Available online at www.sciencedirect.com

ScienceDirect

Information Processing Letters 106 (2008) 52–58

Information Processing Letters

www.elsevier.com/locate/ipl

# A new algorithm for computing the minimum Hausdorff distance between two point sets on a line under translation

Banghe Li [a], Yuefeng Shen [a,b,*], Bo Li [a,b]

[a] *Center of Bioinformatics & Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China*

[b] *Graduate University of Chinese Academy of Sciences, Beijing 100049, China*

Received 22 March 2007; received in revised form 7 September 2007; accepted 3 October 2007

Available online 10 October 2007

Communicated by F.Y.L. Chin

**Abstract**

To determine the similarity of two point sets is one of the major goals of pattern recognition and computer graphics. One widely studied similarity measure for point sets is the Hausdorff distance. So far, various computational methods have been proposed for computing the minimum Hausdorff distance. In this paper, we propose a new algorithm to compute the minimum Hausdorff distance between two point sets on a line under translation, which outperforms other existing algorithms in terms of efficiency despite its complexity of $O((m+n)\lg(m+n))$, where $m$ and $n$ are the sizes of two point sets.
© 2007 Elsevier B.V. All rights reserved.

*Keywords:* Hausdorff distance; Pattern recognition; Computational geometry

## 1. Introduction

One of the major goals of pattern recognition and computer graphics is to measure the similarity of two point sets, and Hausdorff distance has been widely used as a similarity measure for its tolerance of small position error [3] and irrelevancy of sequence. In applications of computer graphics and pattern recognition [3,6], we want to translate an image taken by a camera against a template such that the Hausdorff distance between them is minimum. This problem has been focused for a long time and various algorithms have been proposed [1,2,7, 4,5]. In addition, Hausdorff distance has also been applied to molecular biology [8,9].

In this paper, we consider one-dimensional version of this problem, in which the point sets are on a line. Huttenlocher and Kedem [2] solved this problem in $O(mn\lg(mn))$ time and Rote [1] solved it in $O((m+n)\lg(m+n))$ time and proved that the complexity of $O((m+n)\lg(m+n))$ is optimal for this problem based on the algebraic decision tree model.

We propose a new algorithm for this problem, which outperforms Rote's algorithm in terms of efficiency despite its complexity of $O((m+n)\lg(m+n))$. We prove that our algorithm is as efficient as Rote's in the worst case and usually much more efficient. Numerical experiments show that our algorithm runs about 15 times faster than Rote's algorithm.

[*] Corresponding author at: Center of Bioinformatics & Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China.
*E-mail addresses:* libh@amss.ac.cn (B. Li), shyf@amss.ac.cn (Y. Shen), libo205@mails.gucas.ac.cn (B. Li).

0020-0190/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2007.10.003

To make the problem clear, we first present some preliminary knowledge about Hausdorff distance and formulate the problem in a mathematical form.

The Hausdorff distance between two given point sets $A = \{a_1, a_2, \dots, a_m\}$ and $B = \{b_1, b_2, \dots, b_n\}$ in $k$-dimensional space ($\mathbb{R}^k$) is defined as

$$H(A, B) = \max\Big\{\max_{a\in A}\min_{b\in B} d(a, b), \max_{b\in B}\min_{a\in A} d(b, a)\Big\}.$$

We define $d(a, B) = \min_{b\in B} d(a, b)$ and $d(b, A) = \min_{a\in A} d(b, a)$, then $H(A, B)$ can be written as

$$H(A, B) = \max\Big\{\max_{a\in A} d(a, B), \max_{b\in B} d(b, A)\Big\}.$$

Now we consider the Hausdorff distance between $A$ and $B$ under translation. Without loss of generality, we fix $B$ and translate $A$ by $t \in \mathbb{R}^k$ , then $d(a + t, B) = \min_{b\in B} d(a + t, b)$ and $d(b, A + t) = \min_{a\in A} d(b, a + t)$. Thus

$$H(A + t, B) = \max\Big\{\max_{a\in A} d(a + t, B), \max_{b\in B} d(b, A + t)\Big\}.$$

Therefore, we can formulate the problem of computing the minimum Hausdorff distance between two point sets in $\mathbb{R}^k$ under translation by

$$H_T(A, B) = \min_{t\in\mathbb{R}^k} H(A + t, B).$$

In this paper all the points in $A$ and $B$ are in $\mathbb{R}^1$, that is $a_i \in \mathbb{R}^1$ and $b_j \in \mathbb{R}^1$ for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. So, $d(a_i, b_j)$ indeed is the absolute difference between $a_i$ and $b_j$, which can be denoted as $|a_i - b_j|$. Therefore,

$$d(a_i, B) = \min_{j\in J} |a_i - b_j|, \quad J = \{1, 2, \dots, n\},$$

$$d(b_j, A) = \min_{i\in I} |b_j - a_i|, \quad I = \{1, 2, \dots, m\},$$

and

$$H_T(A, B) = \min_{t\in\mathbb{R}^1} H(A + t, B).$$

This paper is organized as follows. We review Rote's algorithm in Section 2 and propose our algorithm in Section 3. Some comparisons between Rote's algorithm and ours are addressed in Section 4. Section 5 is the conclusion.

## 2. Rote's algorithm

For convenience, we order the elements in $A$ and $B$ as $a_1 < a_2 < \cdots < a_m$ and $b_1 < b_2 < \cdots < b_n$, respectively. First, transform the primal problem to a variation dealing with the directed Hausdorff distance:

$$DH(\mathbb{A}, \mathbb{B}) = \max_{\alpha\in\mathbb{A}}\min_{\beta\in\mathbb{B}} |\beta - \alpha| = \max_{\alpha\in\mathbb{A}} d(\alpha, \mathbb{B}),$$

where $\mathbb{B}$ is constructed by taking a reflected copy $\tilde{A}$ of $A$ and adding it to the right of $B$ sufficiently far apart, and $\mathbb{A}$ by taking a reflected copy $\tilde{B}$ of $B$ and adding it to the right of $A$, the same distance apart. It is clear that the directed Hausdorff distance between the two new sets is the same as the Hausdorff distance between the two original sets, and this equality remains true if we translate any of the sets along the line [1].

Second, construct the function

$$F(t) = DH(\mathbb{A} + t, \mathbb{B}) = \max_{\alpha\in\mathbb{A}} d(\alpha + t, \mathbb{B}).$$

Without loss of generality, we assume that $\mathbb{A}$ and $\mathbb{B}$ are aligned at their right and left endpoints, respectively. For a single point $x \in \mathbb{A}$,

$$f_{x,\mathbb{B}}(t) = d(x + t, \mathbb{B}) \tag{1}$$

is a continuous piecewise linear function of the variable $t$, with edges of slope $\pm$ 1. When $t$ is equal to any element of $\mathbb{B} - x$, $f_{x,\mathbb{B}}(t)$ arrives its minimum 0. The function $F(t)$ is just the upper envelope of them,

$$F(t) = \max_{\alpha\in\mathbb{A}} f_{\alpha,\mathbb{B}}(t). \tag{2}$$

It can be easily proved that

$$F(t) \geqslant g(t) = |t| \tag{3}$$

and each of the $m + n$ functions $f_{\alpha,\mathbb{B}}(t)$, for $\alpha \in \mathbb{A}$ contributes at most two edges to the function $F(t)$ [1]. These two edges form a spike (where a spike consists of a rising edge, a local maximum, and a falling edge). Thus, $F(t)$ has at most $m + n$ spikes.

To find the spike of $f_{\alpha,\mathbb{B}}(t)$ which possibly contributes to $F(t)$, we compute the value $f_{\alpha,\mathbb{B}}(t = 0)$. If $f_{\alpha,\mathbb{B}}(t = 0) = 0$, then function $f_{\alpha,\mathbb{B}}(t)$ contributes no spike to $F(t)$. If $f_{\alpha,\mathbb{B}}(t = 0) > 0$, then function $f_{\alpha,\mathbb{B}}(t)$ possibly contributes at most one spike to $F(t)$. In the latter case we follow the edge in the rising direction to the next peak, which is the local maximum of $f_{\alpha,\mathbb{B}}(t)$, from which a spike can be found. Thus, those spikes possibly contribute to $F(t)$ can be found in $O(m + n)$ time. Now, $F(t)$ is the maximum of all these spikes and the low bound $g(t)$ (see Eq. (3)). We could sort the spikes according to their left endpoints and add them from left to right, which can be done in $O((m + n)\lg(m + n))$ time. After the sorting procedure, $F(t)$ can be computed in $O(m + n)$ time.

Thus,

$$H_T(A, B) = \min_{t\in\mathbb{R}^1} H(A + t, B) = \min_{t\in\mathbb{R}^1} DH(\mathbb{A} + t, \mathbb{B}) = \min_{t\in\mathbb{R}^1} F(t)$$

can be computed in $O((m + n)\lg(m + n))$ time.

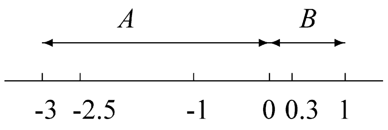

Fig. 1. We translate the sets $A$ and $B$ on a line to align the right endpoint of $A$ and the left endpoint of $B$ on the origin. For simplicity, we choose $A = \{0, 0.5, 2, 3\}$ and $B = \{0, 0.3, 1\}$ as example Figs. 1–3 in this paper.

## 3. Our algorithm

Similarly to Rote's algorithm, we order the elements in $A$ and $B$ as $a_1 < a_2 < \cdots < a_m$ and $b_1 < b_2 < \cdots < b_n$, respectively. Since the initial positions of $A$ and $B$ are irrelevant to the distance we want to calculate, we translate the sets $A$ and $B$ such that $a_m = b_1 = 0$ (see Fig. 1). Then

$$H(A+t, B) = \max\left\{\max_{i\in I} d(a_i + t, B), \max_{j\in J} d(b_j, A+t)\right\}.$$

Indeed, $H(A+t, B)$ is the upper envelope of all the functions $d(a_i + t, B)$ and $d(b_j, A+t)$, $i \in I$ and $j \in J$.

Let's consider function $d(a_i + t, B)$ first. We define

$$f_{a_i,B}(t) = d(a_i + t, B),$$

then

$$f_{a_i,B}(t) = d(t, -a_i + B) = \min_{j\in J} d(t, -a_i + b_j). \tag{4}$$

The function $f_{a_i,B}(t)$ is continuous and piecewise linear, with edges of slope $\pm 1$. Furthermore, $f_{a_i,B}(t) = 0$ if and only if $t \in B - a_i$. In other words, function $f_{a_i,B}(t)$ is determined by $B - a_i$ (see Fig. 2). By the discussion above, we have the following lemma:

**Lemma 1.** *All the functions $f_{a_i,B}(t)$ for $i \in I$ are same under translation (see Fig. 2).*

Similarly,

$$f_{b_j,A}(t) = d(b_j, A+t) = d(t, b_j - A) = \min_{i\in I} d(t, b_j - a_i).$$

So the function $f_{b_j,A}(t)$ is continuous and piecewise linear, with edges of slope $\pm 1$. Furthermore, $f_{b_j,A}(t) = 0$ if and only if $t \in -A + b_j$. And symmetrically to Lemma 1, we get

**Lemma 2.** *All the functions $f_{b_j,A}(t)$ for $j \in J$ are same under translation (see Fig. 2).*

**Theorem 3.** *If $|a_1 - a_m| \geqslant |b_1 - b_n|$, then*

$$H(A+t, B) = \begin{cases} f_{a_1,B}(t), & \text{if } t < 0; \\ f_{a_m,B}(t), & \text{if } t > -a_1 + b_n. \end{cases}$$

*If $|a_1 - a_m| < |b_1 - b_n|$, then*

$$H(A+t, B) = \begin{cases} f_{b_n,A}(t), & \text{if } t < 0; \\ f_{b_1,A}(t), & \text{if } t > -a_1 + b_n. \end{cases}$$

**Proof.** When $t < 0$, we have

$$\begin{aligned} f_{a_i,B}(t) &= \min_{j\in J} d(t, -a_i + b_j) \\ &= \min_{j\in J}\big((-a_i + b_j) - t\big) \\ &= (-a_i + b_1 - t) = -a_i - t. \end{aligned}$$

For $a_1 < a_2 < \cdots < a_m = 0$, we have $f_{a_1,B}(t) > f_{a_2,B}(t) > \cdots > f_{a_m,B}(t)$.

Similarly,

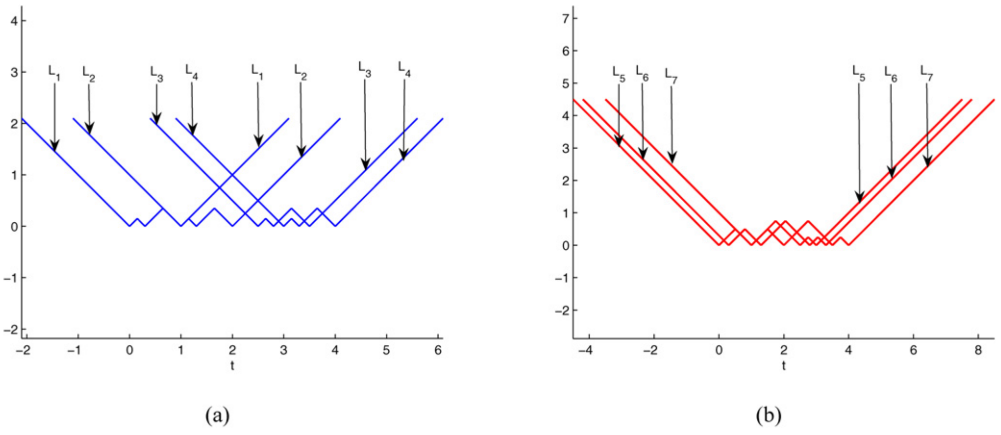

Fig. 2. (a) $L_1, \ldots, L_4$ denote the functions $f_{a_4,B}(t), \ldots, f_{a_1,B}(t)$, respectively. They are basically the same under translations. (b) $L_5, L_6, L_7$ denote the functions $f_{b_1,A}(t)$, $f_{b_2,A}(t)$, $f_{b_3,A}(t)$, respectively. They are also basically the same under translations.

$$f_{b_j,A}(t) = \min_{i\in I} d(t, b_j - a_i) = \min_{i\in I}\big((b_j - a_i) - t\big)$$
$$= b_j - a_m - t = b_j - t.$$

For $0 = b_1 < b_2 < \cdots < b_n$, we have $f_{b_1,A}(t) < f_{b_2,A}(t) < \cdots < f_{b_n,A}(t)$.

So we know that

$$H(A + t, B) = \max\{f_{b_n,A}(t), f_{a_1,B}(t)\},$$

if $t \leqslant 0$.

Furthermore, $-a_1 \geqslant b_n$ holds if and only if $f_{a_1,B}(t) \geqslant f_{b_n,A}(t)$ holds when $t < 0$.

When $t > (b_n - a_1)$, we have

$$f_{a_i,B}(t) = \min_{j\in J} d(t, -a_i + b_j) = \min_{j\in J}\big((t - b_j) + a_i\big)$$
$$= (t - b_n) + a_i > 0,$$

for any $i \in I$. For $a_1 < a_2 < \cdots < a_m = 0$, we have $f_{a_1,B}(t) < f_{a_2,B}(t) < \cdots < f_{a_m,B}(t)$.

Similarly,

$$f_{b_j,A}(t) = \min_{i\in I} d(t, b_j - a_i) = \min_{i\in I}(t - b_j + a_i)$$
$$= t - b_j + a_1 > 0,$$

for any $j \in J$. For $0 = b_1 < b_2 < \cdots < b_n$, we have

$$f_{b_1,A}(t) > f_{b_2,A}(t) > \cdots > f_{b_n,A}(t).$$

So we know that

$$H(A + t, B) = \max\{f_{b_1,A}(t), f_{a_m,B}(t)\},$$

if $t > (b_n - a_1)$.

Furthermore, $-a_1 \geqslant b_n$ holds if and only if $f_{a_m,B}(t) \geqslant f_{b_1,A}(t)$ holds when $t > (b_n - a_1)$.

Thus, we conclude that if $|a_1 - a_m| \geqslant |b_1 - b_n|$, that is $-a_1 \geqslant b_n$, then $H(A + t, B) = f_{a_m,B}(t)$ when $t$ is sufficiently large, and $H(A+t, B) = f_{a_1,B}(t)$ when $t$ is sufficiently small. If $|a_1 - a_m| < |b_1 - b_n|$, that is $-a_1 < b_n$, then $H(A + t, B) = f_{b_1,A}(t)$ when $t$ is sufficiently large, and $H(A+t, B) = f_{b_n,A}(t)$ when $t$ is sufficiently small. □

Without loss of generality, we assume below that $-a_1 \geqslant b_n$. In this case, let $f(t)$ be

$$f(t) = \begin{cases} f_{a_1,B}(t), & \text{if } t \leqslant \frac{b_n - a_1}{2}; \\ f_{a_m,B}(t), & \text{if } t > \frac{b_n - a_1}{2}. \end{cases} \tag{5}$$

Indeed,

$$f(t) = \max\{f_{a_1,B}(t), f_{a_m,B}(t), f_{b_1,A}(t), f_{b_n,A}(t)\},$$

which is the upper envelope of these four functions (see Fig. 3).

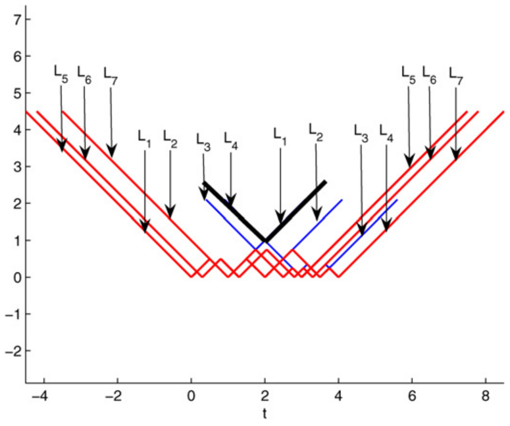

Fig. 3. This is the union of $L_1, \ldots, L_7$. The bold line is the upper envelope of the functions $\{f_{a_i,B}(t), f_{b_j,A}(t)\}$ for $i \in I = \{1, 2, 3, 4\}$ and $j \in J = \{1, 2, 3\}$, respectively.

According to the definitions of $f_{a_1,B}(t)$ and $f_{a_m,B}(t)$, we can write $f(t)$ as

$$f(t) = \begin{cases} -t - a_1, & \text{if } t \leqslant \frac{b_n - a_1}{2}; \\ t - b_n, & \text{if } t > \dfrac{b_n - a_1}{2}. \end{cases} \tag{6}$$

By Theorem 3 we know that $H(A + t, B) \geqslant f(t)$, and the equality holds when $t$ is sufficiently large or sufficiently small. Considering the shapes of the functions $f_{a_i,B}(t)$ and $f_{b_j,A}(t)$, each of them contributes at most one spike to function $H(A + t, B)$. Therefore, $H(A + t, B)$ is the upper envelope of all these spikes and the function $f(t)$. Actually, Lemmas 4 and 5 below show the conditions on which the spikes possibly contribute to $H(A + t, B)$.

**Lemma 4.** *For $i \in I$, the strict inequality $f_{a_i,B}(t) > f(t)$ holds for some $t$ if and only if there exists only one index $j \in J$ satisfying that $(-a_i + b_j) > -a_1$ and $(-a_i + b_{j-1}) < b_n$.*

**Proof.** By the definition of $f_{a_i,B}(t)$ (see Eq. (4)), we have

$$f_{a_i,B}(-a_i + b_j) = 0,$$
$$f_{a_i,B}(-a_i + b_{j-1}) = 0.$$

When $(-a_i + b_{j-1}) \leqslant t \leqslant (-a_i + b_j)$, $f_{a_i,B}(t)$ forms a spike. This spike crosses the function $f(t)$ (see Eq. (6)) if and only if $(-a_i + b_j) > -a_1$ and $(-a_i + b_{j-1}) < b_n$. □

**Lemma 5.** *For $j \in J$, the strict inequality $f_{b_j,A}(t) > f(t)$ holds for some $t$ if and only if there exists only one index $i \in I$ satisfying that $(b_j - a_i) > -a_1$ and $(b_j - a_{i+1}) < b_n$.*

The proof of Lemma 5 is similar to that of Lemma 4.

**Theorem 6.** *If there are $i$ and $j$ such that $-a_i + b_j < b_n$ and $-a_i + b_{j+1} > -a_1$, then $-a_1 + a_i \neq b_k$, for $k = 1, 2, \dots, n$. If there are $i$ and $j$ such that $b_j - a_i < b_n$ and $b_j - a_{i-1} > -a_1$, then $b_j \neq -a_1 + a_l$, for $l = 1, 2, \dots, m$.*

**Proof.** We prove the first assertion by contradiction. Suppose there is $k$ such that $-a_1 + a_i = b_k$ holds, then we have $-a_i = -a_1 - b_k$. Thus, $-a_i + b_j = -a_1 - b_k + b_j$ and $-a_i + b_{j+1} = -a_1 - b_k + b_{j+1}$. By the conditions of $-a_i + b_j < b_n$ and $-a_i + b_{j+1} > -a_1$, we have $-a_1 - b_k + b_j < b_n$ and $-a_1 - b_k + b_{j+1} > -a_1$, which means that $-b_k + b_j < 0$ and $-b_k + b_{j+1} > 0$. But this contradicts with the structure of $B$. Thus, the first assertion has been proved.

The second assertion can be proved similarly, and we do not repeat it. □

Actually, Theorem 6 says that if there exists a spike of function $f_{a_i,B}(t)$ or function $f_{b_j,A}(t)$ crossing $f(t)$ (see Eq. (6)), there must be a spike crossing $g(t) = |t|$ (see Eq. (3)) in Rote's algorithm.

To validate this assertion, we construct $\mathbb{A}$ and $\mathbb{B}$ as follows:

$$\begin{aligned}
\mathbb{A} = \{&a_1 - a_1, \dots, a_i - a_1, \dots, a_m - a_1,\\
&a_m - a_1 + l + (b_n - b_n), \dots, a_m - a_1 + l\\
&+ (b_n - b_{n-j+1}), \dots, a_m - a_1 + l + (b_n - b_1)\},\\
\mathbb{B} = \{&b_1, \dots, b_j, \dots, b_n,\\
&b_n + l + (a_m - a_m), \dots, b_n + l\\
&+ (a_m - a_{m-i+1}), \dots, b_n + l + (a_m - a_1)\},
\end{aligned}$$

where $l$ is a large number, for example $l = 3((a_m - a_1) + (b_n - b_1))$. For $a_m = b_1 = 0$, $\mathbb{A}$ and $\mathbb{B}$ are aligned at their right and left endpoints, respectively.

By Lemma 4, if there are $i$ and $j$ such that $-a_i + b_j < b_n$ and $-a_i + b_{j+1} > -a_1$, there exists a spike of $f_{a_i,B}(t)$ crossing $f(t)$ (see Eq. (6)). And by Theorem 6, we have $-a_1 + a_i \neq b_k$, for $k = 1, 2, \dots, n$. It means that $f_{(-a_1+a_i),\mathbb{B}}(0) > 0$ (see Eq. (1)), where $(-a_1 + a_i) \in \mathbb{A}$. So, there is a spike crossing $g(t) = |t|$ (see Eq. (3)) in Rote's algorithm.

Similarly, by Lemma 5, if there are $i$ and $j$ such that $b_j - a_i < b_n$ and $b_j - a_{i-1} > -a_1$, there exists a spike of $f_{b_j,A}(t)$ crossing $f(t)$. And by Theorem 6, we have $b_j \neq -a_1 + a_l$, for $l = 1, 2, \dots, m$. It means that $f_{(a_m - a_1 + l + (b_n - b_j)),\mathbb{B}}(0) > 0$, where $(a_m - a_1 + l + (b_n - b_j)) \in \mathbb{A}$. So, there is a spike crossing $g(t) = |t|$ in Rote's algorithm.

Thus, there cannot be in our algorithm more spikes than in Rote's algorithm.

Let $S_1$ be a set of intervals $[(-a_i + b_{j-1}), (-a_i + b_j)]$ satisfying $(-a_i + b_j) > -a_1$ and $(-a_i + b_{j-1}) < b_m$ for any $i \in I$ and $j \in J$. Let $S_2$ be a set of intervals $[(b_j - a_{i+1}), (b_j - a_i)]$ satisfying $(b_j - a_i) > -a_1$ and $(b_j - a_{i+1}) < b_m$, for any $i \in I$ and $j \in J$. Let $S$ be the union of $S_1$ and $S_2$, so all the spikes possibly contribute to the function $H(A + t, B)$ correspond to the intervals in $S$. Now we present a novel algorithm to compute $H_T(A, B)$. The following is the outline of our algorithm:

```
Initialization:
  1. Input two point sets $A$ and $B$ in $\mathbb{R}^1$.
  2. Sort $A$ as $a_1 < a_2 < \cdots < a_m$ and $B$ as $b_1 < b_2 < \cdots < b_n$.
  3. If $a_m - a_1 < b_n - b_1$, denote $A$ as $B$ and $B$ as $A$.
  4. Translate $A$ and $B$ such that $a_m = b_1 = 0$.
Find the set $S$:
  1. Find the set $S_1$ according to Lemma 4.
  2. Find the set $S_2$ according to Lemma 5.
  3. Let $S$ be the union of $S_1$ and $S_2$.
Find the spikes exactly contribute to $H(A + t, B)$:
  1. Sort the intervals in $S$ into nondecreasing order
     with respect to their left endpoints.
  2. Delete the intervals which are completely contained in others from $S$.
     Thus, the remaining intervals correspond to spikes that contribute to the
     function $H(A + t, B)$.
Construct function $H(A + t, B)$:
  1. Construct $H(A + t, B)$ by computing the upper envelope of $f(t)$
     and all the spikes found in the above step.
  2. Compute the minimum of $H(A + t, B)$ with respect to $t \in \mathbb{R}^1$.
```

In this algorithm, there are two sorting steps, one is the initial sorting of the input points and the other is sorting the intervals in $S$. Except for the sorting procedures, all steps of our algorithm can be performed in $O(m + n)$ time. Thus the complexity of our algorithm is $O((m + n)\lg(m + n))$.

## 4. Algorithm comparison

In this section, we give a detailed comparison of the above two algorithms for computing the minimum Hausdorff distance between two point sets on a line under translation. To make the comparison clear, we list the steps of both algorithms in Table 1.

Table 1
The comparison between Rote's algorithm and ours

| | Rote's algorithm | Our algorithm |
|---|---|---|
| step 1 | Sort $A$ and $B$ in increasing order, respectively. | Denote $A$ as the set with larger diameter and $B$ as the other set. Sort $A$ and $B$ in increasing order, respectively. |
| step 2 | Construct $\mathbb{A}$ and $\mathbb{B}$ from $A$ and $B$. | |
| step 3 | Shift $\mathbb{A}$ and $\mathbb{B}$ such that the endpoints of them are superimposed. | Shift $A$ and $B$ such that $a_m = b_1 = 0$. |
| step 4 | Find spikes of each $f_{\alpha,\mathbb{B}}(t)$, $\alpha \in \mathbb{A}$ above the function $g(t) = \|t\|$. | Find spikes of $f_{a_i,B}(t)$ and $f_{b_j,A}(t)$ above the function $f(t)$. |
| step 5 | Sort the spikes found by step 4. | Sort the spikes found by step 4. |
| step 6 | Compute the minimum Hausdorff distance under translation. | Compute the minimum Hausdorff distance under translation. |

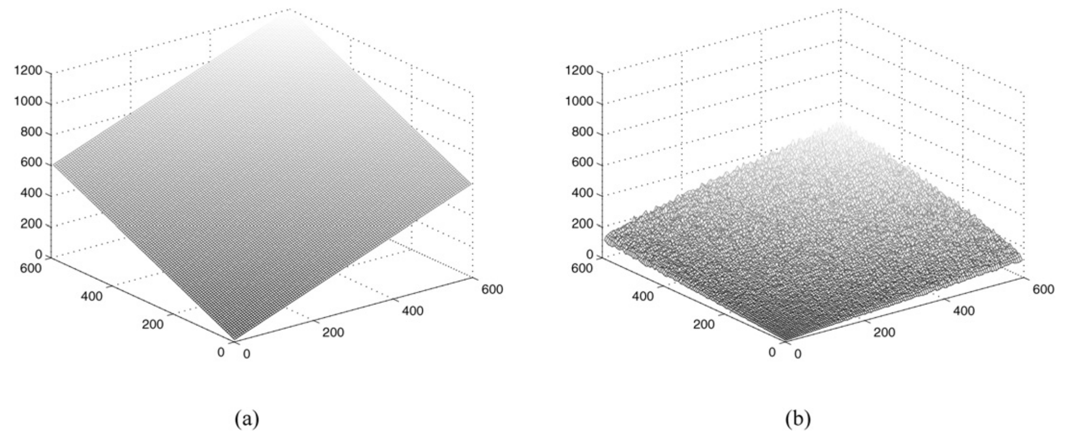

Fig. 4. (a) shows the mean value of the number of spikes found by Rote's algorithm for each pair of sizes. (b) shows the mean value of the number of intervals found by our algorithm for each pair of sizes.

Step 2 exists only in Rote's algorithm, but step 4 causes the major difference of these two algorithm. Both of the algorithms aim to find the spikes which possibly contribute to the upper envelope function $DH(\mathbb{A}+t,\mathbb{B})$ or $H(A+t,B)$. But the number of spikes found by Rote's algorithm is at least as large as the number of spikes found by our algorithm (see Theorem 6). Furthermore, in most cases Rote's algorithm find much more spikes than our algorithm.

To support this assertion, we designed an experiment. We chose the sizes of sets $A$ and $B$ from $\{5, 10, \ldots, 595, 600\}$. For each pair of these sizes, we randomly generated $A$ and $B$ in interval $[0, 1]$ for 400 times. Every time we found the spikes according to the two algorithms, respectively. For each pair of sizes, we recorded the mean value of the number of spikes found by Rote's algorithm and our algorithm (see Fig. 4). Results show that, when we chose the input sets $A$ and $B$ randomly, the average number of spikes found by Rote's method is nearly $m+n-2$, which is about 3 times the average number of spikes found by our algorithm. In an example given by Huttenlocher and Kedem [2], there are $m+n-4$ spikes found by Rote's algorithm, but 0 by ours. Relaxing the conditions of this example, no intervals can be found by our algorithm if the distance of any two adjacent points in sets $A$ and $B$ is less than the difference of the diameters of sets $A$ and $B$.

Step 5 also differs in runtime, since our algorithm sorts less spikes than Rote's (as discussed in step 4).

Thus our algorithm is more efficient, though the complexity of our algorithm is still $O((m+n)\cdot \lg(m+n))$.

In order to show the practical power of our algorithm, we computed the minimum Hausdorff distance of two input sets $A$ and $B$ with uniform distribution for 200 times. The average runtime of each algorithm with cor-

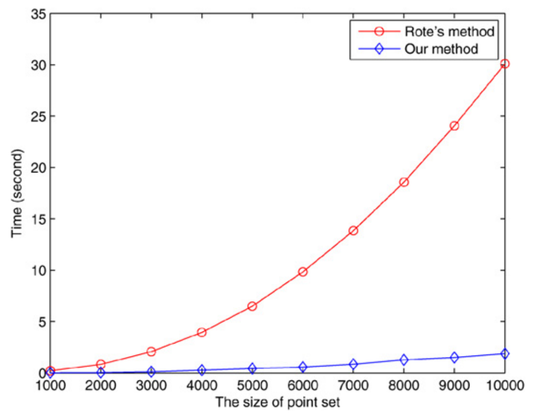

Fig. 5. We compute the minimum Hausdorff distance of two uniformly distributed input sets of same size by Rote's algorithm and ours. In each test, we record the runtime spent by both algorithms. After doing 200 tests with same size, we compute the average runtime. This diagram shows the average runtime spent by both algorithms with corresponding sizes of input sets.

Table 2
This table shows the runtime of Rote's algorithm and our algorithm

| | | 1000 | 2000 | 3000 | 4000 | 5000 |
|---|---|---|---|---|---|---|
| Rote's algorithm | 1000 | 0.2179 | 0.4630 | 0.8380 | 1.3668 | 2.0308 |
| | 2000 | 0.4676 | 0.8581 | 1.3946 | 2.0578 | 2.8702 |
| | 3000 | 0.8164 | 1.3998 | 2.0785 | 2.9052 | 3.9127 |
| | 4000 | 1.4017 | 2.0581 | 2.9073 | 3.9195 | 5.0566 |
| | 5000 | 2.0113 | 2.8898 | 3.9027 | 5.0410 | 6.4663 |
| Our algorithm | 1000 | 0.0149 | 0.0255 | 0.0433 | 0.0601 | 0.0930 |
| | 2000 | 0.0284 | 0.0480 | 0.0786 | 0.1149 | 0.1389 |
| | 3000 | 0.0468 | 0.0721 | 0.1307 | 0.1435 | 0.2552 |
| | 4000 | 0.0606 | 0.0955 | 0.1839 | 0.2468 | 0.3512 |
| | 5000 | 0.0904 | 0.1881 | 0.2573 | 0.2856 | 0.3702 |

The numbers in the first row and the second column are the sizes of the input sets. We computed the minimum Hausdorff distance of two random input sets with corresponding sizes for 200 times. Each entry in the table shows the average runtime (in seconds) spent by Rote's algorithm and ours.

responding sizes of two input sets is listed in Table 2. The average runtime spent by Rote's algorithm is about 15 times of that spent by our algorithm with the input sets of same size. In addition, we have recorded the average runtime spent by both algorithms with two point sets of same size for 200 times. When the size varies from 1000 to 10000 with interval 1000, the corresponding runtime spent by each algorithm is shown in Fig. 5. By these data, we can easily see the advantage of our algorithm.

## 5. Conclusion

Hausdorff distance has been used in many fields, and it is considered to be a useful tool to measure similarities between objects. So people pay close attention to the efficiency of algorithms to compute the minimum Hausdorff distance under translation. In this paper we give a novel and efficient algorithm to solve the problem in 1-dimensional case. Experimental results indicate that our algorithm runs about 15 times faster than Rote's on average.

In the sense of practical application, algorithms for the 2-dimensional and 3-dimensional cases are more useful. Huttenlocher, Kedem and Sharir [4] solved the 2-dimensional and 3-dimensional cases in $O(mn(m+n)\lg mn)$ and $O(mn)^2(m+n)^{1+\varepsilon}$, respectively. We hope to find more efficient algorithms for 2 and 3-dimensional cases in our future work.

## Acknowledgement

Many thanks to two anonymous referees for their comments and suggestions during the peer review process. This work was supported by 973 project (2004CB318000) of P.R. China.

## References

[1] G. Rote, Computing the minimum Hausdorff distance between two point sets on a line under translation, Inform. Process. Lett. 38 (1991) 123–127.

[2] D.P. Huttenlocher, K. Kedem, Computing the minimum Hausdorff distance for point sets under translation, in: Proc. ACM Symp. Computational Geometry, 1990, pp. 340–349.

[3] D.P. Huttenlocher, G.A. Klanderman, W.A. Rucklidge, Comparing images using the Hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (9) (1993) 850–863.

[4] D.P. Huttenlocher, K. Kedem, M. Sharir, The upper envelope of Voronoi surfaces and its applications, Discrete and Computational Geometry 9 (1993) 267–291.

[5] D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane, in: Proceedings of the Eighth Annual ACM Symposium on Computational Geometry (1992) pp. 110–119.

[6] H. Alt, L.J. Guibas, Discrete geometric shapes: Matching, interpolation, and approximation, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999, pp. 121–153.

[7] P.P. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, D. Kravets, Geometric pattern matching under Euclidean motion, in: Proc. Fifth Canadian Conference on Computational Geometry, University of Waterloo, Ontario, 1993, pp. 151–156.

[8] L.P. Chew, K. Kedem, J. Kleinberg, D. Huttenlocher, Fast detection of common geometric substructure in proteins, Journal of Computational Biology (1999).

[9] I. Halperin, B. Ma, H. Wolfson, R. Nussinov, Principles of docking: An overview of search algorithms and a guide to scoring functions, Proteins: Structure, Function, and Genetics 47 (4) (2002) 409–443.