A local start search algorithm to compute exact Hausdorff Distance for arbitrary point sets

Yilin Chen, Fazhi He, Yiqi Wu, Neng Hou

PII: S0031-3203(17)30055-9
DOI: 10.1016/j.patcog.2017.02.013
Reference: PR 6049

To appear in: *Pattern Recognition*

Received date: 11 June 2016
Revised date: 3 January 2017
Accepted date: 5 February 2017

Highlights

- This paper discovers the spatial locality feature of point sets and proposes a novel search algorithm called local start search (LSS) to compute the exact Hausdorff Distance. The LSS algorithm can greatly reduce the running time when dealing with large scale of point set, in which the spatial continuity and distance continuity are very common.
- LSS maintains high performance in both overlap and non-overlap situations of a pair of regular point sets, while EARLYBREAK experiences degraded performance in overlap situations.
- Furthermore, for general point sets in overlap situations, the preprocess of excluding the intersection in EARLYBREAK will fail. However, LSS can still process the general point sets with different point sizes after ordering the sets.
- Our algorithm outperforms the state-of-the-art algorithm. Experiments demonstrate the efficiency and accuracy of the proposed method.

A local start search algorithm to compute exact Hausdorff Distance for arbitrary point sets

Yilin Chen[a,b], Fazhi He[a,b,*], Yiqi Wu[a,b], Neng Hou[a,b]

[a]*State Key Laboratory of Software Engineering Wuhan University, Wuhan, 430072, China*
[b]*School of computer science and technology Wuhan University, Wuhan, 430072, China*

Abstract

The Hausdorff Distance (HD) is a very important similarity measurement in Pattern Recognition, Shape Matching and Artificial Intelligence. Because of its inherent computational complexity, computing the HD using the NAIVEHD (brute force) algorithm is difficult, especially for comparing the similarity between large scale point sets in the time of big data. To overcome this problem, we propose a novel, efficient and general algorithm for computing the exact HD for arbitrary point sets, which takes advantage of the spatial locality of point sets, namely, Local Start Search (LSS). Different from the state-of-the-art algorithm EARLYBREAK in PAMI 2015, our idea comes from the observation and fact that the neighbor points of a break position in the current loop have higher probability to break the next loop than other points. Therefore, in our algorithm, we add a new mechanism to record the current break position as a start position, which is initialized as search center of the next loop. Then, LSS executes the next loop by scanning the neighbor points around the center. In this way, LSS maintains high performance in both overlap and non-overlap situations. Furthermore, the LSS algorithm can process arbitrary data by adopting the Morton Curve to establish the order of scattered point sets, while the EARLYBREAK is mainly applied to regular data which require the same grid size, such as medical images or voxel data. In the non-overlapping situation when

*Corresponding author
Email address: `fzhe@whu.edu.cn` (Fazhi He)

comparing pairs of arbitrary point sets, LSS achieves performance as good as EARLYBREAK algorithm. While in the overlapping situation when comparing pairs of arbitrary point sets, LSS is faster than EARLYBREAK by three orders of magnitude. Thus, as a whole, LSS outperforms EARLYBREAK. In addition, LSS compared against the increment hausdorff distance calculation algorithm (INC) and significantly outperforms it by an order of magnitude faster. Experiments demonstrate the efficiency and accuracy of the proposed method.

Keywords: Hausdorff Distance, Pattern Recognition, Shape Matching, Similarity Measurement, Point Sets

1. Introduction

The Hausdorff Distance (HD) is a useful measurement to determine the degree of resemblance between two point sets. There are many domains that benefit from an efficient computation algorithm of the HD, including Pattern Recognition, Shape Matching, CAD/CAM and Artificial Intelligence. For example, the HD is commonly used in applications like face recognition[1, 2, 3, 4], shape matching [5, 6, 7, 8, 9, 10], CAD/CAM[11, 12, 13, 14, 15, 16], image retrieval [17, 18, 19], mesh simplification [20, 21, 22] and evaluation of segmentation techniques in medical images [23, 24, 25, 26, 27, 28]. However, computing the Hausdorff Distance is challenging because its inherent characteristics of MAX-MIN contain both maximization and minimization rather than just one or the other. Consequently, the HD computation has attracted considerable research attention in both academia and industry. Many efficient algorithms which aim to reduce the computational complexity of the HD have been proposed [29, 30, 31].

The HD is a MAX-MIN distance; given two point sets A and B in $\mathbb{R}^3$, the Hausdorff distance $H(A, B)$ is the maximum of the directed Hausdorff distances $h(A, B)$ and $h(B, A)$. The Hausdorff distance between A and B as a mathematical formula is

$$H(A, B) = max(h(A, B), h(B, A)) \tag{1}$$

The Directed Hausdorff distance $h(A, B)$ between A and B is the maximum of distances between each point $a \in A$ and its nearest neighbor $b \in B$. The two directed Hausdorff distances are as follows

$$h(A, B) = \max_{a \in A}(\min_{b \in B} dist(a, b)) \tag{2}$$

and

$$h(B, A) = \max_{b \in B}(\min_{a \in A} dist(b, a)) \tag{3}$$

where $dist(.,.)$ is the Euclidean distance in $\mathbb{R}^3$. $h(A, B)$ and $h(B, A)$ denote the directed Hausdorff distances between two point sets. Note that $h(A, B) \neq h(B, A)$ in the general case.

Many methods have been proposed in recent decades with the purpose of reducing the computational complexity of the Hausdorff distance. Recently, EARLYBREAK was proposed as an efficient algorithm for computing the exact Hausdorff Distance [32]. As far as we know, EARLYBREAK is the most notable and represents the state-of-the-art. However, we observe that there still exist some disadvantages of [32]. Firstly, when the two point sets are partially overlapping, the running time of the HD algorithm cannot be vastly reduced. Secondly, in the worst case, the two point sets completely overlap, in other words, the value of HD is equal to zero; then, the algorithm has the complexity of $O(n * m)$, where m and n are the vertex counts. Finally, although the algorithm introduces an excluding intersection strategy to deal with the overlap situation, it only works for regular data with the same grid size and does not work for general 3D point sets. Different from the EARLYBREAK algorithm, we present a novel, efficient and general algorithm based on spatial and distance continuity to improve the probability of breaking for arbitrary point sets.

The main contributions and advantages of this paper include:

- As best as we know, this paper discover and utilize the spatial locality feature of point sets and propose a novel search algorithm called local start search (LSS) to compute the exact Hausdorff Distance. The LSS algorithm can greatly reduce the running time when dealing with large scale of point set, in which the spatial continuity and distance continuity are very common. It has a nearly linear running time and an efficient performance for large-scale point sets.

- LSS maintains high performance in both overlap and non-overlap situations of a pair of regular point sets, while EARLYBREAK experiences degraded performance in overlap situations. Although EARLYBREAK can deal with the overlap situation via a preprocess of excluding the intersection for regular data of the same grid size, it requires extra cost.
- Furthermore, for general point sets in overlap situations, the preprocess of excluding the intersection in EARLYBREAK will fail. However, LSS can still process the general point sets with different point sizes after ordering the sets using the Morton Curve.

2. Related Work

The Hausdorff Distance computation has been intensely studied in various fields and utilized in a variety of applications, particularly in Pattern Recognition, Shape Matching and Artificial Intelligence. Many methods have been proposed in recent decades with the purpose of reducing the computational complexity of the Hausdorff distance. In this section, we briefly review the papers that are most relevant to ours. For a more general overview of this field, we refer readers to the survey of this field [33, 34].

Generally speaking, based on the types of data being processed, Hausdorff Distance algorithms can be classified as polygonal models, mesh surfaces and point sets. Hausdorff Distance algorithms can also be roughly divided into approximation approaches and exact approaches. Approximate algorithms aim to efficiently find an approximation of the Hausdorff distance. These algorithms have been widely used in time critical applications, such as finding the penetration depth (PD) for physically-based animation [35, 36, 37]. Exact algorithms aim to efficiently compute the exact Hausdorff distance for a specific category of point sets [32, 38] or special types of objects, such as polygons [39], line segments or special curves [40, 41]. According to the types of data being processed, we review the Hausdorff Distance algorithms as follows.

Polygonal Models. Atallah M J et al.[42] proposed a linear time algorithm for computing the Hausdorff distance for convex polygons. Given two convex polygons with m and n vertices, respectively, the algorithm presented by [42] has a computational complexity of $O(m+n)$. The algorithm is simple and obtains an efficient computational complexity while it is constrained to convex polygons. Recently, since the high computational complexity of the exact Hausdorff distance calculation, several approximate methods have been brought forward. Guthe M et al. [20] adopted a polygon subdivision strategy to approximate the Hausdorff distance within a specific error bound. However, these methods still cannot satisfy the performance demand of interactive applications. Therefore, Tang M et al. [35] proposed a novel method for approximating the Hausdorff distance calculation, which can reach interactive rates between complicated, polygonal models. The main idea of the method is to use Voronoi subdivision combined with cross-culling between two models based on bounding volume hierarchy. The cross-culling phase discards non-contribution polygon pairs for the Hausdorff distance. This method can be applied in the penetration depth computation even when it runs at highly interactive rates for a complicated dynamic scene.

Mesh Surfaces. Guthe M et al.[20] developed an efficient algorithm for computing the Hausdorff distance between a simplified mesh and the original mesh. To avoid sampling all the points between meshes that are compared, this algorithm only samples in the regions where the maximum distance between the meshes is expected. Refinement of the sampling process in the high distance regions is achieved by using a bi-hierarchical search strategy to quickly find regions of possibly high geometric distance. This algorithm can achieve efficient performance, while it is built on the specific characteristics of meshes; therefore, it lacks universality.

Krishnamurthy A et al.[43] proposed a parallel GPU-accelerated method for the Hausdorff distance calculation between two NURBS surfaces. Their method takes advantage of the accelerated structure and the hardware computation capability of a Graphics Processing Unit (GPU) to accelerate the Hausdorff distance calculation. Broadly speaking, the method consists of three steps. In the first step, the GPU is used to construct bounding-box hierarchies by enclosing sub-patches of the NURBS surfaces. In the second step, the bounding-box hierarchies are traversed by the GPU and, simultaneously, a virtual 2D array of min-max distance ranges for pairs of bounding-boxes from the two NURBS is built. The traversal process is started from the coarsest level to the finest level of the hierarchies. At each level, two culling tests are executed to cull the pairs of bounding-boxes that could not involve sub-patches probably contributing to the Hausdorff distance. Eventually, the remainder of patches between surfaces are computed at the finest level of the hierarchies to obtain the Hausdorff distance. Although it can achieve efficient performance, the method has its limitation, in which the error bounds will be large when the NURBS surface has a very high curvature.

Point Sets. Given two point sets A and B with m and n points respectively, a brute force algorithm for computing the Hausdorff distance requires $O(m * n)$ time. Nutanong S et al. [29] proposed an efficient method for the Hausdorff distance calculation between two trajectories (represented as point sets). To avoid the iteration of all points in A, the method using two R-Trees for each point set and determines the aggregate nearest neighbor simultaneously in both directions. However, the complex structures adopted by the method require extra computational effort, and R-Tree is not suitable for dense point sets.

Hence, Taha A A et al. [32] presented a fast and creative algorithm for the exact Hausdorff distance calculation between point sets, representing the state-of-the-art. The Hausdorff distance can be considered as a MAX-MIN problem, with the inner loop for minimization and the outer loop for maximization. They observed that the inner loop can break as soon as a distance is found that is less than the temporary HD ($cmax$) because the HD searches for the maximum of the minimums. On the other hand, the algorithm can break the inner loop, dose not completely scan the inner loop, and continues with the next point of the outer loop. To obtain further improvement in performance, a random sampling strategy is raised to avoid similar distances in successive iterations. However, there are three shortcomings of this algorithm. Firstly, when the two point sets are partially overlapping, the computational complexity of the HD algorithm cannot be vastly reduced. Secondly, in the worst case, the two point sets completely overlap, in other words, the value of HD is equal to zero; then, the algorithm has the complexity of $O(n * m)$, where m and n are the vertex counts. Finally, although the algorithm introduces an excluding intersection strategy to deal with the overlap situation, it only works for regular data with the same grid size and does not work for 3D point clouds. If the algorithm is used to deal with 3D point sets, the conversion of 3D point sets into voxel data is required; thus, the approximate data of the original data is obtained and an approximate solution of the HD is obtained.

Different from EARLYBREAK [32], we first find and take advantage of the spatial locality feature of point sets and present a novel, efficient and general algorithm, i.e.,Local Start Search. Furthermore, LSS can deal with arbitrary point sets; therefore, the proposed algorithm outperforms the state-of-the-art algorithm as a whole.

3. Brute Force And Early Breaking

Prior to the introduction of our method, we first describe the brute force method and Early Breaking for the Hausdorff distance calculation to ease interpretation in this section. We present them step by step.

Brute Force Method. The NAIVEHD (brute force method) algorithm is described in Algorithm 1. Apparently, there are two loops (outer loop and inner loop) and a complete scan through all points. Therefore, the computational complexity of Algorithm 1 is $O(m * n)$, where m and n are the point counts of A and B, respectively.

Reducing the computational complexity is the research focus.

Algorithm 1 NAIVEHD, i.e., brute force method, for computing the directed Hausdorff distance

```
Require: Two finite point sets $A$ and $B$
Ensure: The Directed Hausdorff distance $h(A, B)$
 1: $cmax \leftarrow 0$
 2: for all $a \in A$ do
 3:   $cmin \leftarrow \infty$
 4:   for all $b \in B$ do
 5:     $dist \leftarrow dist(a, b)$
 6:     if $dist < cmin$ then
 7:       $cmin \leftarrow dist$
 8:     end if
 9:   end for
10:   if $cmax < cmin$ then
11:     $cmax \leftarrow cmin$
12:   end if
13: end for
14: return $cmax$
```

Early Breaking. Taha A A et al.[32] creatively noted that it is not necessary to completely scan all points of the inner loop of the NAIVEHD algorithm.

Due to the Hausdorff distance being a MAX-MIN problem, once a distance below the temporary HD (*cmax*) is found, the *cmax* cannot be altered further in the remainder of this inner loop. Therefore, the inner loop can be broken early once the situation occurs. The EARLYBREAK method is shown in Algorithm 2. Line 10 was misprinted and was corrected by the author of [32].

Algorithm 2 EARLYBREAK, i.e., the early break, for computing the directed Hausdorff distance

```
Require: Two finite point sets $A$ and $B$
Ensure: The Directed Hausdorff distances $h(A, B)$
 1: $cmax \leftarrow 0$
 2: $C \leftarrow A \backslash (A \cap B)$
 3: $C_r \leftarrow randomize(C)$
 4: $B_r \leftarrow randomize(B)$
 5: for all $c \in C_r$ do
 6:     $cmin \leftarrow \infty$
 7:     for all $b \in B_r$ do
 8:         $dist \leftarrow dist(a, b)$
 9:         if $dist < cmax$ then
10:             $cmin \leftarrow 0$
11:             break
12:         end if
13:         if $dist < cmin$ then
14:             $cmin \leftarrow dist$
15:         end if
16:     end for
17:     if $cmax < cmin$ then
18:         $cmax \leftarrow cmin$
19:     end if
20: end for
21: return $cmax$
```

Taha A A et al. [32] also indicated that if no early break occurs at position X in the current loop, it is likely that the loop will not break at points near X. Therefore, EARLYBREAK uses random sampling to continue the search in another region that is spatially far from the current point.

4. The Proposed Method

Neighbor Points. Before the introduction of our algorithm, we first define the concept of neighbor points. There are two kinds of neighbor points: the left neighbor point and the right neighbor point. Figure 1 shows an example to facilitate comprehension. Given a sequence point set *abcde*, whose characteristic is that its points are adjacent in sequence and also adjacent in space, if c is the current point, b and d are its left and right neighbors, respectively.

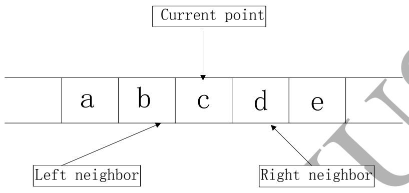


Figure 1: A sequence point set *abcde*, if c is the current point, b and d are its left and right neighbors, respectively.

Local Start Search. Now we present our algorithm. The Hausdorff Distance is a MAX-MIN problem that contains both maximization and minimization rather than just one or the other. The key to reducing the running time of HD is to reduce the average number of iterations in the inner loop. The focus of our research is how to reduce the average number of iterations in the inner loop. There's no need to fully scan the points in the inner loop because a majority of points do not contribute to HD. Therefore, how to quickly determine the non-contributing points is the key to reducing the number of iterations in the inner loop. A point is considered to be one of the non-contributing points if it has a distance $d < cmax$. This paper notes that if an early break occurs at position X in the current loop, it is quite possible that the break will occur at points near X in the next loop. Therefore, we present a strategy to fully utilize the spatial locality of point sets.

In the LSS algorithm, to quickly find a distance below $cmax$ in each inner loop, we use a variable *preindex* to preserve the location of break occurrence.

Therefore, execution of the next outer loop from the inner loop can start from *preindex* and scan its neighbors until finding a distance below *cmax*.

The pseudo-code of our method is elaborately described in Algorithm 3. Given two finite point sets A and B with point counts m and n, respectively, to deal with general-type point sets and obtain better locality of the data points, we use the well-known space-filling Morton curve for ordering points. The Morton curve is described in section 4.3. The loops in Algorithm 3 Line 3 and 5 are called the outer loop and the inner loop, respectively.

LSS For Scattered Point Sets. Furthermore, because the LSS algorithm is based on the spatial locality of point sets, we can process general point sets after ordering the sets.

Therefore, LSS integrates the well-known space-filling Morton curve, also called the z-order curve, for ordering points. The Morton code was first introduced in 1966 by [44]. Morton code can map multidimensional data to one dimension while preserving the locality of the data points. A point's order along the curve is determined by the Morton code. The calculation of the Morton code for a multidimensional point simply interleaves the binary representations of the point coordinate values.

Given a 3D point, we can easily quantize each of the three dimensional coordinates of the point into k-bit integers so that a 3D point is represented by a 3k-bit Morton code. The calculation of the Morton code simply involves interleaving the continuous bits of each quantized coordinate value. Figure 2 illustrates the Morton code construction process of the 2D points. Sorting the points in increasing order is done by sorting their Morton codes. Thus, the points are in order along a Morton curve. Because of this property, sorting unordered points based on their Morton codes has been used to represent the regularity of data points [45, 46].

Running time Analysis. The computational complexity of HD is $O(m * n)$ using the brute force method. It completely scans each point in the inner loop. EARLYBREAK has a running time of $O(m*n)$ in its worst case and a Running

Algorithm 3 LSS, i.e., local start search, for computing the directed HD

```
Require: Two finite point sets $A$ and $B$ with point counts m and n respectively.
Ensure: The Directed Hausdorff distances $h(A, B)$
1: $cmax \leftarrow 0$ , $preindex \leftarrow 0$ , $A_z \leftarrow Zorder(A)$, $B_z \leftarrow Zorder(B)$
2: for $cmin \leftarrow \infty$ , $minplace \leftarrow 0$, $i = 0; i < m; i++$ do
3:   for $j = 0; j < n; j++$ do
4:     while $0 <= preindex - j < n || 0 <= preindex + j < n$ do
5:       if $0 <= preindex - j < n$ then
6:         if $(distleft \leftarrow dist(A_z[i], B_z[preindex - j])) < cmax$ then
7:           $preindex \leftarrow preindex - j$, $cmin \leftarrow 0$
8:           break
9:         end if
10:        if $distleft < cmin$ then
11:          $cmin \leftarrow distleft$, $minplace \leftarrow preindex - j$
12:        end if
13:      end if
14:      if $0 <= preindex + j < n$ then
15:        if $(distright \leftarrow dist(A_z[i], B_z[preindex + j])) < cmax$ then
16:          $preindex \leftarrow preindex + j$, $cmin \leftarrow 0$
17:          break
18:        end if
19:        if $distright < cmin$ then
20:          $cmin \leftarrow distright$, $minplace \leftarrow preindex + j$
21:        end if
22:      end if
23:    end while
24:  end for
25:  if $cmax < cmin$ then
26:    $cmax \leftarrow cmin$ , $preindex \leftarrow minplace$
27:  end if
28: end for
29: return $cmax$
```

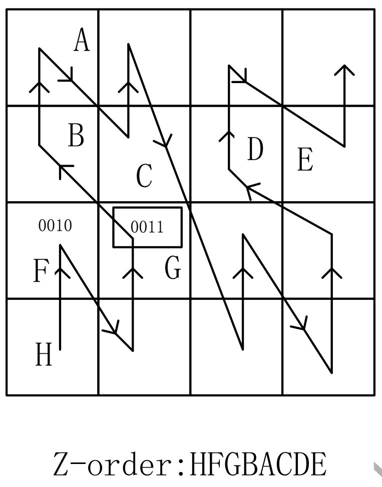


Figure 2: Example 2D Morton code for ordering of points.

time of $O(m)$ in its best case. The worst case is the same as for the brute force method. For another, the best case occurs when a break always occurs at the start of each iteration in the inner loop. Our algorithm, without the worst case, has a running time of $O(m)$ in its best case and a computational time of $O(m*k)$ in its general case. We use k to denote the average number of iterations in the inner loop. Because the best and worst case occur under certain conditions, we pay more attention to the general case. Obviously, the key to reducing the running time of HD is to reduce the average number of iterations in the inner loop. Now let us discuss our algorithm's effectiveness in reducing the average number of iterations through the analysis of probability theory.

In Algorithm 3, the distance measured a point $b \in B$ in the inner loop and the current reference point $a \in A$ is d. If the distance d is less than $cmax$, we define this event as e. Let us suppose that the probability of event e occurring is $P(e) = q$. Apparently, the probability of event $\overline{e}$ occurring is $p = 1 - q$. The event $\overline{e}$ means that d is larger than $cmax$. We also define the variable K as the number of successive events $\overline{e}$ until the event e occurs. The general formula for calculating the probability of k-1 events $\overline{e}$ $(d > cmax)$ before the first event e

$(d \leq cmax)$ is given by

$$P(K = k) = (1 - q)^(k - 1)q = p^(k - 1)q \qquad for \quad k = 1, 2, 3, ... \tag{4}$$

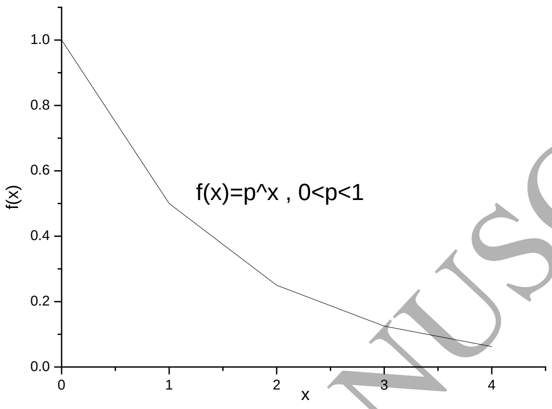


Figure 3: The probability density function of a geometrical distribution.

which is a geometric distribution. Figure 3 displays the probability density function $f(x)$ of geometrical distribution. According to probability knowledge, the expected value $E[K]$ of the geometric distribution indicates the average number of iterations before an early break occurs.

$$E[K] = \sum_{k=0}^{\infty} kf(k) = \sum_{k=0}^{\infty} kp^(k - 1)q \tag{5}$$

Then, Equation 5 can be developed in the form of a polynomial, as follows.

$$E[K] = q + 2pq + 3p^2q + 4p^3q + ...+ \tag{6}$$

We can find a simpler formula for this sum by multiplying both sides of the above equation by $1 - p$; all the other terms cancel.

$$(1 - p)E[K]/q = 1 + p + p^2 + p^3 + ... \tag{7}$$

Then, we can substitute $q = 1-p$ into Equation 7 to get the convenient formula.

$$E[K] = 1/(1 - p) = 1/q \tag{8}$$

It's easy to know the important fact that the number of iterations until a break occurs only depends on the probability of finding a point with a distance below *cmax* based on Equation 8. The higher q is, the lower the number of iterations in the inner loop and vice versa. Thus, as long as we get a high probability q, we can reduce the average number of iterations in the inner loop. In this paper, we note that a point that breaks in the current loop with high probability will break in the next loop. Based on this fact, we propose the local start search algorithm. Figure 4 shows the average number of iterations in the inner loop as outer loop changes. At the beginning of LSS, there must be one full scan of the inner loop to initialize *cmax* and *preindex*. After the initialization, the largest number of iterations of the inner loop is two. Thus, the curve drops sharply.

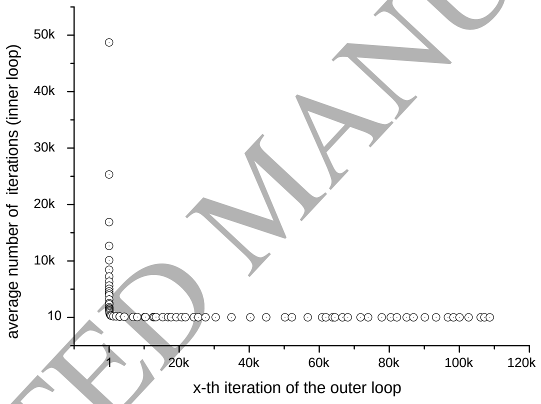


Figure 4: The average number of iterations in the inner loop as the outer loop changes.

Convergence of the cmax. Now we discuss the convergence of the temporary HD *cmax*. According to algorithm 3, we know that the value of *cmax* constantly increases and updates during the progress of the outer loop. After a small proportion of the total number of iterations, the value of *cmax* converges and reaches the same value as that of the true HD, as show in Figure 5. The value of *cmax* is strictly increased monotonically from zero with the progress of the outer loop before it converges to the true HD, which was obtained via the brute force method. In each iteration, the value of *cmax* is updated under the circumstances of a distance larger than *cmax*. There is no *cmax* update when the distance to the current point is smaller than *cmax*. Through such an update mechanism, *cmax* converges to the true Hausdorff Distance. Figure 5 indicates that the proposed algorithm has the features of rapid convergence of *cmax* to the true HD. After about five iterations in the outer loop, the value of *cmax* can reach about ninety percent of the true HD. And after about one hundred iterations, the value of *cmax* is very close to the true HD. We compared the final convergence value of *cmax* with the result of HD obtained by brute force; the value of both is the same. Based on this analysis, our algorithm obtains the exact solutions of HD.

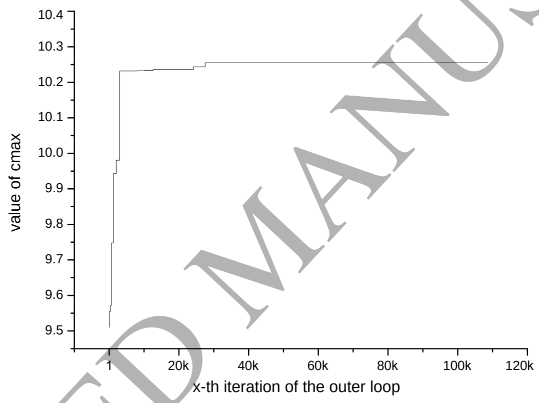


Figure 5: The value of temporary HD *cmax* convergences as the outer loop changes.

5. Experiments and Results

Our method is independent of the characteristics of the point sets. We have evaluated our method using three different types of data points: real brain tumor segmentations (MRI 3D volumes), the 3D bunny point data and trajectories generated from road network. Our methods have been compared with the state-of-the-art works [29, 32] and outperform it.

Evaluating using the real brain tumor segmentations, we have performed three different experiments, compared against the EARLYBREAK algorithm.

There are two versions of the proposed algorithm: one is with the Morton curve procedure for scattered point sets and the other one is without the Morton curve procedure for regular data. Through these experiments, the influence of the degree of overlap on the algorithm and the advantage of locality are shown. Obviously, a larger value of HD leads to a larger value of $cmax$ and vice versa. The first experiment (Section 5.1) calculated the HD ($H(A, B) = max(h(A, B), h(B, A))$) between non-overlapping images from the automatic brain tumor segmentations and the relevant ground truth segmentations. For the convenience of description, we denote the two contrast images from the automatic brain tumor segmentations and the relevant ground truth segmentations, respectively, as A and B in the experiments. This experiment explains the influence of non-overlapping on the proposed algorithm. The second experiment (Section 5.2) calculated the two directed HD ($h(B, A)$ *and* $h(A, B)$). This experiment illustrates the influence of partial overlapping between images for the proposed algorithm. The third experiment (Section 5.3) calculated the two directed HD ($h(A, B)$, A and B are the same) between the same two images. This experiment takes into account the worst case (completely overlapping) in EARLYBREAK (when $h(A, B)$ is zero), while our method obtains a good performance.

In the fourth experiment (Section 5.4), we used the widely used 3D point cloud model, i.e.,the Bunny model, and its simplification models to test our method. The purpose of this experiment is to prove that our method works for any type of point set, and is not limited to the regular point sets.

Finally, we used the trajectories to compare the proposed algorithm with incremental Hausdorff distance calculation algorithm (INC) [29] in the last experiment (Section 5.5).

The first four experiments were implemented on a desktop computer with an Intel Pentium E5400 CPU at 2.70GHZ (2 cores), 4 GB RAM, and an Nvidia GeForce GTX 650 GPU. The OS was Windows 7 (64 Bit). All algorithms were implemented in C++ using the VS2010 compiler. Morton code was parallel implemented using the NVIDIA CUDA programming language [47]. Parallel-implemented sorting was done using the Thrust library [1]. The last experiment was done under the same condition as the specification described in [29].

Non-Overlapping. In this experiment, the non-overlapping situation has been considered. We used real brain tumor segmentation that included automatic brain tumor segmentations (MRI 3D volumes), which were generated by segmentation algorithms proposed in the BRATS2012 challenge[2], and the corresponding ground truth segmentation produced by human specialists. Based on the fact that brain tumors exist in different places in the brain, we selected pairs of images from the two kinds of volumes. Each pair of images is non-overlapping. We compared against the state-of-the-art algorithm EARLYBREAK proposed by [32].

The performance of the proposed algorithm compared with that of the EARLYBREAK algorithm is illustrated in Figure 6. The execution time contains two directed HD execute times. Figure 6 shows that our algorithm achieves performance as good as that of the EARLYBREAK algorithm in the non-overlapping situation.

Partially Overlapping. The purpose of this experiment is to take the partial overlapping situation between two images into account. Thus, we chose two images to compute HD: one from the volumes generated by segmentation algorithms (A) and one from the corresponding ground truth (B). The excluding intersection strategy in the EARLYBREAK algorithm was designed to solve the overlapping situation limited to voxel data with the same grid size, which lacks commonality. If two point sets have m and n points, respectively, with different grid size, then the computational complexity of the excluding intersection strategy is $O(m * n)$. Therefore, here, we use the EARLYBREAK algorithm without the excluding intersection strategy in all experiments for generality to compare

[1] Thrust Library, www.cudazone.nvidia.cn/thrust/.

[2] MICCAI 2012 Challenge on Multimodal Brain Tumor Segmentation, www.imm.dtu.dk/projects/BRATS2012

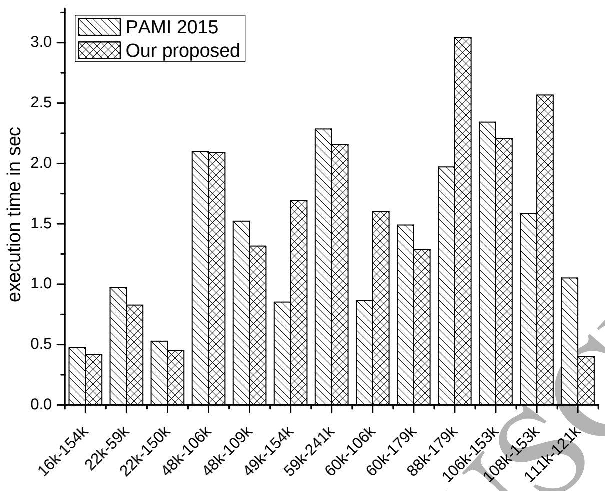


Figure 6: Non-Overlapping: Testing with non-overlapping images; X and Y axis represent the point size of pair images in kilo voxels and the execution time in sec, respectively.

with our algorithm in this paper. The total execution time of HD computation include two directed HD computation times. We test using the volumes generated by segmentation algorithms and the corresponding ground truth produced by human specialists, which are partially overlapping.

Table 1 indicates that the total execution time of our algorithm is far less than that of the EARLYBREAK algorithm proposed in PAMI 2015. We need to emphasize the reason why the two directed HD computation times varied considerably in PAMI 2015. When computing $h(A, B)$, at the first outer loop, a large $cmax$ was obtained; then, the subsequent loop broke after a small number of iterations. While computing $h(B, A)$, in the long iterations the value of $cmax$ stayed at zero, which meant no early break occurred. The proposed algorithm requires almost less than one sec in all moderately sized point sets, even for large-scale point sets with very high efficiency. Although the two data sets are partially overlapping, our method still can quickly determine a distance below $cmax$ and then break the loop. The details of the computational complexity analysis were given in section 4.4.

set size	Our Proposed		PAMI 2015	
n1...n2	$h(A, B)$	$h(B, A)$	$h(A, B)$	$h(B, A)$
16k-388k	0.485 sec	0.108 sec	0.137 sec	897.85 sec
34k-517k	0.477 sec	0.250 sec	0.229 sec	261.67 sec
42k-542k	0.314 sec	0.109 sec	0.203 sec	2735.5 sec
77k-479k	0.671 sec	0.188 sec	0.432 sec	1264.7 sec
106k-1198k	1.077 sec	0.566 sec	1.868 sec	4769.5 sec
108k-580k	2.435 sec	0.383 sec	1.254 sec	3073.8 sec
179k-720k	1.024 sec	1.585 sec	1.467 sec	122.96 sec

Table 1: Partially Overlapping:Testing with the volumes generated by segmentation algorithms and the corresponding ground truth produced by human specialists, which are partially overlapping.

Completely Overlapping. The aim of this experiment was to demonstrate the performance of the proposed algorithm in the completely overlapping situation, which is the worst case for the EARLYBREAK algorithm. In the completely overlapping situation, the EARLYBREAK algorithm cannot break early until scanning all the points in the inner loop; it then degenerates to the NAIVEHD algorithm. Meanwhile, the proposed algorithm remains efficient as in the other situations.

Table 2 indicates that the proposed algorithm achieved a very efficient performance in the completely overlapping situation. Computing HD between 1590k-1590k pairs of point sets requires less than one sec. The EARLYBREAK algorithm execution time was too long to run. The experiment result indicates that using the spatial locality of point sets can make it easy to handle the overlap situation.

Scattered Point Sets. To evaluate the generality of the proposed algorithm, we used the famous Bunny point set provided by the Stanford 3D Scanning Repository in this experiment. We obtained five different-sized bunny point sets through by simplification [48]. Comparing with two different size bunny

set size	Our Proposed		PAMI 2015	
n1...n2	$h(A,B)$	$h(B,A)$	$h(A,B)$	$h(B,A)$
16k-16k	0.004 sec	0.005 sec	14.18 sec	14.61 sec
106k-106k	0.060 sec	0.054 sec	1361 sec	1409 sec
241k-241k	0.088 sec	0.096 sec	5383 sec	4823 sec
1198k-1198k	0.582 sec	0.506 sec	too long	too long
1590k-1590k	0.486 sec	0.496 sec	too long	too long

Table 2: Completely Overlapping: Testing with two same images, in this case, the HD equals zero. The EARLYBREAK algorithm cannot break early until scanning all the points in the inner loop. While the proposed algorithm can break after several iterations in the inner loop.

point sets (Non-Overlapping situation), we firstly adopted the Morton Curve to represent the regularity of scattered point sets. Then, after the preprocessing, the LSS algorithm was executed. Because the preprocessing involves easy parallel implementation on GPU hardware, the time required by the preprocessing, which is short, can be ignored. The randomization time of the EARLYBREAK algorithm is also not contained.

Figure 7 indicates that the proposed algorithm can process scattered point sets and achieve a better performance than the EARLYBREAK algorithm. When dealing with 5k-10k pairs of point sets, the proposed algorithm requires 0.22 sec, while the EARLYBREAK algorithm requires 5.94 sec. The results show that our algorithm outperforms the EARLYBREAK algorithm by approximately thirty times.

Comparing Against Incremental Hausdorff Distance. Nutanong et al. [29] proposed the incremental Hausdorff distance calculation algorithm (INC) based on R-Trees. In this experiment, the proposed algorithm compared against the incremental Hausdorff distance calculation algorithm (INC) under the same condition as the specification described in [29]. The test dataset is trajectories

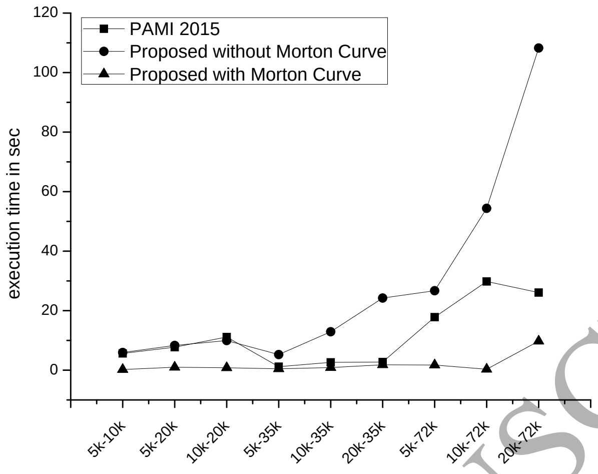
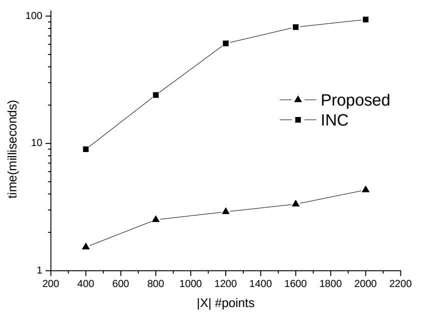


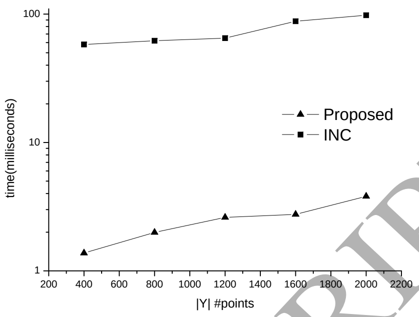
Figure 7: Scattered Point Sets: Testing with Bunny point sets; X and Y axis represent the size of pair point sets and the execution time in sec, respectively.

generated from the Oldenburg road network[3]. Each trajectory is the shortest path between two randomly picked points from the road network with a length of 2000 units and has five different resolutions. The dataset contains five sets according to five different resolution: 400, 800, 1200, 1600 and 2000 sampled points. In one experiment, we varied the size of Y with a fixed size of X to compute HD(X,Y) between two trajectories. And in the other one experiment, we varied the size of X with a fixed size of Y to compute HD(X,Y) between two trajectories. The results of these experiments are compared with the results published in [29], Section 7.1, Figure 8. Each result is the average of 200 different pairs of trajectories. Both Figures 8(a) and 8(b) show that the proposed algorithm significantly outperforms the INC algorithm by an order of magnitude faster.

[3]Oldenburg road network, www.cs.utah.edu/ lifeifei/SpatialDataset.htm



(a) The size of X is fixed and the size of Y varies.



(b) The size of Y is fixed and the size of X varies.

Figure 8: The proposed algorithm compared with the incremental hausdorff distance calculation algorithm (INC) in the execution time of the Hausdorff Distance computation HD(X,Y). X and Y axis represent the size of the point set and the execution time in milliseconds, respectively.

6. Conclusion and future work

This paper proposed a novel, efficient and general algorithm for computing the exact Hausdorff Distance for arbitrary point sets, which outperforms the state-of-the-art algorithm. We find and discover the spatial locality feature of point sets. A point is considered to be likely to cut the search space if it is close to a point that break in the previous iteration. Empirical evidence shows a substantial improvement in the cases of partially- or fully-overlapping sets.

In future work, we will explore three directions but not limited. (1) Since many-core GPU has been a popular acceleration platform in scientific computation [49, 50, 51, 52, 53], we will try to accelerate our algorithm with GPU. (2) We also want to extend our idea to other areas in computer vision and intelligent computing [54, 55, 56, 57, 58, 59, 60]. (3) We should also absorb the ideas from other successful nonlinear models, such as surface irrigation simulation models and temperature-based models [61, 62, 63, 64, 65].

Acknowledgments

In this paper, the point sets were obtained from the MICCAI 12 BRATS challenge[66], the Stanford 3D Scanning Repository [67] and the Real Datasets for Spatial Databases. We would like to thank all the anonymous reviewers for their valuable comments. This work is supported by the National Science Foundation of China (Grant No. 61472289) and the National Key Research and Development Project (Grant No. 2016YFC0106305).

References

[1] N. Sudha, et al., Robust hausdorff distance measure for face recognition, Pattern Recognition 40 (2) (2007) 431–442. doi:10.1016/j.patcog.2006.04.019.

[2] K. Lin, K. Lam, W.-C. Siu, Spatially eigen-weighted hausdorff distances for human face recognition, Pattern Recognition 36 (8) (2003) 1827–1834. doi:10.1016/S0031-3203(03)00011-6.

[3] Y. Gao, M. K. Leung, Face recognition using line edge map, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (6) (2002) 764–779. doi:10.1109/TPAMI.2002.1008383.

[4] E. Sangineto, Pose and expression independent facial landmark localization using dense-surf and the hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (3) (2013) 624–638. doi:10.1109/TPAMI.2012.87.

[5] H. Alt, B. Behrends, J. Blömer, Approximate matching of polygonal shapes, Annals of Mathematics and Artificial Intelligence 13 (3-4) (1995) 251–265. doi:10.1007/BF01530830.

[6] L. B. Kara, T. F. Stahovich, An image-based trainable symbol recognizer for sketch-based interfaces, in: AAAI Fall Symposium, 2004, pp. 99–105. doi:10.1016/j.cag.2005.05.004.

[7] M. Field, S. Valentine, J. Linsey, T. Hammond, Sketch recognition algorithms for comparing complex and unpredictable shapes, in: IJCAI, Vol. 11, 2011, pp. 2436–2441. doi:10.5591/978-1-57735-516-8/IJCAI11-406.

[8] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, H. Bunke, Approximation of graph edit distance based on hausdorff matching, Pattern Recognition 48 (2) (2015) 331–343. doi:10.1016/j.patcog.2014.07.015.

[9] A. Jain, V. Kanhangad, Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures, Pattern Recognition Letters 68 (2015) 351–360. doi:10.1016/j.patrec.2015.07.004.

[10] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. Van Nguyen, R. Ohbuchi, et al., A comparison of methods for non-rigid 3d shape retrieval, Pattern Recognition 46 (1) (2013) 449–461. doi:10.1016/j.patcog.2012.07.014.

[11] Y. Kim, Y. Oh, S. Yoon, M. Kim, G. Elber, Efficient hausdorff distance computation for freeform geometric models in close proximity, Computer-Aided Design 45 (2) (2013) 270–276. doi:10.1016/j.cad.2012.10.010.

[12] D. Zhang, F. He, S. Han, X. Li, Quantitative optimization of interoperability during feature-based data exchange, Integrated Computer-Aided Engineering 23 (1) (2016) 31–50. doi:10.3233/ICA-150499.

[13] X. Li, F. He, X. Cai, D. Zhang, Y. Chen, A method for topological entity matching in the integration of heterogeneous cad systems, Integrated Computer-Aided Engineering 20 (1) (2013) 15–30. doi:10.3233/ICA-120416.

[14] X. Lv, F. He, W. Cai, Y. Cheng, A string-wise crdt algorithm for smart and large-scale collaborative editing systems, Advanced Engineering Informatics Online. doi:DOI10.1016/j.aei.2016.10.005.

[15] Y. Cheng, F. He, Y. Wu, D. Zhang, Meta-operation conflict resolution for human-human interaction in collaborative feature-based cad systems, Cluster Computing 19 (1) (2016) 237–253. doi:10.1007/s10586-016-0538-0.

[16] Y. Wu, F. He, D. Zhang, X. Li, Service-oriented feature-based data exchange for cloud-based design and manufacturing, IEEE Transactions on Services Computing Online. doi:10.1109/TSC.2015.2501981.

[17] Y. Gao, M. Wang, R. Ji, X. Wu, Q. Dai, 3-d object retrieval with hausdorff distance learning, Industrial Electronics, IEEE Transactions on 61 (4) (2014) 2088–2098. doi:10.1109/TIE.2013.2262760.

[18] M. Grana, M. A. Veganzones, An endmember-based distance for content based hyperspectral image retrieval, Pattern Recognition 45 (9) (2012) 3472–3489. doi:10.1016/j.patcog.2012.03.015.

[19] B. Leng, J. Zeng, M. Yao, Z. Xiong, 3d object retrieval with multitopic model combining relevance feedback and lda model, IEEE Transactions on Image Processing 24 (1) (2015) 94–105. doi:10.1109/TIP.2014.2372618.

[20] M. Guthe, P. Borodin, R. Klein, Fast and accurate hausdorff distance calculation between meshes, Journal of WSCG 13 (2) (2005) 41–48.

[21] M. Mandad, D. Cohen-Steiner, P. Alliez, Isotopic approximation within a tolerance volume, ACM Transactions on Graphics (TOG) 34 (4) (2015) 64. doi:10.1145/2766950.

[22] P. Li, B. Wang, F. Sun, X. Guo, C. Zhang, W. Wang, Q-mat: Computing medial axis transform by quadratic error minimization, ACM Transactions on Graphics (TOG) 35 (1) (2015) 8. doi:10.1145/2753755.

[23] K. O. Babalola, B. Patenaude, P. Aljabar, J. Schnabel, D. Kennedy, W. Crum, S. Smith, T. F. Cootes, M. Jenkinson, D. Rueckert, Comparison and evaluation of segmentation techniques for subcortical structures in brain mri, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008, Springer, 2008, pp. 409–416. doi:10.1007/978-3-540-85988-8_49.

[24] H. Khotanlou, O. Colliot, J. Atif, I. Bloch, 3d brain tumor segmentation in mri using fuzzy classification, symmetry analysis and spatially constrained deformable models, Fuzzy Sets and Systems 160 (10) (2009) 1457–1473. doi:10.1016/j.fss.2008.11.016.

[25] F. Nicolier, S. Lebonvallet, E. Baudrier, S. Ruan, Hausdorff distance based 3d quantification of brain tumor evolution from mri images, in: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, IEEE, 2007, pp. 5597–5600. doi:10.1109/IEMBS.2007.4353615.

[26] B. Ni, F. He, Y. Pan, Z. Yuan, Using shapes correlation for active contour segmentation of uterine fibroid ultrasound images in computer-aided therapy, Applied Mathematics-A Journal of Chinese Universities 31 (1) (2016) 37–52. doi:10.1007/s11766-016-3340-0.

[27] B. Ni, F. He, Z. Yuan, Segmentation of uterine fibroid ultrasound images using a dynamic statistical shape model in hifu therapy, Computerized Medical Imaging and Graphics 46 (2015) 302–314. doi:10.1016/j.compmedimag.2015.07.004.

[28] H. Yu, F. He, Y. Pan, X. Chen, An efficient similarity-based level set model for medical image segmentation, Journal of Advanced Mechanical Design, Systems, and Manufacturing 10 (8) (2016) 16–62. doi:10.1299/jamdsm.2016jamdsm0100.

[29] S. Nutanong, E. H. Jacox, H. Samet, An incremental hausdorff distance calculation algorithm, Proceedings of the VLDB Endowment 4 (8) (2011) 506–517. doi:10.14778/2002974.2002978.

[30] M. J. Hossain, M. A. A. Dewan, K. Ahn, O. Chae, A linear time algorithm of computing hausdorff distance for content-based image analysis, Circuits, Systems, and Signal Processing 31 (1) (2012) 389–399. doi:10.1007/s00034-011-9284-y.

[31] G. Gerig, M. Jomier, M. Chakos, Valmet: A new validation tool for assessing and improving 3d object segmentation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001, Springer, 2001, pp. 516–523. doi:10.1007/3-540-45468-3_62.

[32] A. A. Taha, A. Hanbury, An efficient algorithm for calculating the exact hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (11) (2015) 2153–2163. doi:10.1109/TPAMI.2015.2408351.

[33] D. P. Huttenlocher, G. A. Klanderman, W. J. Rucklidge, Comparing images using the hausdorff distance, Pattern Analysis and Machine Intelligence, IEEE Transactions on 15 (9) (1993) 850–863. doi:10.1109/34.232073.

[34] H. Alt, L. J. Guibas, Discrete geometric shapes: Matching, interpolation, and approximation, Handbook of computational geometry 1 (1999) 121–153. doi:10.1.1.57.1364.

[35] M. Tang, M. Lee, Y. J. Kim, Interactive hausdorff distance computation for general polygonal models, in: ACM Transactions on Graphics (TOG), Vol. 28, ACM, 2009, p. 74. doi:10.1145/1531326.1531380.

[36] L. Zhang, Y. J. Kim, G. Varadhan, D. Manocha, Generalized penetration depth computation, Computer-Aided Design 39 (8) (2007) 625–638. doi:10.1016/j.cad.2007.05.012.

[37] C. Je, M. Tang, Y. Lee, M. Lee, Y. J. Kim, Polydepth: Real-time penetration depth computation using iterative contact-space projection, ACM Transactions on Graphics (TOG) 31 (1) (2012) 5. doi:10.1145/2077341.2077346.

[38] F. Lafarge, P. Alliez, Surface reconstruction through point set structuring, in: Computer Graphics Forum, Vol. 32, Wiley Online Library, 2013, pp. 225–234. doi:10.1111/cgf.12042.

[39] M. Bartoň, I. Hanniel, G. Elber, M.-S. Kim, Precise hausdorff distance computation between polygonal meshes, Computer Aided Geometric Design 27 (8) (2010) 580–591. doi:10.1016/j.cagd.2010.04.004.

[40] Y. Kim, Y. Oh, S. Yoon, M. Kim, G. Elber, Precise hausdorff distance computation for planar freeform curves using biarcs and depth

buffer, The Visual Computer 26 (6-8) (2010) 1007–1016. doi:10.1007/s00371-010-0477-3.

[41] X. D. Chen, W. Ma, G. Xu, J. C. Paul, Computing the hausdorff distance between two b-spline curves, Computer-Aided Design 42 (12) (2010) 1197–1206. doi:10.1016/j.cad.2010.06.009.

[42] M. J. Atallah, A linear time algorithm for the hausdorff distance between convex polygons, Information processing letters 17 (4) (1983) 207–209. doi:10.1016/0020-0190(83)90042-X.

[43] A. Krishnamurthy, S. McMains, I. Hanniel, Gpu-accelerated hausdorff distance computation between dynamic deformable nurbs surfaces, Computer-Aided Design 43 (11) (2011) 1370–1379. doi:10.1016/j.cad.2011.08.022.

[44] G. M. Morton, A computer oriented geodetic data base and a new technique in file sequencing, International Business Machines Company New York, 1966.

[45] T. Karras, Maximizing parallelism in the construction of bvhs, octrees, and k-d trees, in: Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics, Eurographics Association, 2012, pp. 33–37. doi:10.2312/EGGH/HPG12/033-037.

[46] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, D. Manocha, Fast bvh construction on gpus, in: Computer Graphics Forum, Vol. 28, Wiley Online Library, 2009, pp. 375–384. doi:10.1111/j.1467-8659.2009.01377.x.

[47] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with cuda, Queue 6 (2) (2008) 40–53. doi:10.1145/1365490.1365500.

[48] M. Pauly, M. Gross, L. P. Kobbelt, Efficient simplification of point-sampled surfaces, in: Proceedings of the conference on Visualization'02, IEEE Computer Society, 2002, pp. 163–170. doi:10.1109/VISUAL.2002.1183771.

[49] T. Van Luong, N. Melab, E. G. Talbi, Gpu computing for parallel local search metaheuristic algorithms, IEEE Transactions on Computers 62 (1) (2013) 173–185. doi:10.1109/TC.2011.206.

[50] S. Williams, A. Waterman, D. Patterson, Roofline: an insightful visual performance model for multicore architectures, Communications of the ACM 52 (4) (2009) 65–76. doi:10.1145/1498765.1498785.

[51] S. Dalton, L. Olson, N. Bell, Optimizing sparse matrixmatrix multiplication for the gpu, ACM Transactions on Mathematical Software (TOMS) 41 (4) (2015) 25. doi:10.1145/2699470.

[52] Y. Zhou, F. He, Y. Qiu, Optimization of parallel iterated local search algorithms on graphics processing unit, The Journal of Supercomputing 72 (2016) 2394–2416. doi:10.1007/s11227-016-1738-3.

[53] Y. Zhou, F. He, Y. Qiu, Dynamic strategy based parallel ant colony optimization on gpus for tsps, Sci. China. Ser. Online.doi:10.1007/s11432-015-0594-2.

[54] D. Lee, J. Sim, C. Kim, Visual tracking using pertinent patch selection and masking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3486–3493. doi:10.1109/CVPR.2014.446.

[55] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (3) (2015) 583–596. doi:10.1109/TPAMI.2014.2345390.

[56] K. Li, F. He, X. Chen, Real-time object tracking via compressive feature selection, Frontiers of Computer Science (2016) onlinedoi:10.1007/s11704-016-5106-5.

[57] Z. Huang, F. He, X. Cai, Z. Zou, J. Liu, M. Liang, X. Chen, Efficient random saliency map detection, Science China Information Sciences 54 (6) (2011) 1207–1217. doi:10.1007/s11432-011-4263-2.

[58] J. Sun, F. He, Y. Chen, X. Chen, A multiple template approach for robust tracking of fast motion target, Appl. Math. J. Chinese Univ. 31 (2) (2016) 177–197. doi:10.1007/s11766-016-3378-z.

[59] X. Yan, F. He, Y. Chen, Z. Yuan, An efficient improved particle swarm optimization based on prey behavior of fish schooling, Journal of Advanced Mechanical Desigh, Systems, and Manufacturing 9 (4) (2015) 15–00009. doi:10.1299/jamdsm.2015jamdsm0048.

[60] X. Yan, F. He, N. Hou, A novel hardware/software partitioning method based on position disturbed particle swarm optimization with invasive weed optimization, Journal of Computer Science and Technology, Accepted.

[61] M. Valipour, Comparison of surface irrigation simulation models: full hydrodynamic, zero inertia, kinematic wave, Journal of Agricultural Science 4 (12) (2012) 68. doi:10.5539/jas.v4n12p68.

[62] M. Valipour, Sprinkle and trickle irrigation system design using tapered pipes for pressure loss adjusting, Journal of Agricultural Science 4 (12) (2012) 125. doi:10.5539/jas.v4n12p125.

[63] M. Valipour, S. Eslamian, Analysis of potential evapotranspiration using 11 modified temperature-based models, International Journal of Hydrology Science and Technology 4 (3) (2014) 192–207. doi:10.1504/ijhst.2014.067733.

[64] M. Valipour, M. E. Banihabib, S. M. R. Behbahani, Comparison of the arma, arima, and the autoregressive artificial neural network models in forecasting the monthly inflow of dez dam reservoir, Journal of hydrology 476 (2013) 433–441. doi:10.1016/j.jhydrol.2012.11.017.

[65] M. Mahdizadeh Khasraghi, M. A. Gholami Sefidkouhi, M. Valipour, Simulation of open-and closed-end border irrigation systems using sirmod, Archives of Agronomy and Soil Science 61 (7) (2015) 929–941. doi:10.1080/03650340.2014.981163.

[66] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al., The multimodal brain tumor image segmentation benchmark (brats), IEEE Transactions on Medical Imaging 34 (10) (2015) 1993–2024. doi:10.1109/TMI.2014.2377694.

[67] A. Gardner, C. Tchou, T. Hawkins, P. Debevec, Linear light source reflectometry, in: ACM Transactions on Graphics (TOG), Vol. 22, ACM, 2003, pp. 749–758. doi:10.1145/1201775.882342.

Author Biography

Yilin Chen is currently a Ph.D. candidate in State Key Laboratory of Software Engineering, School of Computer, Wuhan University. His research interests are Pattern Recognition, Image Processing and Computer Graphics.

Fazhi He received Ph.D. degree from Wuhan University of Technology. He was post-doctor researcher in The State Key Laboratory of CAD&CG at Zhejiang University, a visiting researcher in Korea Advanced Institute of Science & Technology and a visiting faculty member in the University of North Carolina at Chapel Hill. Now he is a professor in State Key Laboratory of Software Engineering, School of Computer, Wuhan University. His research interests are Computer Graphics, Computer-Aided Design, Image Processing and Computer Supported Cooperative Work.

Yiqi Wu is currently a Ph.D candidate in State Key Laboratory of Software Engineering, School of Computer, Wuhan University. His research interests include Computer Graphics, Computer Aided Design, Computer Supported Cooperative Work, Cloud-based Design and Manufacturing.

Neng Hou is currently a Ph.D. candidate in State Key Laboratory of Software Engineering, School of Computer, Wuhan University. His research interests include Hardware/Software Co-design, Intelligent Optimization Algorithms, and GPU Computing.