JOURNAL OF COMPUTER AND SYSTEM SCIENCES 29, 36–47 (1984)

An $n \log n$ Algorithm for Determining the Congruity of Polyhedra

KŌKICHI SUGIHARA

Department of Information Science, Faculty of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Japan 464

Received May 25, 1982; revised October 4, 1983

This paper describes an algorithm for determining whether two polyhedra are congruent. The asymptotic time complexity of the algorithm is bounded by a constant times $n \log n$ where n is the number of edges of the polyhedra. It is also shown that under some conditions the problem of partial congruity can be solved in $O(n^2)$ time.

1. INTRODUCTION

Because of recent development of computational geometry, many new and useful results have been obtained for the processing of geometric information in a two-dimensional space. In the case of three-dimensional geometry, on the other hand, we can find only a few results, such as the construction of a convex hull [2, 17], the construction of a plane which separates two families of points [15], and the detection of interference of two polyhedra [5, 14].

In the present paper we consider another computational problem in a three-dimensional space, the problem of determining whether two polyhedra are congruent. This problem arises out of several fields of engineering. In CAD systems for the design of mechanical parts [6, 8], we sometimes have to judge whether a part fits a hole of another part. In scene analysis using multi-view techniques [3, 19], we have to identify objects by searching for prototypes that have the same shapes as the observed objects. These problems are reduced to the problem of recognizing the congruity of polyhedra.

Vertices and edges of a polyhedron form an undirected simple graph. Hence, it seems natural to expect that we may be able to use some techniques for graph isomorphism in order to construct an algorithm for the congruity of polyhedra. The graph isomorphism in a general case is a hard problem [7]. For planar graphs, however, polynomial-order algorithms are known. Especially for triply connected planar graphs, Weinberg [20] found an algorithm of $O(n^2)$, Hopcroft and Tarjan [10] of $O(n \log n)$, and Hopcroft and Wong [11] of $O(n)$.

In the present paper we shall show that the technique in Hopcroft and Tarjan [10] can be applied to the congruity of polyhedra, and thus construct an $O(n \log n)$ algorithm for determining whether two polyhedra are congruent. The

0022-0000/84 $3.00

Hopcroft–Tarjan algorithm can be applied only to triply connected planar graphs, whereas the present algorithm works even if the graphs composed of the vertices and the edges of the polyhedra are not planar or triply connected. This is because the Hopcroft–Tarjan algorithm is essentially for isomorphism of embedded graphs, and a polyhedron can be thought of as a graph embedded on the surface of a solid object. Thus the present algorithm is available also to polyhedra that are neither convex (if a polyhedron is convex, the associated graph is triply connected [4]) nor homeomorphic to a sphere (if a polyhedron is homeomorphic to a sphere, the associated graph is planar).

We shall also show that we can judge in $O(n^2)$ time whether a part of a polyhedron is congruent with some part of another polyhedron. This result seems interesting when we recall that the subgraph isomorphism problem is *NP*-complete [9].

2. Definitions and Notations

Let $\mathbb{R}^3$ denote a three-dimensional Euclidean space. A *polyhedron* is a closed subset of $\mathbb{R}^3$ with finite volume bounded by a finite number of planar polygons. We call the polygons *faces*, the line segments shared by two faces *edges*, and the points shared by three or more faces *vertices*, respectively, of the polyhedron. For a polyhedron P, let $F(P)$, $E(P)$, $V(P)$ denote the set of faces, that of edges, and that of vertices, respectively, of P. Furthermore, let ∂P denote the boundary of P, that is,

$$\partial P = \bigcup \{f: f \in F(P)\}.$$

For any two points x and y in $\mathbb{R}^3$, let $d(x, y)$ denote the Euclidean distance between x and y. A mapping T of $\mathbb{R}^3$ onto itself is said to be *isometric* if $d(x, y) = d(T(x), T(y))$ for any $x, y \in \mathbb{R}^3$. An isometric mapping T is said to be *orientation-preserving* if T maps any right-handed coordinate system onto some right-handed coordinate system. For any subset X of $\mathbb{R}^3$, we define

$$T(X) = \{y: y = T(x) \text{ for some } x \text{ in } X\}.$$

Let P_1 and P_2 be two polyhedra. If an isometric orientation-preserving mapping T satisfies $P_2 = T(P_1)$, then the mapping obtained when we restrict the domain of T to P_1 is called a *congruent mapping* of P_1 onto P_2. P_1 and P_2 are said to be *congruent* if there exists a congruent mapping of P_1 onto P_2.

For a polyhedron P, let $FG(P)$ denote the undirected graph whose node set is $F(P)$ and whose arc set is defined by

$$\{\{f, f'\}: f, f' \in F(P), \text{ the faces } f \text{ and } f' \text{ share a common edge}\}.$$

We call $FG(P)$ the *face-edge graph* of the polyhedron P. For any point x in $\mathbb{R}^3$, we define ball $B(x; r)$ of radius r at x by

$$B(x; r) = \{y: y \in \mathbb{R}^3, d(x, y) \leqslant r\}.$$

In the present paper we consider polyhedra P's that satisfy the following three conditions.

Condition 1. The face-edge graph $FG(P)$ is connected.

This condition excludes those polyhedra in which two parts are disconnected or they touch only at vertices.

Condition 2. For any point x in ∂P, there exists a positive real $t(x)$ such that $B(x; r) \cap P - \partial P$ is nonempty and simply connected for any $0 < r \leqslant t(x)$.

This condition excludes some "unusual" polyhedra. Since $B(x; r) \cap P - \partial P$ is nonempty, the polyhedron P must be thick, that is, P can not be something made of "thin" paper. Furthermore, since $B(x; r) \cap P - \partial P$ is simply connected, P can not have unusual vertices or edges such as those in Fig. 1. Therefore, if P satisfies Condition 2, we can distinguish between the outside and the inside of a face, and we can in a unique order visit all the faces and edges touching a vertex counterclockwise around it from an arbitrarily chosen initial face or edge.

Condition 3. For any face f of P, $f - \bigcup \{e: e \in E(P)\}$ is simply connected.

This condition excludes faces with holes (except for holes that touch the boundary of the faces). Hence, if P satisfies Condition 3, we can travel from any edge on f to any other edge on f along some sequence of edges on f.

For a polyhedron P, let $G(P) = (V(P), E(P))$ be the undirected graph having the node set $V(P)$ and the arc set $E(P)$, where $e \in E(P)$ is considered as an arc connecting the two end vertices of the edge e. We call $G(P)$ the *vertex-edge graph* of G. Every node of $G(P)$ has at least three arcs, because three or more faces of P meet at every vertex. Let $\bar{G}(P) = (V(P), \bar{E}(P))$ be the directed graph that is obtained when we replace every arc of $G(P)$ with the two directed arcs of each direction, that is,

$$\bar{E}(P) = \{(v_1, v_2), (v_2, v_1): \{v_1, v_2\} \in E(P)\}.$$

For any $e = (v_1, v_2)$ in $\bar{E}(P)$, v_1 and v_2 are called the *initial* node and the *terminal*

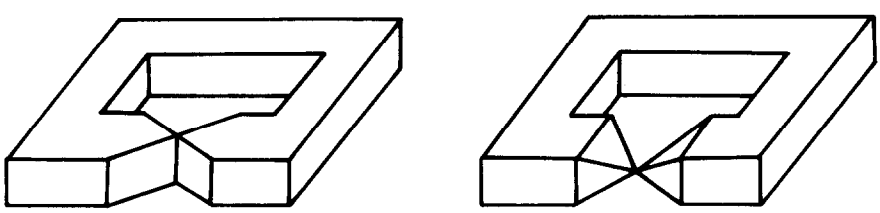

FIG. 1. Polyhedra which do not satisfy Condition 2.

node, respectively, of the directed arc e, and the reversal (v_2, v_1) of e is denoted by e^r. A sequence $((v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n))$ of elements of $\bar{E}(P)$ is called a *directed path* starting with the edge (v_1, v_2) and ending with the edge (v_{n-1}, v_n).

The graph $\bar{G}(P)$ defined above is an "abstract" graph. In what follows, however, we will mean by $\bar{G}(P)$ the actual three-dimensional configuration composed of the vertices and the edges of P, and will use the terms "vertices" and "edges" to indicate the "nodes" and "arcs" of $\bar{G}(P)$. Therefore, for example, a vertex of $\bar{G}(P)$ has its location in $\mathbb{R}^3$, and an edge of $\bar{G}(P)$ has its length. Moreover, for any $v \in V(P)$ and $e = (v, v') \in \bar{E}(P)$, we can uniquely define the order in which we, starting with the edge e, visit counterclockwise all the edges out of the vertex v.

For any $e \in \bar{E}(P)$, let $f_R(e)$ denote the face to the right of the directed edge e, and $f_L(e)$ the face to the left of e (see Fig. 2). By $H_R(e)$ and $H_L(e)$, respectively, we indicate the oriented planar surfaces on which $f_R(e)$ or $f_L(e)$ lies, where an "oriented" surface means that the one side of the surface is labeled as the outside and the other as the inside. Let e be any edge of $\bar{G}(P)$, and e' be any edge going out of the terminal vertex of e. We say that e' is to the *immediate right* of e if $f_R(e) = f_R(e')$, and to the *immediate left* of e if $f_L(e) = f_L(e')$. We represent the edge to the immediate right of e by $g_R(e)$, and that to the immediate left of e by $g_L(e)$. Thus g_R and g_L are one-to-one mappings of $\bar{E}(P)$ onto itself, and consequently the inverses g_R^{-1} and g_L^{-1} are also one-to-one mappings of $\bar{E}(P)$ onto itself, as shown in Fig. 2.

For an edge $e \in \bar{E}(P)$, let $l(e)$ denote the length of the edge e, and $\psi(e)$ the angle between the two side faces $f_R(e)$ and $f_L(e)$ $(0 < \psi(e) \leqslant 2\pi,\ \psi(e) \neq \pi)$, where the angle is assumed to be measured in the inside of the polyhedron and hence $0 < \psi(e) < \pi$ means that the edge e forms a ridge and $\pi < \psi(e) < 2\pi$ means a valley. Furthermore, let $\theta_R(e)$ be the angle between e and $g_R(e)$ and $\theta_L(e)$ the angle between e and $g_L(e)$ $(0 < \theta_R(e), \theta_L(e) < 2\pi)$. We define $\lambda(e)$ by

$$\lambda(e) = (l(e), \psi(e), \theta_R(e), \theta_L(e)).$$

A directed path $(e_1, \ldots, e_n)$ of $\bar{G}(P)$ is said to be *primary* if for each i, $1 \leqslant i \leqslant n-1$,

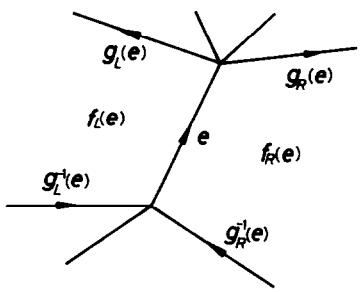

FIG. 2. Definition of g_R, g_L, and λ.

e_{i+1} is either to the immediate right of e_i or to the immediate left of e_i. For a primary path $p = (e_1,..., e_n)$, we define $\alpha(p) = (c_1,..., c_{n-1})$ by

$$c_i = \begin{cases} 1 & \text{if } e_{i+1} = g_R(e_i), \\ -1 & \text{if } e_{i+1} = g_L(e_i). \end{cases}$$

Note that $g_R(e) \neq g_L(e)$ for any $e \in \bar{E}(P)$ (because any vertex of $G(p)$ has three or more edges) and hence c_i is determined without ambiguity.

Suppose that P and P' are two polyhedra. Let $p = (e_1,..., e_n)$ and $p' = (e'_1,..., e'_n)$ be primary paths of $\bar{G}(P)$ and $\bar{G}(P')$, respectively. If $\alpha(p) = \alpha(p')$, then p' is said to be the path *corresponding* to p starting with the edge e'_1. An edge $e_1 \in \bar{E}(P)$ and an edge $e'_1 \in \bar{E}(P')$ are said to be *indistinguishable* if any primary path $p = (e_1,..., e_n)$ starting with e_1 and the path $p' = (e'_1,..., e'_n)$ corresponding to p starting with e'_1 satisfy $\lambda(e_i) = \lambda(e'_i)$ for any i, $1 \leqslant i \leqslant n$, and *distinguishable* if otherwise. Note that if $P = P'$, then to be indistinguishable is an equivalence relation in $\bar{E}(P)$, and thus the indistinguishability introduces an equivalence relation in edge set $\bar{E}(P_1) \cup \cdots \cup \bar{E}(P_s)$ of any number s of polyhedra $P_1,..., P_s$.

3. Basic Results

In this section we derive some results on which our algorithm is based.

Lemma 1. *Let P be a polyhedron satisfying Condition 2, and $p = (e_1,..., e_n)$ be any primary path of $\bar{G}(P)$. If the locations of e_1 and $H_R(e_1)$ in $\mathbb{R}^3$ and the values $\lambda(e_i)$, $1 \leqslant i \leqslant n$, are given, then the locations of e_n, $H_R(e_n)$, $H_L(e_n)$ in $\mathbb{R}^3$ can be determined uniquely.*

Proof. It suffices to show that if the locations of e_i and $H_R(e_i)$ are given, then the locations of e_{i+1}, $H_R(e_{i+1})$, $H_L(e_{i+1})$ are determined uniquely. Now suppose that the locations of e_i and $H_R(e_i)$ in $\mathbb{R}^3$ are given.

Case (i) Suppose that $e_{i+1} = g_R(e_i)$. Since $H_R(e_{i+1}) = H_R(e_i)$, the location of $H_R(e_{i+1})$ in $\mathbb{R}^3$ is uniquely determined. Noting that the angle between e_i and e_{i+1} is given by $\theta_R(e_i)$ (the third component of $\lambda(e_i)$) and the length of e_{i+1} by $l(e_{i+1})$ (the first component of $\lambda(e_{i+1})$), we can uniquely determine the location of e_{i+1} in $\mathbb{R}^3$. The location of $H_L(e_{i+1})$ is also determined because the angle between $H_R(e_{i+1})$ and $H_L(e_{i+1})$ is given by $\psi(e_{i+1})$ (the second component of $\lambda(e_{i+1})$).

Case (ii) Suppose that $e_{i+1} = g_L(e_i)$. From $\psi(e_i)$, we can determine the location of $H_L(e_i)$. Exchanging the right and the left in the case (i), we can easily see that the locations of e_{i+1}, $H_R(e_{i+1})$, $H_L(e_{i+1})$ in $\mathbb{R}^3$ are determined uniquely. Q.E.D.

Lemma 2. *Let P and P' be two polyhedra satisfying Condition 2. Let $p =$*

$(e_1,\ldots,e_n)$ be a primary path of $\bar{G}(P)$ and $p'=(e_1',\ldots,e_n')$ be the path corresponding to p starting with any edge e_1' of $\bar{G}(P')$ such that $\lambda(e_i)=\lambda(e_i')$ for each i, $1\leqslant i\leqslant n$. Then, the isometric orientation-preserving mapping T of $\mathbb{R}^3$ onto itself such that $e_1'=T(e_1)$ and $H_R(e_1')=T(H_R(e_1))$ also satisfies $e_n'=T(e_n)$, $H_R(e_n')=T(H_R(e_n))$, $H_L(e_n')=T(H_L(e_n))$.

Proof. From Lemma 1, the relative position of e_1, $H_{\mathrm{R}}(e_1)$, e_n, $H_{\mathrm{R}}(e_n)$, $H_{\mathrm{L}}(e_n)$ is the same as that of e_1', $H_{\mathrm{R}}(e_1')$, e_n', $H_{\mathrm{R}}(e_n')$, $H_{\mathrm{L}}(e_n')$. Hence we get the lemma. Q.E.D.

LEMMA 3. *Let P be a polyhedron that satisfies Conditions* 1, 2, *and* 3. *For any e, $e'\in\bar{E}(P)$, there exists a primary path of $\bar{G}(P)$ that starts with e and ends with either e' or $(e')^{\mathrm{r}}$.*

Proof. From Condition 1, there exists a sequence $f_1,\ldots,f_n$ of faces of P such that $f_1=f_{\mathrm{R}}(e)$, $f_n=f_{\mathrm{R}}(e')$ and for any i, $1\leqslant i\leqslant n-1$, f_i and f_{i+1} share a common edge, say, e_i (the direction of e_i will be given a little later). Therefore, we can construct a primary path from e to either e' or $(e')^{\mathrm{r}}$ in the following way. Starting with e, we first travel from edge to edge along the boundary of the face f_1 until we arrive at e_1 (at this point we determine the direction of e_1 in such a way that the direction coincides with the direction of our travel). Next we travel along the boundary of the face f_2 until we arrive at e_2 (at this point we determine the direction of e_2). In a similar manner, we can travel until we arrive at e_{n-1}. Note first that because of Condition 3 we can always go from e_{i-1} to e_i along the boundary of f_i. Note secondly that the path we have taken is primary because we always choose as the next edge either the edge to the immediate right or that to the immediate left. Starting with e_{n-1}, we travel along the boundary of the face f_n and eventually come across either e' or $(e')^{\mathrm{r}}$. Q.E.D.

THEOREM 4. *Let P_1 and P_2 be two polyhedra satisfying Conditions* 1, 2, *and* 3, *and let e_1 and e_2 be edges of $\bar{G}(P_1)$ and $\bar{G}(P_2)$, respectively. There exists a congruent mapping of P_1 onto P_2 that maps e_1 onto e_2 if and only if e_1 and e_2 are indistinguishable.*

Proof. If there exists a congruent mapping of P_1 onto P_2 which maps e_1 onto e_2, then e_1 and e_2 are obviously indistinguishable. Now, supppose that e_1 and e_2 are indistinguishable. Then, there exists a unique isometric orientation-preserving mapping T of $\mathbb{R}^3$ onto itself that satisfies $e_2=T(e_1)$ and $H_{\mathrm{R}}(e_2)=T(H_{\mathrm{R}}(e_1))$. Let e_3 be any edge of $\bar{G}(P_1)$. It follows from Lemma 3 that there is a primary path, say, p_1, starting with e_1 and ending with either e_3 or $(e_3)^{\mathrm{r}}$. Let p_2 be the path corresponding to p_1 starting with e_2, and let e_4 be the last edge of p_2. Since e_1 and e_2 are indistinguishable, it follows from Lemma 2 that $e_4=T(e_3)$, $H_{\mathrm{R}}(e_4)=T(H_{\mathrm{R}}(e_3))$, $H_{\mathrm{L}}(e_4)=T(H_{\mathrm{L}}(e_3))$. Since e_3 is arbitrary, T maps P_1 onto P_2 and hence P_1 and P_2 are congruent. Q.E.D.

4. Algorithm for Congruity Recognition

Theorem 4 has the same "form" as that in Hopcroft and Tarjan [10] on which they construct an algorithm for graph isomorphism. Therefore, we can use their technique for our problem. According to this policy we will in this section construct an algorithm for determining whether two polyhedra are congruent.

We use a random access machine [1] as the model of computation. We assume that the polyhedra P_1 and P_2 are described in some suitable data structure such as the one in [12] or [15], and hence for each $e \in \bar{E}(P_i)$ $(i = 1, 2)$ we can retrieve $g_R(e)$, $g_L(e)$, $g_R^{-1}(e)$, $g_L^{-1}(e)$, the coordinates of the initial and the terminal vertices of e, the surface equations on which $f_R(e)$ or $f_L(e)$ lies, etc., in $O(1)$ steps.

Based on Theorem 4, we can construct the following algorithm for determining whether two polyhedra P_1 and P_2 are congruent.

Algorithm A.

(1) Calculate $\lambda(e)$ for all e in $\bar{E}(P_1) \cup \bar{E}(P_2)$.

(2) Partition $\bar{E}(P_1) \cup \bar{E}(P_2)$ into blocks, say, $B(1)$, $B(2)$,..., $B(k)$ $(k \geqslant 1)$, in such a way that e and e' belong to the same block if and only if $\lambda(e) = \lambda(e')$. If there exists i, $1 \leqslant i \leqslant k$, such that either $B(i) \subseteq \bar{E}(P_1)$ or $B(i) \subseteq \bar{E}(P_2)$, then conclude that P_1 and P_2 are not congruent and stop the processing.

(3) Subdivide $B(1)$,..., $B(k)$ into blocks consisting of indistinguishable edges by the procedure SUBDIVIDE defined in Fig. 3. Let the resulting blocks be $B(1)$, $B(2)$,..., $B(l)$ $(l \geqslant k)$.

(4) If $B(1) \cap \bar{E}(P_1) \neq \varnothing$ and $B(1) \cap \bar{E}(P_2) \neq \varnothing$, then conclude that P_1 and P_2 are congruent. If otherwise, conclude that they are not congruent.

The procedure in Fig. 3 is written in pidgin ALGOL [1]. The procedure subdivides the initial blocks $B(1)$,..., $B(k)$ into blocks consisting of mutually indistinguishable edges. An outline of the procedure is the following. First, all pairs of form (i, D) are stored in stack WAIT, where i denotes a block number and D is either R or L (line 1), and l is set to be the number of blocks in the initial state (line 2). Elements of WAIT are used for checking whether the current partition of the edges is fine enough in the sense that each block consists of indistinguishable edges. An element (i, D) is selected and deleted from the stack WAIT (line 4), and MOVE is defined as a set of edges in such a way that e is in MOVE if and only if some edge in $B(i)$ is to the immediate right (if $D = \mathrm{R}$) or to the immediate left (if $D = \mathrm{L}$) of e (line 5). Then, MOVE is used as a reference set for checking whether a block contains mutually indistinguishable edges. If block $B(j)$ contains both an element belonging to MOVE and an element not belonging to MOVE (this implies that $B(j)$ contains mutually distinguishable edges), then l is replaced by $l + 1$, $B(j)$ is divided into two smaller blocks $B(j) \cap \mathrm{MOVE}$ and $B(j) - B(j) \cap \mathrm{MOVE}$, and they are renamed $B(l)$ and $B(j)$, respectively (lines 6–10). Moreover, if (j, D) is in WAIT, (l, D) is also added to WAIT for later checking, and if otherwise, either (j, D) or (l, D) that corresponds to a smaller block is added to WAIT (lines 11–13). Each time an element is added to

```
    procedure SUBDIVIDE
    begin
1:    WAIT ← {(1, R), (1, L),..., (k, R), (k, L)};
2:    l ← k;
3:    while WAIT is not empty do
      begin
4:      select and delete (i, D) from WAIT;
5:      MOVE ← {g_D^{-1}(e): e ∈ B(i)};
6:      for each j such that B(j) ∩ MOVE ≠ ∅ and B(j) ⊄ MOVE do
        begin
7:        l ← l + 1;
8:        create new block B(l);
9:        B(l) ← B(j) ∩ MOVE;
10:       B(j) ← B(j) − B(l);
11:       if (j, D) ∈ WAIT then add (l, D) to WAIT
          else
12:         if |B(j)| ≤ |B(l)| then add (j, D) to WAIT
13:         else add (l, D) to WAIT
        end
      end
    end
```

FIG. 3. Procedure SUBDIVIDE used in Step 3 of Algorithm A.

WAIT at line 12 or 13, the corresponding block is not greater than half of its previous size. Hence, for each edge e, a block containing e is used for checking at most $O(\log n)$ times, and consequently the total time complexity of the procedure is of $O(n \log n)$.

LEMMA 5. *Suppose that the blocks* $B(1),\ldots, B(k)$ *obtained in Step* 2 *of Algorithm* A *are given to the procedure* SUBDIVIDE *as input. Then the final output* $\{B(1),\ldots, B(l)\}$ *from the procedure* SUBDIVIDE *coincides with the partition of* $\bar{E}(P_1) \cup \bar{E}(P_2)$ *into mutually indistinguishable edges. Furthermore, if* $|E(P_1)| = |E(P_2)| = n$, *then procedure* SUBDIVIDE *executes the processing in* $O(n \log n)$ *time.*

We omit the proof, because it is the same as that given by Hopcroft and Tarjan [10]. (We can also find the proof in a more general framework in Section 4.13 of Aho *et al.* [1].)

THEOREM 6. *Suppose that polyhedra* P_1 *and* P_2 *satisfy Conditions* 1, 2, *and* 3, *and* $|E(P_1)| = |E(P_2)| = n$. *Then, Algorithm* A *correctly determines whether* P_1 *and* P_2 *are congruent, and its time complexity is* $O(n \log n)$.

Proof. The algorithm terminates in either Step 2 or Step 4. If $B(i) \subseteq \bar{E}(P_1)$ for some i in Step 2, edges in $B(i)$ are distinguishable from any edge of $\bar{G}(P_2)$ and hence from Theorem 4 the two polyhedra are not congruent. So is the case in which $B(i) \subseteq \bar{E}(P_2)$. It follows from Lemma 5 that in Step 4 an edge in $B(1)$ is indistinguishable from any other edge in $B(1)$ and is distinguishable from any edge in the other blocks

$B(2),\ldots, B(l)$. Hence from Theorem 4, P_1 and P_2 are congruent if and only if $B(1)$ contains both an edge in $\bar{E}(P_1)$ and one in $\bar{E}(P_2)$. Therefore, the judgment in Step 4 is also correct.

Step 1 takes no more than $O(n)$ time, because we assume that for each $e \in \bar{E}(P_1) \cup \bar{E}(P_2)$ we can calculate $\lambda(e)$ in a constant time. Using any technique for sorting four-character words in the lexicographic order, we can execute Step 2 in $O(n \log n)$ time [1]. From Lemma 5, Step 3 can be executed in $O(n \log n)$ time. Step 4 takes $O(n)$ time. Therefore, the total amount of time needed in Algorithm A is $O(n \log n)$. Q.E.D.

Remark 1. The initial partition of $\bar{E}(P_1) \cup \bar{E}(P_2)$ at Step 2 can be any one provided that it is not coarser than the partition defined by $\lambda(e)$ and is not finer than the partition into indistinguishable edges. Hence, in order to shorten the time for the later processing we can also use miscellaneous information other than $\lambda(e)$. For example, for any $e \in \bar{E}(P)$ we define 8-tuple $\lambda'(e) = (m_1(e), m_2(e), m_3(e), m_4(e), \lambda(e))$, where $m_1(e)$, $m_2(e)$ are the degrees of the initial vertex and the terminal vertex, respectively, of e, and $m_3(e)$, $m_4(e)$ are the numbers of edges belonging to the boundaries of $f_R(e)$ and $f_L(e)$, respectively. If we partition $\bar{E}(P_1) \cup \bar{E}(P_2)$ according to $\lambda'(e)$ rather than $\lambda(e)$, we in general obtain a finer partition as the initial partition for the algorithm, and thus improve the processing time.

Remark 2. In practical situations data usually contain errors due to digitization, etc., and hence $\lambda(e) = \lambda(e')$ does not make sense. We have to judge whether $\lambda(e)$ and $\lambda(e')$ are "nearly equal" under some criterion. For example, we can judge $\lambda(e)$ and $\lambda(e')$ are nearly equal if and only if

$$\max(|l(e) - l(e')|, |\psi(e) - \psi(e')|, |\theta_R(e) - \theta_R(e')|, |\theta_L(e) - \theta_L(e')|) \leqslant \varepsilon,$$

where ε is a small positive constant. However, being nearly equal itself is not an equivalence relation, and hence in order to define the initial blocks $B(1),\ldots, B(k)$ we have to use the transitive closure of "being nearly equal." Note that this revision of the algorithm does not affect the time complexity estimated in Theorem 6, because we can still make use of the techniques for sorting elements in the lexicographic order if we adopt the above criterion for being nearly equal. When, on the other hand, errors in the data are not small, we have to use other approaches such as the one proposed by Ikeuchi [13], because the abstract graph $G(P)$ itself may contain errors.

Remark 3. The procedure SUBDIVIDE decomposes the initial blocks into the blocks of indistinguishable edges. However, it is not always necessary to execute the whole processing: if we come across on the way of the execution a block which is contained by either $\bar{E}(P_1)$ or $\bar{E}(P_2)$, then we can conclude that the polyhedra are not congruent and stop the processing. This check should be executed between line 10 and line 11 of the procedure SUBDIVIDE.

Remark 4. The algorithm can be modified to partition a family of three or more polyhedra $P_1, P_2,\ldots, P_s$ into subfamilies consisting of congruent polyhedra. For this

purpose we use $\bar{E}(P_1) \cup \bar{E}(P_2) \cup \cdots \cup \bar{E}(P_s)$ instead of $\bar{E}(P_1) \cup \bar{E}(P_2)$ as an object to be partitioned. Moreover, if we use $\bar{E}(P_1)$ instead of $\bar{E}(P_1) \cup \bar{E}(P_2)$, we can enumerate all of the congruent mappings of P_1 onto itself.

Remark 5. The algorithm is also used for determining whether two polyhedra are similar in the sense that they differ only in their sizes. The only thing we have to do for this purpose is to normalize the sizes of the two polyhedra. This can be done in $O(n)$ time by expanding or shrinking one of the polyhedra so that the length of its longest edge coincides with that of the other.

5. On Recognition of Partial Congruity

The partial congruity problem is to determine whether a part of a polyhedron is congruent with some part of another polyhedron. This problem often arises in scene analysis, where we have to identify objects from information only about "visible" parts of the objects (see, for example, Sugihara [18] and Oshima and Shirai [16]).

In the case of graphs the subgraph isomorphism problem is *NP*-complete, while a special case of it, the graph isomorphism problem, is not known to be *NP*-complete or not (Garey and Johnson [9]). It seems that the hardness of the subgraph isomorphism problem is partly due to the fact that we are not given information about what parts are deleted from the graphs. In the case of polyhedra, on the other hand, we are usually given information about what parts are "invisible." Making use of this information, we can construct an efficient algorithm for the partial congruity problem.

Suppose that the face set $F(P)$ of a polyhedron P is partitioned into $F^{\mathrm{v}}(P)$, called the set of *visible* faces, and $F^{\mathrm{i}}(P)$, called the set of *invisible* faces, and that we are given data only about the visible faces. Let $E^{\mathrm{v}}(P)$ and $V^{\mathrm{v}}(P)$ be the set of edges and that of vertices, respectively, which belong to boundaries of the visible faces, that is,

$$E^{\mathrm{v}}(P) = \{a: a \in E(P), a \in f \text{ for some } f \in F^{\mathrm{v}}(P)\},$$

$$V^{\mathrm{v}}(P) = \{v: v \in V(P), v \in f \text{ for some } f \in F^{\mathrm{v}}(P)\}.$$

The elements of $E^{\mathrm{v}}(P)$ are called *visible* edges and those of $V^{\mathrm{v}}(P)$ *visible* vertices. Let P^* be another polyhedron, called a *model*, and let X be a subset of $F(P^*)$. If an isometric orientation-preserving mapping T of $\mathbb{R}^3$ onto itself maps $\bigcup \{f: f \in F^{\mathrm{v}}(P)\}$ onto $\bigcup \{f: f \in X\}$ and moreover if it maps the outside of each face in $F^{\mathrm{v}}(P)$ onto the outside of the corresponding face in X, then the mapping obtained when we restrict the domain of T to $F^{\mathrm{v}}(P)$ is called a *partial congruent mapping* of $F^{\mathrm{v}}(P)$ onto X. We say that P is *partly congruent* with P^* with respect to $F^{\mathrm{v}}(P)$ if there exists a subset X of $F(P^*)$ and a partial congruent mapping of $F^{\mathrm{v}}(P)$ onto X.

We restrict our consideration to a polyhedron P whose visible part satisfies

Condition 4. The subgraph of the face-edge graph $FG(P)$ induced by the node set $F^{\mathrm{v}}(P)$ is connected.

Let $\bar{G}^v(P) = (V^v(P), \bar{E}^v(P))$ be the directed graph with the node set $V^v(P)$ and the arc set

$$\bar{E}^v(P) = \{(v_1, v_2), (v_2, v_1): \{v_1, v_2\} \in E^v(P)\}.$$

We partition $\bar{E}^v(P)$ into

$$\bar{E}^v_B(P) = \{e: \text{both } f_R(e) \text{ and } f_L(e) \text{ are visible}\},$$
$$\bar{E}^v_R(P) = \{e: f_R(e) \text{ is visible but } f_L(e) \text{ is not}\},$$
$$\bar{E}^v_L(P) = \{e: f_L(e) \text{ is visible but } f_R(e) \text{ is not}\}.$$

For any $e \in \bar{E}^v(P)$, let us define $\mu(e)$ by

$$\mu(e) = \begin{cases} (l(e), \psi(e), \theta_R(e), \theta_L(e)) & \text{if } e \in \bar{E}^v_B(P), \\ (l(e), *, \theta_R(e), *) & \text{if } e \in \bar{E}^v_R(P), \\ (l(e), *, *, \theta_L(e)) & \text{if } e \in \bar{E}^v_L(P), \end{cases}$$

where * means that the value is not defined. For the model P^*, we define $\mu(e)$ by considering all the faces as visible. Furthermore we define that $\mu(e_1) = \mu(e_2)$ if and only if they coincide componentwise except for the components in which $\mu(e_1)$ or $\mu(e_2)$ has *. A primary path $p = (e_1, ..., e_n)$ is called a *visible* primary path if the face f_i on which both e_i and e_{i+1} lie is visible for any i, $1 \leqslant i \leqslant n-1$.

An edge $e_1 \in \bar{E}^v(P)$ is said to be *partly indistinguishable with respect to* $F^v(P)$ from $e_1' \in \bar{E}(P^*)$ if any visible primary path $p = (e_1, ..., e_n)$ starting with e_1 and the path $p' = (e_1', ..., e_n')$ corresponding to p starting with e_1' satisfy $\mu(e_i) = \mu(e_i')$ for any i, $1 \leqslant i \leqslant n$. Then, we get the following theorem.

THEOREM 7. *Suppose that two polyhedra P, P^* satisfy Conditions* 1, 2, *and* 3 *and the set $F^v(P)$ of visible faces of P satisfies Condition* 4. *Let e be any edge in $\bar{E}^v(P)$ and e' be any edge in $\bar{E}(P^*)$. There exist a subset X of $F(P^*)$ and a partial congruent mapping of $F^v(P)$ onto X that maps e onto e' if and only if e is partly indistinguishable with respect to $F^v(P)$ from e'.*

The proof is very similar to that of Theorem 4, and hence we omit it.

Using this theorem, we can determine whether P is partly congruent with P^* with respect to $F^v(P)$ in the following way. First, we select and fix an arbitrary element e_0 of $\bar{E}^v(P)$. Next, for each element e' of $\bar{E}(P^*)$ we see if e_0 is partly indistinguishable with respect to $F^v(P)$ from e'. If we happen to come across an edge $e^* \in \bar{E}(P^*)$ from which e_0 is partly indistinguishable, we conclude that P is partly congruent with P^* and end the processing. If e_0 is not partly indistinguishable from any edge in $\bar{E}(P^*)$, then we conclude that P is not partly congruent with P^*. The test of whether or not e_0 is partly indistinguishable from e' can be executed by usual graph search techniques such as the breadth-first search in $O(|E^v(P)|)$ time. Therefore, the time complexity is of $O(mn)$, where $n = |E(P^*)|$ and $m = |E^v(P)|$.

ACKNOWLEDGMENTS

The author would like to express his appreciation to Professor Masao Iri of the University of Tokyo and Professor Noboru Sugie of Nagoya University for valuable discussions.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
2. A. APPEL AND P. M. WILL, Determining the three-dimensional convex hull of a polyhedron, *IBM J. Res. Develop.* (November, 1976), 590–601.
3. M. ASADA AND S. TSUJI, Representation of three-dimensional motion in dynamic scenes, *Comput. Vision Gr. Image Process.* **21** (1983), 118–144.
4. D. BARNETTE AND B. GRÜNBAUM, On Steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs, *in* "The Many Facets of Graph Theory," Lecture Notes in Mathematics No. 110, pp. 27–40, Springer-Verlag, Berlin/New York, 1969.
5. J. W. BOYSE, Interference detection among solids and surfaces, *Comm. ACM* **22** (1979), 3–9.
6. I. C. BRAID,. The synthesis of solids bounded by many faces, *Comm. ACM* **18** (1975), 209–216.
7. D. G. CORNEIL AND D. G. KIRKPATRICK, A theoretical analysis of various heuristics for the graph isomorphism problem, *SIAM J. Comput.* **9** (1980), 281–297.
8. J. ENCARNACAO (Ed.), "Computer Aided Design," Lecture Notes in Computer Science No. 89, Springer-Verlag, Berlin/New York, 1980.
9. M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability," Freeman, San Francisco, 1979.
10. J. E. HOPCROFT AND R. E. TARJAN, A $V \log V$ algorithm for isomorphism of triconnected planar graphs, *J. Comput. System Sci.* **7** (1973), 323–331.
11. J. E. HOPCROFT AND J. K. WONG, Linear time algorithm for isomorphism of planar graphs, *in* "Proceedings, 6th Annual ACM Symposium on Theory of Computing, 1974," pp. 172–184.
12. M. HOSAKA AND F. KIMURA, An interactive geometrical design system with handwriting input, *in* "Information Processing 77," pp. 167–172, North-Holland, Amsterdam, 1977.
13. K. IKEUCHI, Recognition of 3-D objects using the extended Gaussian image, *in* "Proceedings, 7th International Joint Conference on Artificial Intelligence, 1981," pp. 595–600.
14. T. LOZANO-PÉRES AND M. A. WESLEY, An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. ACM* **22** (1979), 560–570.
15. D. E. MULLER AND F. P. PREPARATA, Finding the intersection of two convex polyhedra, *Theoret. Comput. Sci.* **7** (1978), 217–236.
16. M. OSHIMA AND Y. SHIRAI, A scene description method using three-dimensional information, *Pattern Recognition* **11** (1979), 9–17.
17. F. P. PREPARATA AND S. T. HONG, Convex hulls of finite sets of points in two and three dimensions, *Comm. ACM* **20** (1977), 87–93.
18. K. SUGIHARA, Range-data analysis guided by a junction dictionary, *Artificial Intelligence* **12** (1979), 41–69.
19. S. ULLMAN, The interpretation of structure from motion, *Proc. Roy. Soc. London Ser. B* **203** (1979), 405–426.
20. L. WEINBERG, A simple and efficient algorithm for determining isomorphism of planar triply connected graphs, *IEEE Trans. Circuit Theory* **CT-13** (1966), 142–148.