robotics Research*, vol. 27, no. 7, pp. 832–854, 2008.

[10] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed., Springer-Verlag, 2000, ch. 3.

[11] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., The MIT Press, 2001, ch. 24.