アルゴリズム 31－3
（1 9 9 3. 1. 2 5）

3 次元点集合の合同性を判定する並列アルゴリズム

阿久津 達也
機械技術研究所

本稿では 3 次元空間上の 2 つの点集合（要素数 n）が合同かどうかを判定する CREW PRAM 上の $O((\log n)^3)$ 時間 $O(n/\log n)$ プロセッサ並列アルゴリズムを示す。アルゴリズムのポイントは、3 次元空間上で、ある定数個以下の点集合を選ぶことにより、3 次元の問題を 2 次元の問題に変換することである。なお、このアルゴリズムは点集合以外にも頂点が座標値を持つグラフなどの図形にも適用でき、また、正規形計算や相似性判定にも拡張可能である。

A PARALLEL ALGORITHM FOR DETERMINING THE CONGRUITY OF POINT SETS IN THREE DIMENSIONS

Tatsuya AKUTSU
Mechanical Engineering Laboratory,
1-2 Namiki, Tsukuba, Ibaraki, 305 Japan.
e-mail: akutsu@mel.go.jp

This paper presents an $O((\log n)^3)$ time $O(n/\log n)$ processors parallel algorithm for determining the congruity of two point sets in three dimensions on a CREW PRAM. In the algorithm, the original problem is reduced to the two dimensional congruity problem by selecting a set of at most 12 points in three dimensions. The algorithm can be applied for graphs such that each vertex corresponds to a point and each edge corresponds to a line segment. The algorithm can be modified for computing the canonical forms and determining the similarity.

1 Introduction

In this paper, we discuss on the problem of determining whether two objects are congruent or not in three dimensional Euclidean space. Especially, we consider such objects as point sets and graphs such that each vertex corresponds to a point and each edge corresponds to a line segment. Objects such as mechanical parts in mechanical CAD systems and three dimensional structures of chemical compounds in chemical database systems are represented as such graphs. Thus, determining the congruity of two such graphs is practically important.

Several studies have been done for the congruity problem. $O(n \log n)$ time algorithms for determining the congruity of various objects in two dimensions were developed by Manacher [12], Atallah [4] and Highnam [8]. In each algorithm, the original problem is reduced to the substring matching problem. Sugihara developed an $O(n \log n)$ time algorithm for determining the congruity of two polyhedra [13] based on the planar graph isomorphism algorithm. Atkinson developed an $O(n \log n)$ time algorithms for determining the congruity of two point sets in three dimensions [5]. Although his algorithm was complicated, Alt et.al. developed another simple algorithm for the same problem [1]. In their algorithm, the original problem is reduced to the planar graph isomorphism problem. Moreover, they showed that the congruity of point sets in d-dimensions was determined in $O(n^{d-2} \log n)$ time and they claimed that their algorithm could be modified for such objects as graphs. Although these works and this paper are concerned with exact matchings, approximate matchings are practically important. Approximate matchings have been studied extensively [1, 3, 7, 10].

By the way, we consider parallel algorithms for exact matchings. It is easy to modify the algorithms in two dimensions to parallel ones with a little overhead. However, it seems difficult to parallelize the algorithms in three dimensions with a little overhead. To my knowledge, the best parallel algorithm for planar graph isomorphism is an $O((\log n)^3)$ time $O(n^{1.5}/(\log n)^2)$ processors algorithm developed by Gazit [6]. Therefore, about $O(\sqrt{n})$ overhead is required to parallelize the algorithm of Alt et.al [1]. It seems difficult to parallelize Atkinson's algorithm with a little overhead, too. Therefore, we develop a parallel algorithm based on another idea. In this paper, we show an $O((\log n)^3)$ time $O(n/\log n)$ processors parallel algorithm on a CREW PRAM for determining the congruity of point sets or graphs in three dimensions. We assume that each arithmetic operation including square root and trigonometric functions for real numbers is carried out in one step.

2 Preliminaries

Let U denote the three dimensional Euclidean space. An *object* is a (finite or infinite) set of points in U. Two types of objects are considered in this paper. One is a finite set of points. The other is an undirected graph in which each vertex is a point in U and each edge is a line segment connecting two vertices.

We define the congruence of objects in a similar way as Ref.[13]. A mapping T of U onto itself is said to be *isometric* if $|\overline{PQ}| = |\overline{T(P)T(Q)}|$ for any two points P and Q. For any object X, we define $T(X) = \{Q \mid (\exists P \in X)(Q = T(P))\}$. Let X and Y be two objects. If there exists an isometric mapping T which satisfies $Y = T(X)$, X and Y are said to be *congruent*. Moreover, such T is called a *congruent mapping* of X onto Y. It is well known that T is composed from translations, rotations and reflections.

For searching a database which consists of objects, canonical forms are useful. A data structure $C(X)$ representing an object X is called a *canonical form* of X if it satisfies the condition that $C(X) = C(Y)$ holds if and only if X and Y are congruent. Once canonical forms are computed, the congruity can be tested only by comparing them. Moreover, such technique as binary search can be used to search the database for an identical object.

3 Parallel Algorithm in Two Dimensions

In this section, a parallel algorithm for determining the congruity of two graphs (as well as two point sets) in the plane is described. It reduces the original problem to the string matching problem. The similar reductions are used in the sequential algorithms [4, 8, 12].

Let $G(V, E)$ and $G'(V', E')$ be graphs in the plane. We assume w.l.o.g. (without loss of generality) that $|V| = |V'| = n$ and $|E| = |E'| = m$. The case of point sets can be treated by considering graphs with no edges. We assume w.l.o.g. that congruent mappings do not contain reflections.

Before describing a parallel algorithm, we show a sequential algorithm. Let $V = \{P_1, \cdots, P_n\}$ and $V' = \{P'_1, \cdots, P'_n\}$. First, the centroid O of V is computed. For each P_i, θ_i denotes the angle $\angle P_1 O P_i$. The angles are defined in such a way as the range of the angles is $[0, 2\pi)$. We define $P_i < P_j$ as follows:

$$(P_i < P_j) \iff ((\theta_i < \theta_j) \vee ((\theta_i = \theta_j) \wedge (|\overline{OP_i}| < |\overline{OP_j}|))) .$$

By sorting V according to '<', P_is are renumbered so that $P_1 < P_2 < \cdots < P_n$ holds. P'_is are renumbered in the same way.

Let $A = \{\angle P_{i-1} P_i P_{i+1}\} \cup \{\angle P'_{i-1} P'_i P'_{i+1}\}$ and $L = \{|\overline{P_i P_{i+1}}|\} \cup \{|\overline{P'_i P'_{i+1}}|\}$ where indices are computed in modulo n. We sort A and L, respectively. Let sorted lists be $(\alpha_1, \alpha_2, \cdots, \alpha_h)$ and $(l_1, l_2, \cdots, l_k)$, respectively. Let $\alpha(P_i)$ denote j such that $\alpha_j = \angle P_{i-1} P_i P_{i+1}$. Let $l(P_i)$ denote j such that $l_j = |\overline{P_i P_{i+1}}|$. $\alpha(P'_i)$ and $l(P'_i)$ are defined in the same way.

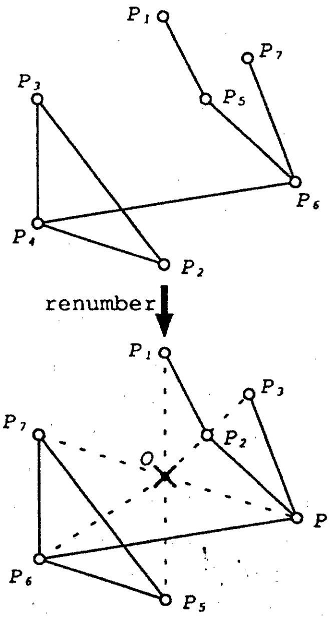

Figure 1: Construction of $str(G)$. $str(P_2)$ ="#7:1:2:2:6#" where $\alpha(P_2) = 7$ and $l(P_2) = 1$. $str(P_6)$ ="#4:3:3:1:5:6#" where $\alpha(P_6) = 4$ and $l(P_6) = 3$.

We construct a string $str(G)$ as follows (see Fig.1). Let I_j denotes $j + n - i$ modulo n. For each P_i, $\{P_{n_1}, P_{n_2}, \cdots, P_{n_j}\}$ is a set of the adjacent vertices of P_i where $I_{n_1} < I_{n_2} < \cdots < I_{n_j}$ holds. For two strings s_1 and s_2, $s_1 s_2$ denotes a concatenation of s_1 and s_2. For each P_i, we define a string $str(P_i)$ as follows:

$$str(P_i) = "\#" \ \alpha(P_i) \ ":" \ l(P_i) \ ":" \ j \ ":" \ I_{n_1} \ ":" \ I_{n_2} \ ":" \ \cdots \ ":" \ I_{n_j} \ "\#" \ .$$

$str(G)$ is defined as $str(G) = str(P_1)str(P_2) \cdots str(P_n)$. $str(G')$ is defined in the same way. Then, $G(V, E)$ and $G'(V', E')$ are geometrically congruent if and only if $str(G')$ is a substring of $str(G)str(G)$ (note that the lengths of $str(G)$ and $str(G')$ are the same). Next, we analyze the time complexity. Since it is easy to see that the other parts can be done in $O(n + m)$ time, we consider the time required for the sorting and the substring matching. The length of $str(G)$ (resp.$str(G')$) is $O(n+m)$ where we assume that the size of each index is $O(1)$. It is easy to see that the total number of elements to be sorted is $O(n + m)$. Since substring matching can be done in linear time and sorting of N elements can be done in $O(N \log N)$ time, the time complexity (the total work) is $O((n + m)\log(n + m))$.

Next, we consider a parallel implementation. Sorting and substring matching are critical for a parallel implementation. Sorting of N elements is done in $O(\log N)$ time using $O(N)$ processors on a CREW PRAM (as well as EREW PRAM) [11]. Substring matching of length N and M ($N > M$) for general alphabet is done in $O(\log M)$ time using $O(N/\log M)$ processors on a CRCW PRAM [11]. Therefore, the sorting and the substring matching can be done in $O((\log(n + m))^2)$ time using $O((n+m)/\log(n+m))$ processors on a CREW PRAM. Since it is easy to see that the other parts can be done in $O((\log(n + m))^2)$ time using $O((n + m)/\log(n + m))$ processors, the following theorem holds.

[Theorem 1] Whether two graphs $G(V, E)$ and $G'(V', E')$ in two dimensional space are geometrically congruent or not is determined in $O((\log n)^2)$ time using $O(n/\log n)$ processors on a CREW PRAM where $n = \max\{|V|, |E|, |V'|, |E'|\}$. □

The canonical form of G is computed in the following way. $a_j \cdots a_N a_1 \cdots a_{j-1}$ is said to be the canonical form of a (circular) string $a_1 \cdots a_N$ if the following condition is satisfied[9]:

$$(1 \leq \forall i \leq N)(\ a_j \cdots a_N a_1 \cdots a_{j-1} \leq a_i \cdots a_N a_1 \cdots a_{i-1}).$$

It is easy to see that the concatenation of $(\alpha_1, \cdots, \alpha_h)$, $(l_1, \cdots, l_k)$ and the canonical form of $str(G)$ becomes the canonical form of G. Since the canonical form of a circular string of length N over an alphabet of size $O(N)$ can be computed in $O(\log N)$ time using $O(N)$ processors on a CRCW PRAM [9], the canonical form of G can be computed in $O((\log n)^2)$ time using $O(n)$ processors on a CREW PRAM. Note that both determining the congruity and computing the canonical form can be done in $O(\log n)$ time using $O(n)$ processors on a CRCW PRAM.

4 Parallel Algorithm in Three Dimensions

In this section, a parallel algorithm for determining the congruity in three dimensions is described. It reduces the three dimensional problem to the two dimensional problem. Although the algorithm for point sets is described, it can be modified for such objects as graphs.

4.1 Reduction to the Two Dimensional Problem

Alt et.al. showed that the d dimensional congruity problem could be reduced to the $d - 1$ dimensional congruity problem [1]. Since their reduction is not efficient, we develop a similar but different reduction.

[Definition 1] A mapping $cp(S)$ which maps a point set $S \subset R^d$ to a point set $cp(S) \subset R^d$ is called a *CMC* (Characteristic Mapping for Congruity) if the following conditions are satisfied:

- For any isometric transformation T, $cp(T(S)) = T(cp(S))$,
- $|cp(S)|$ is bounded by a constant K,
- $cp(S)$ does not contain the centroid of S.

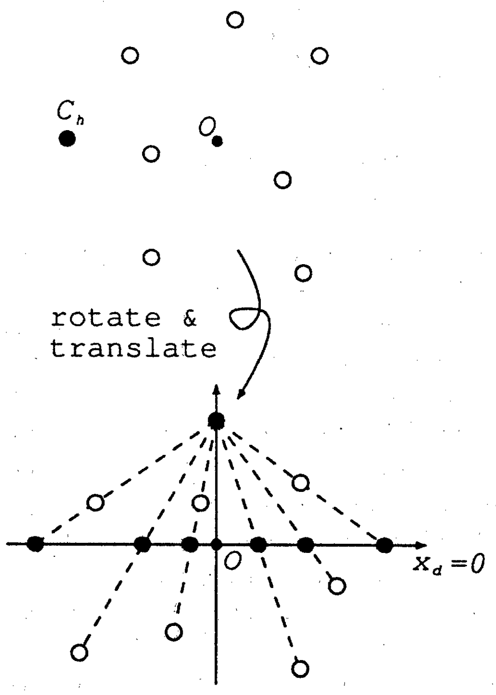

Figure 2: Projection to a Hyperplane

If there is a CMC $cp(S)$, the problem of determining the congruity of point sets in d dimensions can be reduced to the problem of determining the congruity of (labeled) point sets in $d-1$ dimensions as follows. We assume w.l.o.g. that $S = \{P_1, \cdots, P_n\}$, $S' = \{P'_1, \cdots, P'_n\}$, $cp(S) = \{C_1, \cdots, C_K\}$ and $cp(S') = \{C'_1, \cdots, C'_K\}$ are given.

We construct a point set $proj(S, C_h)$ in $d-1$ dimensions for each C_h as follows (see Fig.2). $proj(S', C'_h)$ is constructed in the same way. First, S and C_h are translated and rotated so that the origin O coincides with the centroid of S and $\overrightarrow{OC_h} = t\overrightarrow{OU}$ for some $t > 0$ where $U = (0, 0, \cdots, 0, 1)$. Hereafter, we assume that S and C_h are already transformed in this way. Next, each $P_i = (x_1, \cdots, x_d)$ in S is projected on the hyperplane $x_d = 0$, that is, each P_i is projected to $(\frac{tx_1}{t-x_d}, \frac{tx_2}{t-x_d}, \cdots, \frac{tx_{d-1}}{t-x_d}, 0)$. Then, a set of projected points $\{Q_1, \cdots, Q_m\}$ becomes $proj(S, C_h)$. Since multiple points may be projected on the same point, projected points are labeled. Let $\alpha(Q_i) = \{ t_j \mid \overrightarrow{C_h P_j} = t_j \overrightarrow{C_h Q_i}$ where P_j is projected to $Q_i \}$. Each Q_i is labeled so that $label(Q_i) = label(Q_j)$ holds if and only if $\alpha(Q_i) = \alpha(Q_j)$ and $|\overline{C_h Q_i}| = |\overline{C_h Q_j}|$ hold.

It is easy to see that, for each C_h, S and S' are congruent if and only if there is C'_i such that $proj(S, C_h)$ and $proj(S', C'_i)$ are congruent (including labels). Thus, the congruity in d dimensions can be determined by testing the congruity in $d-1$ dimensions at most K times. Note that the parallel algorithm in the plane can be modified for the case where points are labeled. It is easy to see that $proj(S, C_h)$ can be computed in $O((\log n)^2)$ time using $O(n)$ processors on a CREW PRAM.

4.2 Computing CMC

In this subsection, we show that there is a CMC in 3 dimensions which can be computed efficiently in parallel. Indeed, the procedure $CMC(S)$ computes such a CMC.

In the procedure, SP denotes the surface of the sphere of radius 1 whose center is the centroid O of the point set S. $cv(V)$ denotes the convex hull of the point set V. For each point $V_i \in V$, $deg(V_i)$ denotes the vertex degree of

V_i when the convex hull $cv(V)$ is regarded as a planar graph. $F(cv(V))$ denotes the set of faces of $cv(V)$. For each face $f \in F(cv(V))$, $fdeg(f)$ denotes the number of edges of f. For each face $f \in F(cv(V))$, $pr(f)$ denotes the projected point of the centroid of the face f (see Fig.3).

```
Procedure $CMC(S)$
begin
  Let $O$ be the centroid of $S$;
  Project each $P_i \in S$ from $O$ on $SP$;
    /* let $V = \{V_1, \cdots, V_k\}$ be a set
      of projected points */
  if $k \leq 12$ then return $V$;
  if $V$ lies on a plane then return $\{U_1, U_2\}$;
    /* $U_1, U_2$ are the points on $SP$ such that
      $\overrightarrow{OU_i}$ is perpendicular to the plane */
  Construct the convex hull $cv(V)$ of $V$;
  if $cv(V)$ is not a regular graph then
    begin
      $D_1 \leftarrow \{ V_i \mid deg(V_i)$ is the minimum $\}$;
      $D_2 \leftarrow V - D_1$;
      if $|D_1| \leq |D_2|$ then return $CMC(D_1)$
      else return $CMC(D_2)$
    end;
  $F_1 \leftarrow \{ f \mid f \in F(cv(V)) \wedge fdeg(f) = 3 \}$;
  $F_2 \leftarrow \{ f \mid f \in F(cv(V)) \wedge fdeg(f) > 3 \}$;
  if $cv(V)$ is 3-regular and $|F_1| = 0$ then
    return $CMC(\{pr(f) \mid f \in F_2\})$
  else if $|F_1| \leq |F_2|$ then
    return $CMC(\{pr(f) \mid f \in F_1\})$
  else
    return $CMC(\{pr(f) \mid f \in F_2\})$
end
```

[Proposition 1] If $cv(V)$ is d-regular, d is 3, 4 or 5.

In the followings, v denotes the number of vertices of $cv(V)$, e denotes the number of edges of $cv(V)$, f denotes the number of faces of $cv(V)$, f_1 denotes the number of faces of $cv(V)$ such that $fdeg(f) = 3$ and f_2 denotes the number of faces of $cv(V)$ such that $fdeg(f) > 3$.

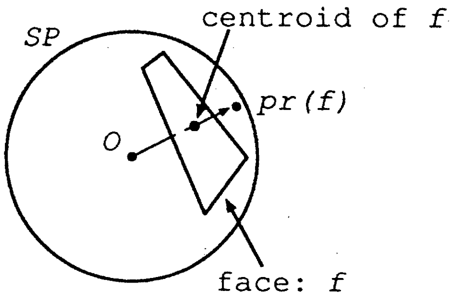

Figure 3: Definition of $pr(f)$

[Proposition 2] If $cv(V)$ is 5-regular and $v > 12$, then $f_1 \leq \frac{5}{6}v \vee f_2 \leq \frac{5}{6}v$ holds.
(Proof) From Euler formula, $v - e + f = 2$. (1)
Since each face is surrounded by at least 3 edges, $3f \leq 2e$. (2)
Since $cv(V)$ is 5-regular, $5v = 2e$. (3)
From (2) and (3), $f \leq \frac{2}{3}e \leq \frac{5}{3}v$. (4)
From (1) and (3), $2f - 3v = 4$. (5)
If $f_2 = 0$, $v = 12$ is derived from $2f_1 - 3v = 4$ and $3f_1 = 2e = 5v$. Thus, $f_2 > 0$.
From $f_1 + f_2 = \frac{3}{2}v + 2$ and $3f_1 + 4f_2 \leq 2e \leq 5v$, $f_1 \geq v + 8 > 0$.
Therefore, since $f_1 > 0 \wedge f_2 > 0$ holds, the proposition holds. □

[Proposition 3] If $cv(V)$ is 4-regular and $v > 6$, then $f_1 \leq \frac{2}{3}v \vee f_2 \leq \frac{2}{3}v$ holds.
(Proof) As in Proposition 2, $e = 2v$ (1) and $f \leq \frac{2}{3}e \leq \frac{4}{3}v$ (2) hold.
From (1) and Euler formula, $f = v + 2$. (3)
If $f_2 = 0$, $v = 6$ is derived from $f_1 = v + 2$ and $3f_1 = 2e = 4v$. Thus, $f_2 > 0$.
From $f_1 + f_2 = v + 2$ and $3f_1 + 4f_2 \leq 2e \leq 4v$, $f_1 \geq 8$.

Therefore, since $f_1 > 0 \wedge f_2 > 0$ holds, the proposition holds. □

[Proposition 4] If $cv(V)$ is 3-regular and $v > 4$, then $(f_1 = 0 \rightarrow f_2 \leq \frac{3}{4}v) \wedge (f_1 > 0 \rightarrow f_1 \leq \frac{1}{2}v \vee f_2 \leq \frac{1}{2}v)$ holds.
(Proof) From $3v = 2e$ and $3f \leq 2e$, $f \leq v$ holds.
If $f_2 = 0$, $v = 4$ is derived from $2f_1 = v + 4$ and $3f_1 = 2e = 3v$. Thus, $f_2 > 0$.
If $f_1 = 0$, $f_2 \leq \frac{3}{4}v$ holds from $4f_2 \leq 2e = 3v$.
If $f_1 > 0$, $f_1 \leq \frac{1}{2}v$ or $f_2 \leq \frac{1}{2}v$ holds from $f \leq v$.
□

[Lemma 1] The number of recursive calls of $CMC(S)$ is $O(\log|S|)$.
(Proof) Let S_i be the parameter of ith recursive call of $CMC(S)$. If $cv(V)$ for S_i is not regular, $|S_{i+1}| \leq \frac{1}{2}|S_i|$ holds. Otherwise, $|S_{i+1}| \leq \frac{5}{6}|S_i|$ holds from the above propositions. Thus, $|S_{i+1}| \leq \frac{5}{6}|S_i|$ holds for all i and the lemma follows. □

[Theorem 2] The congruity of two point sets in three dimensions is determined in $O((\log n)^3)$ time using $O(n/\log n)$ processors on a CREW PRAM.
(Proof) It is easy to see that the procedure $CMC(S)$ selects a CMC which contains at most 12 points. Therefore, making the projection and applying the algorithm in two dimensions, the congruity of point sets can be determined.

Next, we consider the time complexity and the number of processors. Since $O((\log n)^3)$ time and $O(n/\log n)$ processors are enough for the projection and determining the congruity in two dimensions, we consider the procedure $CMC(S)$. It is known that the convex hull of N points can be constructed in $O((\log N)^2)$ time using $O(N)$ processors on a CREW PRAM [2]. It is easy to see that $O(\log n)$ time and $O(n)$ processors are enough for executing the other part of one recursive step of $CMC(S)$. Since $CMC(S)$ is executed $O(\log n)$ times, $O((\log n)^3)$ time and $O(n)$ processors are enough.

The number of processors can be reduced by applying Brent's theorem [11] as follows. Let S_i be the parameter of ith recursive step of $CMC(S)$. Since $|S_{i+1}| \leq \frac{5}{6}|S_i|$ holds, the total work for $CMC(S)$ is

$$\begin{aligned}
&\sum_i O(|S_i|(\log|S_i|)^2) \\
\leq\ & (1 + \frac{5}{6} + (\frac{5}{6})^2 + \cdots)O(|S|(\log|S|)^2) \\
=\ & O(n(\log n)^2)\,.
\end{aligned}$$

Since $O((\log n)^3)$ time is required, $O(n/\log n)$ processors are enough from Brent's theorem. Indeed, if $O(n/\log n)$ processors are allocated, at most $\max(c_1(\frac{5}{6})^{(i-1)}(\log n)^3, c_2(\log n)^2)$ time is required for ith recursive step of $CMC(S)$ where c_1 and c_2 are the appropriate constants. Therefore, $O((\log n)^3)$ time is enough. □

Note that the algorithm can be modified for computing the canonical forms if the modified algorithm in two dimensions is applied.

5 Concluding Remarks

In this paper, an $O((\log n)^3)$ time $O(n/\log n)$ processors parallel algorithm on a CREW PRAM for determining the congruity of two point sets or two graphs in three dimensions is shown.

Although we assumed a real PRAM as a computational model in which each arithmetic operation including square root and trigonometric functions for real numbers is carried out in one step, it seems possible that square root and trigonometric functions are removed and the algorithm is modified for rational numbers.

The algorithm can be modified for determining whether two objects are similar in the sense that they differ only in their sizes. This problem is discussed in Ref.[13] and a similar method can be applied for our cases. For that purpose, it is enough that the size of each input object is normalized by expanding or shrinking it according to the length between the centroid and the farthest vertex from it.

While exact matchings are treated in this paper, approximate matchings are more important for practical applications. Although several algorithms have been developed for approximate matchings [1, 3, 7, 10], they do not seem to be practical. Therefore, practical algorithms for approximate matchings should be developed.

References

[1] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. "Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, Vol. 3, pp. 237–256, 1988.

[2] N. M. Amato and F. P. Preparata. "The parallel 3D convex hull problem revisited". *International Journal of Computational Geometry and Applications*, Vol. 2, pp. 163–173, 1992.

[3] E. M. Arkin, K. Kedem, J. S. B. Mitchell, J. Sprinzak, and M. Werman. "Matching points into noise regions: combinatorial bounds and algorithms". In *Proceeding of ACM-SIAM Symposium on Discrete Algorithms*, pp. 42–51, 1991.

[4] M. J. Atallah. "On symmetry detection". *IEEE Transactions on Computers*, Vol. C-34, pp. 663–666, 1985.

[5] M. D. Atkinson. "An optimal algorithm for geometrical congruence". *Journal of Algorithms*, Vol. 8, pp. 159–172, 1987.

[6] H. Gazit. "A deterministic parallel algorithm for planar graphs isomorphism". In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pp. 723–732, 1991.

[7] P. J. Heffernan and S. Schirra. "Approximate decision algorithm for point sets congruence". In *Proceedings of ACM Symposium on Computational Geometry*, pp. 93–101, 1992.

[8] P. T. Highnam. "Optimal algorithms for finding the symmetries of a planar point set". *Information Processing Letters*, Vol. 22, pp. 219–222, 1986.

[9] C. S. Iliopoulos and W. F. Smyth. "Optimal algorithms for computing the canonical form of a circular string". *Theoretical Computer Science*, Vol. 92, pp. 87–105, 1992.

[10] K. Imai, S. Sumino, and H. Imai. "Minimax geometric fitting of two corresponding sets of points". In *Proceedings of ACM Symposium on Computational Geometry*, pp. 266–275, 1989.

[11] J. JáJá. *"An Introduction to Parallel Algorithms"*. Addison-Wesley, Massachusetts, 1992.

[12] G. Manacher. "An application of pattern matching to a problem in geometrical complexity". *Information Processing Letters*, Vol. 5, pp. 6–7, 1976.

[13] K. Sugihara. "An $n \log n$ algorithm for determining the congruity of polyhedra". *Journal of Computer and System Sciences*, Vol. 29, pp. 36–47, 1984.