An Efficient Algorithm for Calculating the Exact Hausdorff Distance

Abdel Aziz Taha and Allan Hanbury

Abstract—The Hausdorff distance (HD) between two point sets is a commonly used dissimilarity measure for comparing point sets and image segmentations. Especially when very large point sets are compared using the HD, for example when evaluating magnetic resonance volume segmentations, or when the underlying applications are based on time critical tasks, like motion detection, then the computational complexity of HD algorithms becomes an important issue. In this paper we propose a novel efficient algorithm for computing the exact Hausdorff distance. In a runtime analysis, the proposed algorithm is demonstrated to have nearly-linear complexity. Furthermore, it has efficient performance for large point set sizes as well as for large grid size; performs equally for sparse and dense point sets; and finally it is general without restrictions on the characteristics of the point set. The proposed algorithm is tested against the HD algorithm of the widely used national library of medicine insight segmentation and registration toolkit (ITK) using magnetic resonance volumes with extremely large size. The proposed algorithm outperforms the ITK HD algorithm both in speed and memory required. In an experiment using trajectories from a road network, the proposed algorithm significantly outperforms an HD algorithm based on R-Trees.

Index Terms—Hausdorff distance, algorithm, evaluation, runtime analysis, computational complexity

1 Introduction

The Hausdorff distance (HD) is a measure of dissimilarity between two point sets. The HD is an important metric that is commonly used in many domains like image processing and pattern matching as well as evaluating the quality of clustering. For example it is common to use the Hausdorff distance in the medical domain in applications like evaluation of medical segmentations and registration. In many cases medical images, such as magnetic resonance (MRI) and computed tomography (CT) volumes are compared e.g., to evaluate the performance of registration [4], [5] and segmentation algorithms [19], [3], [15].

There are various types of measures used to compare two point clouds. *Overlap based measures*, e.g., the Dice coefficient, consider an imaginary grid on the union of the two point sets and calculate the overlap (intersection) between the point sets with respect to the grid. Points are assigned to subsets depending on whether they are or are not in the intersection to build the confusion matrix. The drawback of such measures is that the positions of the points are ignored: once a point is not in the intersection, it makes no difference where it is located [16]. Furthermore, such measures are not suitable for sparse point sets [24]. *Information theoretic based measures* like the mutual information consider entropy and joint entropy. *Probabilistic based measures* such as the distance between densities generally compare probability densities and other statistics of the point sets to measure the similarity [20], [14]. Measures in the last two categories are known to be robust, but not sensitive to the spatial positions of the points and thus not suitable for applications where the positions of the individual points are important [23]. *Spatial distance based measures* generally consider the pairwise distances between the compared point sets. Examples from this category are the average distance, i.e., the average of all pairwise distances, and the Mahalanobis distance, which compares estimates of the point sets as two hyper-ellipses. Both examples are also not sensitive to the positions of the individual points because in the case of the average distance, distances of far points are compensated by other near points and in the case of the Mahalanobis distance, estimating the point sets as hyper-ellipses means ignoring details of the positions of the points. The HD is a max-min distance, and hence has the advantage that it takes into consideration the spatial position of each individual point, which makes it capable of considering the spatial properties in the measurement, e.g., the boundary of an object. However, in some applications this makes it sensitive to outliers [7], [12].

The directed Hausdorff distance $\check{H}$ between two point sets A and B is the maximum of distances between each point $x \in A$ to its nearest neighbor $y \in B$. That is

$$\check{H}(A,B) = max_{x \in A}\{min_{y \in B}\{||x,y||\}\}, \quad (1)$$

where $||.,.||$ is any norm e.g., the euclidean distance function. Note that $\check{H}(A,B) \neq \check{H}(B,A)$ and thus the directed Hausdorff distance is not symmetric. The Hausdorff distance H is the maximum of the directed Hausdorff distances in both directions and thus it is symmetric. H is given by

$$H(A,B) = max\{\check{H}(A,B), \check{H}(B,A)\}. \quad (2)$$

• *The authors are with the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Vienna, Austria. E-mail: {taha, hanbury}@ifs.tuwien.ac.at.*

Manuscript received 12 Nov. 2013; revised 29 Jan. 2015; accepted 18 Feb. 2015. Date of publication 2 Mar. 2015; date of current version 7 Oct. 2015. Recommended for acceptance by R. Vidal.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2408351

For a point set A, we define the point set size to be the number of elements in A. Images and image volumes are a special class of point sets, where the elements are pixels (or voxels for volumes) that are in pre-defined locations on a grid. For an image A, we define the point set size to be the number of pixels/voxels in A that are not in the background (non-zero pixels/voxels). Also we define the grid size to be the dimensions of the entire image including background (width x length x height). Note that the proposed algorithm is equivalently applicable on images and volumes, so we will not strictly differentiate between them. The same applies to pixel and voxel.

Many researchers have noted the computational complexity of the HD [8], [11], [21]. The most important characteristics to optimize are runtime and memory required. However, evaluating the quality of a HD algorithm should take into consideration how these two characteristics vary in relation to the following parameters:

- Point set size. For example, a brain MRI volume could reach a million voxels and that of a whole body could reach 10 million voxels. The runtime of the algorithm should stay reasonable when the set size increases extremely.
- Grid size. It is desirable that the complexity of the algorithm depends only on the point set size rather than the grid size. For example, in brain tumor segmentations, the volume of the tumor is normally a small fraction of the grid size and the rest is background. The background should not be included in the computation.
- Density and sparsity. An algorithm could perform better with sparse point sets like geographical locations and worse with dense point sets like MRI segmentations and vice versa.
- Generality. Algorithms restricted to a special class of point sets cannot be applied in a general situation.

The algorithm proposed in this paper is optimized to efficiently perform in all situations above. It has a nearly-linear complexity and an efficient performance for extreme point set sizes as well as for extreme grid size. It outperforms the standard HD algorithm of the ITK Library,[1] the leading platform for image processing in the medical domain. Furthermore the proposed algorithm performs equally for sparse and dense point sets, and finally it is general without restrictions on the characteristics of the point set. The remainder of the paper is organized as follows. Related work is discussed in Section 2. In Section 3 we propose the novel algorithm for computing the Hausdorff distance and provide a runtime analysis of the algorithm. Experiments and results are presented in Section 4 and finally the paper is concluded in Section 5.

2 Related Work

Several approaches have been proposed that aim to overcome the computational complexity of the Hausdorff distance. These approaches can be generally divided into two categories, namely approximation and exact calculation of the Hausdorff distance. The first category contains those methods that try to efficiently find an approximation of the Hausdorff distance. This category is especially common with runtime-critical applications, for example pattern matching under transformation (e.g., moving object detection). Because the method proposed in this paper aims to calculate the exact Hausdorff distance, this category of research is actually not directly in the focus of this paper; we therefore only give some representative references for this category. Alt et al. [1] used Voronoi diagrams to efficiently approximate the HD between simple polygons. Indyk et al. [13] proposed an algorithm for approximating the HD for matching patterns under transformation by using the Halls Theorem to reduce the necessary geometrical matching. Hossain et al. [11] proposed a linear time algorithm for finding an approximation of the HD with lower approximation error. Most algorithms belonging to the second category try to efficiently compute the exact Hausdorff distance for specific classes of point sets or special types of objects like polygons, line segments, or special curves. The rest of these algorithms use complex structures that require a preprocessing phase causing long computation time and high memory need. In the next sections we highlight some work related to this category in more detail.

2.1 Polygons

Atallah [2] provided an algorithm for computing the Hausdorff distance for a special case of point sets, namely non-intersecting, convex polygons. The algorithm has the complexity of $O(n+m)$ where m and n are the vertex counts. The algorithm is mainly based on the fact that when minimizing/maximizing distances between two non-intersecting convex polygons, then points with extreme distances are always the vertices. This implies that only distances between vertices need to be computed to find the Hausdorff distance. Although this algorithm is simple and computationally efficient, it is restricted to a special class of point sets.

2.2 R-Trees

Papadias et al. [22] proposed an algorithm for finding aggregate nearest neighbors (ANN) in databases. Given two databases in form of point sets A and B, the algorithm finds for a given point $a \in A$ the nearest point $b \in B$. That is $ANN(a,B) = b \in B : dist(a,b) = minddist(a,B)$. The query data points are spatially indexed to produce an R-Tree which is then used to optimize searching for the ANN. In fact, $ANN(a,B)$ can be an elementary function for computing the directed Hausdorff distance because $\check{H}(A,B) = max_{a \in A} xANN(a,B)$. But because the algorithm deals with B as a single object, it follows that the direct use of ANN to compute Hausdorff distance means iterating all points $a \in A$ and performing ANN each time. Nutanong et al. [21] extended the algorithm proposed in [22] by avoiding the iteration of all points in A: the algorithm achieves this by performing the aggregate nearest neighbor simultaneously in both directions, that is to use two R-Trees at the same time, one for each point set.

However, the drawbacks of both methods above are (i) they use complex structures with additional computational

1. National Library of Medicine insight segmentation and registration toolkit www.itk.org.

effort needed for building the index and (ii) the methods assume sparse point sets that are suitable for building an efficient R-Tree. If the underlying point sets are very dense or in the worst case rigid objects (e.g., medical segmentation), algorithms based on R-Trees may not be the best choice.

2.3 Distance Transform Based Algorithms

A distance transform (called also distance maps) is a representation of an image in which each pixel becomes a label that reflects its distance to the boundary or background. There are various transforms depending on the distance metric used [18]. A common way to efficiently compute the HD in image processing is to use the distance transform. These methods compute the Hausdorff distance in linear time, given a distance transform, but the time required for computing the distance transform is proportional to the grid size, as it also takes background into account. Furthermore, these methods are based on labeling the pixels which makes them restricted to images, and thus they are not general. The ITK Library[1] uses distance transforms for computing the HD, described in [25]. Ciesielski et al. [6] investigated the computational complexity of the algorithm described in [25]. We use the ITK implementation of this algorithm as a reference to compare the performance of the proposed algorithm in Section 4.

2.4 HD for Mesh Surfaces

Guthe et al. [10] proposed an algorithm for calculating the Hausdorff distance between mesh surfaces. This algorithm makes use of the specific characteristics of meshes to avoid sampling all points in the compared surfaces. To achieve this, two strategies are used. In the first strategy, the algorithm aims to recognize areas in the two compared surfaces where the pairwise triangles are expected to have maximum distance between them. Only these areas are intensively sampled thereby avoiding sampling all triangles of the surfaces. This is achieved by building a grid, in particular an octree, on each of the surfaces and then calculating the min/max distances between cells. The second strategy is to avoid sampling all points in a particular triangle when calculating the min/max distances of the cells. This is achieved by measuring the distances of the triangle vertices to the other mesh surface in a first step. Then, sampling further points inside the triangle is stopped if all distances of the vertices are less than the actual (yet unknown) HD. As mentioned above, this algorithm is based on the specific characteristics of meshes and thereby lacks generality. In particular, the second strategy is only applicable on surfaces consisting of triangles (meshes) and because it is used in the first strategy, this implies that also the first strategy can be only applied on meshes efficiently.

3 Proposed Methods

In this section we propose a novel algorithm for calculating the exact Hausdorff distance. Before starting with the new algorithm, we will define some notations that will hold through the rest of the paper. We also present the straightforward algorithm for calculating the Hausdorff distance in Algorithm 1 to ease explanation. Let $A = \{x_1, x_2, \ldots, x_m\}$ and $B = \{y_1, y_2, \ldots, y_n\}$ be two point sets in $\mathbb{R}^k$ and let $||x, y||$ be any norm $\mathbb{R}^k \longrightarrow \mathbb{R}$ where $x, y \in \mathbb{R}^k$. In the usual case this is the euclidean distance function. Recall equations (1) and (2) and note that the Hausdorff distance is the maximum of the two directed Hausdorff distances in both directions. Thus, from now we will only concentrate on computing the directed Hausdorff distance $\check{H}(A, B)$.

Algorithm 1. NAIVEHDD Straightforwardly Computes the Directed Hausdorff Distance

Require: Two finite point sets A, B.
Ensure: Directed Hausdorff distance
1: $cmax \leftarrow 0$
2: **for** $x \in A$ **do**
3: $\quad cmin \leftarrow \infty$
4: $\quad$ **for** $y \in B$ **do**
5: $\quad\quad d \leftarrow ||x, y||$
6: $\quad\quad$ **if** $d < cmin$ **then**
7: $\quad\quad\quad cmin \leftarrow d$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **if** $cmin > cmax$ **then**
11: $\quad\quad cmax \leftarrow cmin$
12: $\quad$ **end if**
13: **end for**
14: **return** $cmax$

Note that we will only use two dimensional point sets in illustrations for simplicity, although the proposed algorithm is applicable for point sets in $\mathbb{R}^k$.

Obviously Algorithm 1 runs in $O(m * n)$ time where $m = |A|$ and $n = |B|$ because both loops in Algorithm 1, Lines 2 and 4, always run through all points. From now, we will call these loops the outer loop and the inner loop respectively.

Hereafter the three parts of the proposed algorithm are presented: in the first part (Section 3.1), we show that a complete scan in the inner loop is not always necessary (early breaking). The second part (Section 3.2) presents a sampling method that can replace the trivial scanning and considerably enhance the performance. The combination of early breaking and the sampling method provides a significant efficiency increase compared to the application of these optimizations individually. In the third part (Section 3.3), a refinement technique is presented that excludes the intersection of the compared sets from computation in advance in the case where the intersection is defined, e.g., when the compared sets are images or volumes, which additionally provides a small increase in the speed of the algorithm. Finally, in Section 3.4 we present the runtime analysis of the proposed algorithm.

3.1 Early Breaking

It is not always necessary that the scan in the inner loop (Algorithm 1, Line 4) runs completely through. Since the Hausdorff distance aims to find the maximum of the minimums, the inner loop can actually break as soon as a distance is found that is below the temporary HD ($cmax$), because in this case $cmax$ will definitely not change in the rest of the loop. This means the algorithm can break the

inner loop and continue with the next point of the outer loop. Through the rest of the paper, we will call stopping the inner loop because of finding some distance $d < cmax$ the early break. We modify Algorithm 1 to consider the early break as illustrated in Algorithm 2, Line 9.

Algorithm 2. EARLYBREAK Computes the Directed HDD Using the Early Break Technique and the Random Sampling

Require: Two finite point sets A, B
Ensure: Directed Hausdorff distance
1: $cmax \leftarrow 0$
2: $E \leftarrow A \backslash (A \cap B)$ {described in Section 3.3}
3: $E_r \leftarrow randomize(E)$ {Randomization described in Section 3.2}
4: $B_r \leftarrow randomize(B)$ {Randomization described in Section 3.2}
5: **for all** $x \in E_r$ **do**
6: $cmin \leftarrow \infty$
7: **for all** $y \in B_r$ **do**
8: $d \leftarrow ||x, y||$
9: **if** $d < cmax$ **then** {Early break described in Section 3.1}
10: break
11: **end if**
12: **if** $d < cmin$ **then**
13: $cmin \leftarrow d$
14: **end if**
15: **end for**
16: **if** $cmin > cmax$ **then**
17: $cmax \leftarrow cmin$
18: **end if**
19: **end for**
20: **return** $cmax$

Note that the run time of Algorithm 2 depends on at least the following factors:

- The order in which the outer loop iterates the points in A: detecting a point with a relatively large distance to B leads to a larger value of $cmax$ and consequently to a higher probability of the occurrence of an early break. In fact detecting the point with maximum distance to B at the beginning leads to the best case.
- The order in which the inner loop performs the scan in B: Here it is advantageous to pick points with smaller distances, because a distance below $cmax$ leads to an early break. Fig. 1 illustrates the relation between the iteration order and the occurrence of the early break.

3.2 Random Sampling in Place of Scanning

Now the question is how much is the improvement from using the early break alone? According to object coherence [9] based on the principle of spatial locality, in some classes of point sets, like images and volumes, the points are likely to be spatially distributed in a way that points iterated successively (e.g., line-wise or column-wise in an image) in the first set have similar distances to some reference point in the second set, which means that the early break could likely be delayed more than necessary.

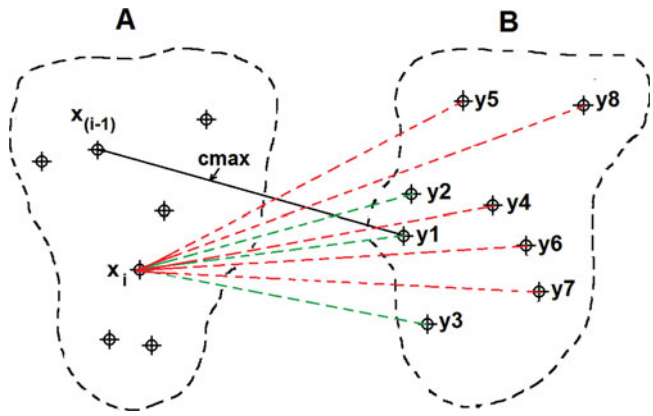

Fig. 1. $x_{(i-1)}$ is the point already minimized in the previous iteration where $cmax$ was found to be the current maximum (temporary HD). Point x_i is being currently minimized by calculating its distances to B. Points $y_1 \dots y_8 \in B$ are numbered according to their distance to x_i. An iteration order beginning with y_1, y_2 or y_3 is good because it will cause an immediate break whereas an iteration order beginning with other points is worse because the scan will continue.

In other words, if no early break occurs, it is likely that it will not occur when a nearby point is tried. It is better in this case to continue the search in another region which is spatially far from the current point. In this section we describe how to use random sampling instead of the trivial scanning to improve performance. This method leads to an algorithm with nearly-linear runtime as will be shown in Section 3.4

In random sampling, the aim is to avoid similar distances in successive iterations. This is achieved by randomly iterating the points in the inner loop. However, we randomize the sampling order also in the outer loop. We found that randomizing the sampling order additionally in the outer loop makes the runtime more efficient in some special cases. E.g., when the two point sets form generally linear shapes and one of them is nearly on the extension line of the other. The randomization additionally in the outer loop reduces the probability of worst cases with such point sets (this is e.g., frequent when the compared point sets are trajectories). For other cases, it is enough to only randomize the inner loop. But because the randomization doesn't need much computational effort and because there are no cases where it has a negative effect on the efficiency, we always randomize both of the loops. To achieve this, we prepare a list B_r with all points $y \in B$ randomly ordered and we use this set for iteration in the inner loop instead of set B. The same is done with iterating in the outer loop, more specifically the set E is also randomized. Algorithm 2, Lines 3 and 4 show the additional randomization steps

Note that preparing the random set in advance is necessary, because picking random candidates in the loop cannot ensure iterating through all the points. Generating the random order is possible in linear runtime by swapping each point in the set with a randomly selected point from the same set. Algorithm 3 illustrates this linear randomization.

The random scan eliminates the effect of the spatial locality in the point set and provides a significant improvement as shown in Sections 3.4 and 4.

Algorithm 3. RANDOMIZE Finds a Random Order of a Given Point Set

Require: A finite point set S
Ensure: Random order of S
  $S_r \leftarrow S$
  for all $p_1 \in S_r$ **do**
    $p_2 \leftarrow randompoint(S_r)$
    $swap(p_1, p_2)$
  end for
  return S_r

3.3 Excluding Intersection

In this section we describe a refinement that is applicable when the compared point sets are not disjunct but rather have a computable intersection. This is the case when the point sets are defined on a grid. For instance in the evaluation of medical volume segmentations, the compared images (test image and ground truth image) mostly have a large portion of voxels in common. We describe a technique that improves the performance of calculating the Hausdorff distance by excluding the intersection from the computation. This optimization generally provides a small increase in speed beyond the combination of early breaking and randomization. It is not a core part of the general HD algorithm, and can be used to achieve a small speed increase in cases where it is applicable. Let $S = A \cap B$ be the intersection between the compared point sets, then it is easy to conclude that $\check{H}(A, B) = \check{H}(A \backslash S, B)$. This follows from the fact that when iterating points $x \in A$ in the outer loop, $\forall x_i \in S \; \exists s_1 = x_i \in A, s_2 = x_i \in B : ||s_1, s_2|| = 0$, it follows that $dist(x_i, B) = 0$ which means that $cmax$ doesn't change in the corresponding iteration. In other words, for each of the intersection points, a direct early break is guaranteed and therefore it is not necessary to include them in the outer loop and they can be excluded from A in advance. Note that the intersection points must be however included in the inner loop because they could be at minimum distance to some point $y \in B, y \notin S$. Algorithm 2, Line 2 shows the additional step needed to implement this improvement.

3.4 Runtime Analysis

Algorithm 2 has a runtime of $O(m)$ in its best case and a run time of $O(m * n)$ in its worst case. The best case is when an early break occurs directly at the beginning of each iteration in the inner loop, that is when we always select a point with a distance below $cmax$. On the other hand, the worst case occurs when a full scan runs through completely in each iteration. The more important question is about the runtime of the average case.

Informally, it is expected that the average case runtime is biased towards the best case because the worst case generally requires conditions that are more difficult to satisfy. While a definite iteration order in the inner and the outer loops is required for the worst case so that the early break is prevented in each iteration, the best case requires only one condition, namely picking a point with a distance below $cmax$ in each first iteration in the inner loop.

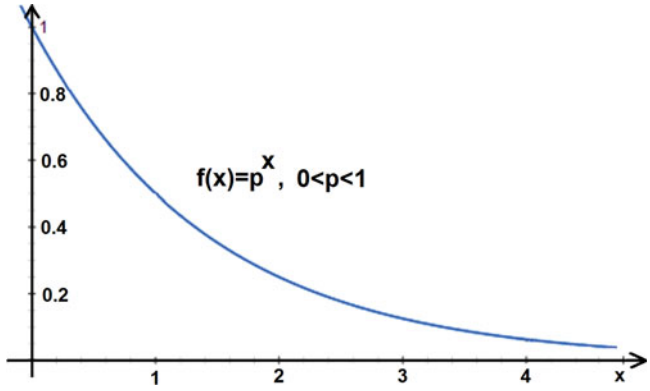

Fig. 2. The probability density function of a geometrical distribution.

Now let us see the average case runtime in a more formal way. We consider the randomly picked point $y \in B$ in Algorithm 2 in the inner loop and define the random variable D to be the distance d measured between the point y and the current reference point $x \in A$. We also define the event e to be that distance d is larger than $cmax$ that is $e \equiv d > cmax$. Note that event e means the non-appearance of an early break. Let us assume that event e always occurs with the probability q that is $P(e) = q$. Obviously the event $\overline{e}$ occurs with probability $p = 1 - q$ and denotes picking a distance $d \leq cmax$.

We also define the random variable R to be the number of successive distances exceeding $cmax$ followed by one distance below $cmax$ i.e., the length of a sequence of successive events e followed by an event $\overline{e}$. For any iteration i, this is equivalent to $i - 1$ distances from the reference point x to the points $y_1, y_2, \ldots, y_{i-1}$ namely $d_1, d_2, \ldots\ldots, d_{i-1} > cmax$ and one distance $d_i \leq cmax$. The probability density function of R is given by

$$\begin{aligned} f(x) &= P(d_1 > cmax, \ldots, d_{x-1} > cmax, d_x \leq cmax) \\ &= q * \ldots . * q * p \\ &= q^{x-1} p \end{aligned} \tag{3}$$

which is a geometrical probability distribution. Fig. 2 shows the probability distribution $f(x)$. Note the strong steepness of $f(x)$ that make longer runs of event e unlikely and intuitively explains the bias of the average case runtime towards the best case.

To formally find the average case runtime, the expected value $E[R]$ of $f(x)$ should be found which is equivalent to the expected number of iterations until an early break

$$E[R] = \sum_{x=1}^{\infty} x f(x) = \sum_{x=1}^{\infty} x q^{x-1} p. \tag{4}$$

$E[R]$ is a geometrical series with $0 \leq p \leq 1$ that converges and has a sum. From Equation (4) it follows:

$$E[R] = p + 2.q.p + 3.q^2.p + 4.q^3.p + \cdots \tag{5}$$

By multiplying both sides with q, subtracting the resulting equation from Equation (5), and then dividing by p

$$\frac{E[R](1-q)}{p} = 1 + q + q^2 + q^3 + \cdots \tag{6}$$

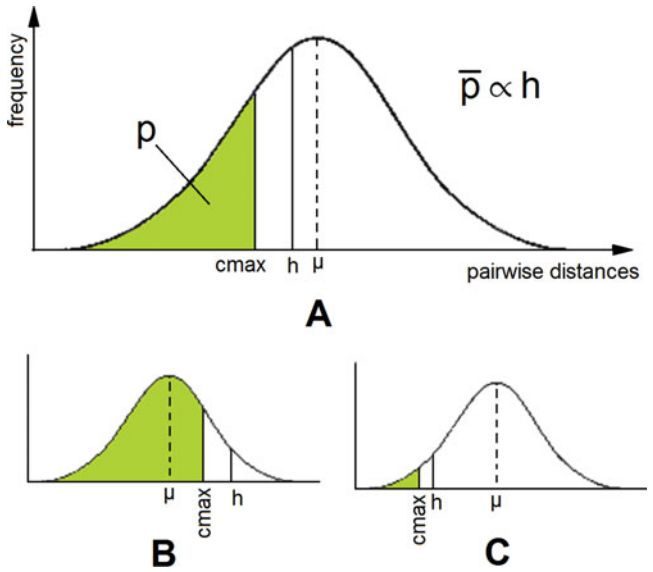

Fig. 3. Distribution of pairwise distances assuming a normal distribution for illustration. (A) Position of the Hausdorff distance h relative to the distribution affects p because $cmax \leq h$. (B) h is large and $cmax$ can reach large values thereby increasing p. (C) h is small and $cmax$ remains small thereby decreasing p.

By multiplying both sides with q, subtracting the resulting equation from Equation (6), and then substituting $q = 1 - p$

$$E[R] = \frac{1}{p}. \tag{7}$$

Equation (7) tells the important fact that the number of tries until an early break depends only on p which denotes the probability of picking a point with distance below $cmax$. The higher p is, the more likely that the inner loop terminates after a lower number of tries and vice versa.

But how high is p actually and what does it depend on? In fact, p depends mainly on how large $cmax$ is and $cmax$ is limited by the HD because ($cmax \leq h$) which means the HD determines how large $cmax$ at most can be. If the Hausdorff distance is large, $cmax$ can take larger values and thus it is more likely that a randomly selected point is below $cmax$ which means a higher value of p and vice versa, that is $p \propto h$. Fig. 3 illustrates the relation between $cmax$, the probability p, the Hausdorff distance, and the distribution of the pairwise distances d_{ij} where $d_{ij} = ||x_i, y_j|| : \forall x_i \in A, y_j \in B$. Here, a normal distribution is just for illustration and the relation holds for any distribution. The example is to show how p does not directly depend on the size of set B, but rather on the Hausdorff distance h and the distribution of the pairwise distances.

Note that we don't have to determine the distribution of the pairwise distances to conclude that the runtime depends only on h, this is because the distribution only determines the value of p, which is irrelevant for whether the runtime is dependent on B or not because the expected value $\frac{1}{p}$ is a constant value for any $p > 0$.

Formally, the average probability that the randomly picked distance $d \leq cmax$ is given by

$$\overline{p} = average\left(\int_{x=0}^{cmax} f(x)dx\right) = c\int_{x=0}^{h} f(x)dx, \tag{8}$$

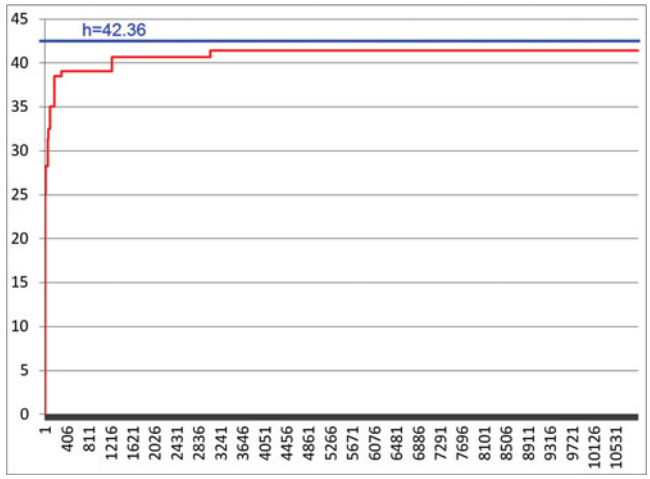

Fig. 4. Progress of $cmax$ in the first 10 thousand iterations (outer loop) when comparing two real brain tumor segmentations. Note that only 10 thousand of 15.6 million iterations in total are shown and thus the curve does not reach the HD.

where f is the probability density function that represents the distribution of the pairwise distances and c is a constant that results by estimating $cmax$ in terms of h; the justification of this estimation is in the next section.

3.5 Convergence of the Temporary HD ($cmax$)

The value of $cmax$ geometrically increases during the progress of the outer loop (Algorithm 2) so that it already reaches values near h after a very small number of iterations compared with the total count of iterations, as demonstrated in Fig. 4.

We explain this geometrical increase as follows: At the beginning of the outer loop $cmax$ is zero, then it increases monotonically with the progress of the outer loop until it reaches the Hausdorff distance h. In each iteration, there are two possibilities, either the distance to the current point is smaller than $cmax$, here no $cmax$ update is performed or the distance is larger than $cmax$, in this case $cmax$ is updated to have the distance value. Let us observe only those iterations where $cmax$ is updated. For any such update iteration i we define $cmax(i)$ to be the $cmax$ value in that iteration (before update) and $d(i)$ to be the distance of the randomly selected point. The possible values that $d(i)$ can have are in the interval $[cmax(i), h]$. This means $d(i)$ has an expected value of $\frac{h-cmax(i)}{2}$ which is subsequently the expected value of $cmax(i+1)$. It follows that in the next iteration $i+1$, the expected interval in which $d(i+1)$ values can be is $[h - \frac{h-cmax(i)}{2}, h]$. Analogously, for iteration $i+2$, we likely get $cmax(i+2) = \frac{h-cmax(i)}{4}$ and an interval $[h - \frac{h-cmax(i)}{4}, h]$ and so on which implies a geometrical convergence of $cmax$ to h. To experimentally verify this geometrical convergence, we computed the Hausdorff distance between 1,000 pairs of trajectories generated from the road network of Oldenburg (described in Section 4.7). Each of the trajectories consists of 2,000 points. For each iteration in the outer loop, two values were recorded, namely the number of iterations until the early break n and the value of $cmax$ at the beginning of each iteration in the outer loop, hence getting 2,000 values for each pair of trajectories, i.e., 2 million

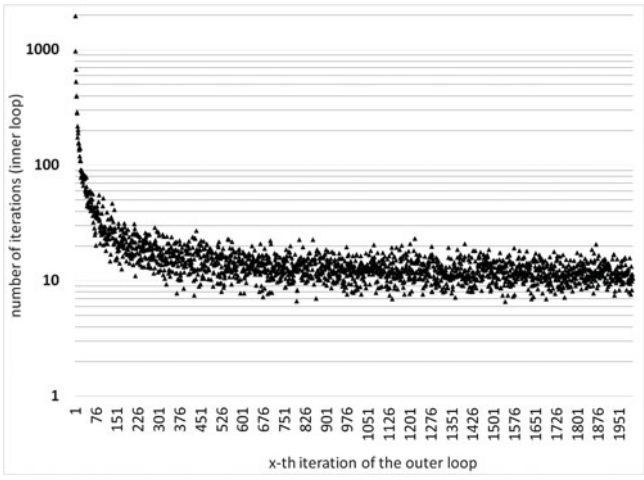

Fig. 5. The average number of iterations in the inner loop until the early break at each iteration of the outer loop. Values are recorded from measuring the HD between 1,000 pairs of trajectories generated from the road network of Oldenburg. Each trajectory contains 2,000 points. Iterations of the outer loop are on the *x*-axis and the number of iterations in the inner loop averaged over all pairs on the *y*-axis.

values in total. Two statistics were computed, the first by averaging the number of iterations until the early break (n) to get ($\overline{n}$) at each iteration of the outer loop. This is visualized with a logarithmic scale in Fig. 5. At first, $\overline{n}$ is very high because $cmax$ is zero and the inner loop is scanned completely. After that, $\overline{n}$ decreases rapidly to converge finally at very low values. This statistic confirms the convergence behavior of the number of iterations until the early break predicted theoretically. The second statistic was made by converting the recorded $cmax$ values to percentage values of the HD between the corresponding pair of trajectories; this is because the HD is different in each pair. From these percentage values, we counted how many values exceed 90 and 99 percent at each iteration in the outer loop. Fig. 6 shows the results. We show only the first 200 iterations to make the plot more readable. The results confirm the quick convergence of $cmax$ to the HD. In about 80 percent of the cases, $cmax$ is already after five iterations above 90 percent of the HD, and already after 50 iterations above 99 percent of the HD.

From Equations (7) and (8), the expected number of iterations until the early break given a Hausdorff distance h is

$$E[R] = \frac{1}{\overline{p}} = \frac{1}{c\int_{x=0}^{h} f(x)dx}. \quad (9)$$

Note that if h is very small, for example $h \approx 0$, the algorithm tends to get low performance. Nevertheless, low values of h mean high match between the point sets which means that it is likely that A and B have high intersection, which also means that the improvement introduced in Section 3.3 will compensate the loss of performance by excluding the intersection and this will keep a low overall runtime.

3.6 Handling of Outliers

The Hausdorff distance is generally sensitive to outliers [7], [12]. The Hausdorff quantile is a method proposed in [12] to solve the problem of outliers: according to the Hausdorff quantile method, the Hausdorff distance is defined to be the

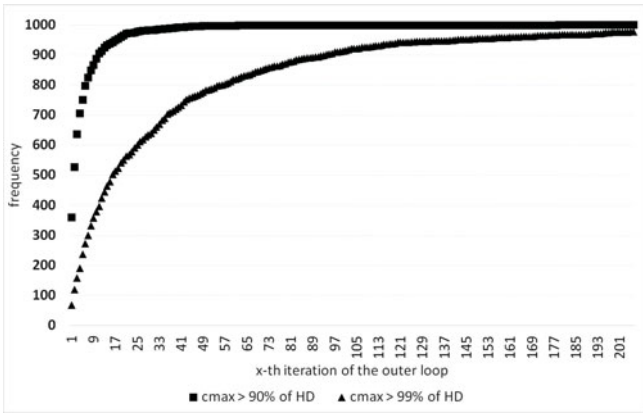

Fig. 6. Convergence behaviour of the temporary HD ($cmax$) along the iterations of the outer loop. Iterations of the outer loop are on the *x*-axis and the frequencies of $cmax$ values exceeding 90 and 99 percent of the corresponding HD at each iteration are on the *y*-axis.

qth quantile of distances instead of the maximum, so that possible outliers are excluded, where q is selected depending on the application and the nature of the measured point sets. The proposed algorithm can be easily extended to support the Hausdorff quantile by saving all distances measured and after the outer loop is finished, the distances are sorted and the qth quantile is returned instead of $cmax$.

4 Experiments

The proposed algorithm was tested with three different types of data, namely real brain tumor segmentations (MRI 3D volumes), trajectories generated from a road network and random 3D Gaussians.

Testing with real brain tumor segmentations is done in four different variants against the ITK HD algorithm and in a fifth variant against a version of the proposed HD algorithm without the random sampling. In the first experiment (Section 4.1), the HD between the volumes and the corresponding ground truth segmentations was calculated. In the second experiment (Section 4.2), images were compared with randomly selected volumes from the same set, so that the volumes in each pair do not overlap to rule out that the general performance is dependent on the overlap between the compared images. In the third experiment (Section 4.3), new images were generated by merging up to eight images in order to test the performance when the point set size increases. In the fourth experiment (Section 4.4), the volumes were increased both in point set size and grid size to test the sensitivity to grid size. In the fifth experiment (Section 4.5), the same test as in the first experiment was performed, but against the proposed algorithm without the random sampling step to show the effect of combining the early break with the random sampling.

In the sixth experiment (Section 4.6), 3D point sets were generated based on random Gaussians and used to test the proposed algorithm to rule out that the efficiency of the proposed algorithm is not dependent on the nature of medical images.

Finally, in the last experiment (Section 4.7), trajectories generated from a road network were used to test the proposed algorithm against the incremental Hausdorff distance calculation algorithm (INC), based on R-Trees.

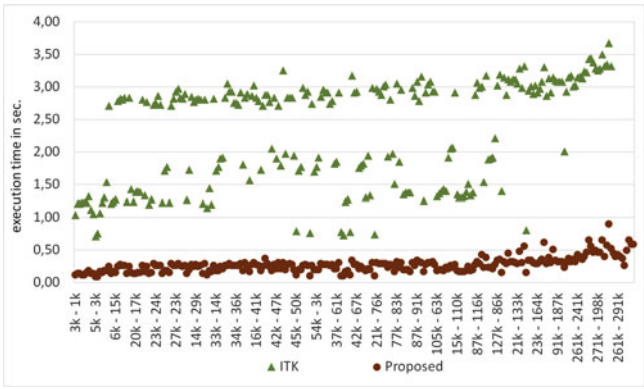

Fig. 7. Comparison between the performance of the proposed algorithm and the ITK algorithm in validating 240 real brain tumor segmentations against the corresponding ground truth. The set size in kilo voxels is on the horizontal axis and the run time in seconds is on the vertical axis. The grid size varies from $125 \times 125 \times 125$ to $250 \times 250 \times 250$ voxels.

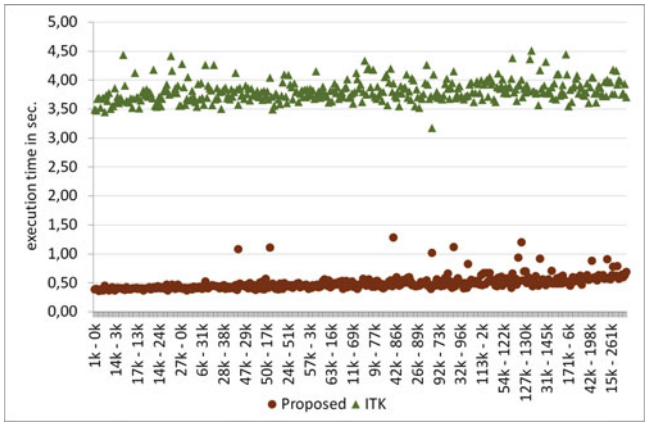

Fig. 8. Comparison between the performance of the proposed algorithm and the ITK algorithm in comparing 300 pairs of volumes selected randomly so that the overlap between volumes in each pair is zero. The set size in voxels is on the horizontal axis and the runtime in seconds on the vertical axis. All volumes have a unified grid size of $250 \times 250 \times 250$ voxels.

The first six experiments were performed on a machine with 3 GHz Intel core processor, 8 GB Memory, and Windows 7 OS. The last experiment (Section 4.7) was done on a machine with the specification described in [21].

4.1 Comparing Volumes with Ground Truth

In this experiment, we used a test set of 300 automatic brain tumor segmentations (MRI 3D volumes) from the BRATS2012 challenge.[2] These volumes were produced by segmentation algorithms proposed by four participants of the BRATS challenge. The volumes vary widely in size and span the range from 2 k to 600 k voxels as point set size and from $125 \times 125 \times 125$ to $250 \times 250 \times 250$ voxels as grid size. Each of these volumes was validated against the corresponding ground truth segmentation made by human experts. The test set consists of 240 volumes and 60 ground truth segmentations. All volumes were validated using three algorithms: the first is an implementation of the straightforward algorithm (Algorithm 1) to ensure that the proposed algorithm computes the correct Hausdorff distance. The second one is the standard Hausdorff distance algorithm of the ITK library,[3] namely the itk::HausdorffDistanceImageFilter, assumed to represent the state-of-the-art. The ITK algorithm is based on the distance transform technique and is described in [25] and [7]. The third algorithm is the proposed algorithm, an implementation of Algorithm 2.

Fig. 7 shows the performance of the proposed algorithm compared with the ITK algorithm: while the ITK algorithm took an average time of 2.09 seconds per volume, all runtimes of the proposed algorithm were below one second and have an average of 0.26 seconds per volume, which means that the proposed algorithm outperforms the ITK algorithm by about 7.6 times.

4.2 Testing with Non-Overlapping Images

The aim of this experiment is to rule out that the performance depends on the overlap between the two compared volumes. To this end, we put all images (segmentations and ground truth volumes) in one pool of 300 images, then 300 pairs were selected from the pool so that the intersection (overlap) between the two images in each pair is zero. This was possible because brain tumors reside in different locations in the brain. The HD was calculated using the ITK algorithm and the proposed algorithm. Note that we had to unify all volumes to one grid size, namely $250 \times 250 \times 250$ because the algorithms accept only pairs consisting of two volumes with the same grid size. Fig. 8 shows the runtime plot of the proposed algorithm compared with that of ITK: again the proposed algorithm outperforms the ITK algorithm about by 7.8 times. While the runtimes of the proposed algorithm rarely exceed one second and have an average of 0.51 sec, the ITK algorithm took an average of 3.82 sec. The result shows that the efficiency of the method is not restricted to overlapped point sets and thus confirms the runtime analysis in Section 3.4, namely Equation (9) that shows that the algorithm tends to have a high efficiency when the HD is large. The increase in the efficiency compensates the efficiency lost when the intersection is not present. This is the case in this experiment because the HD is likely large, given that the images don't overlap.

4.3 Testing with Large Volumes

In this experiment we test the runtime behavior when the set size increases. For this, we constructed a new pool of 300 volumes, where each of them is generated by merging up to 8 randomly selected volumes from the original test set without increasing the grid size, which is still $250 \times 250 \times 250$ voxels, that is $V = V_1 \cup V_2 \cup V_3 \ldots$ where $V_1, V_2, \ldots$ are the randomly selected volumes and V is the resulting test volume. The resulting volumes span a set size range from 150 k to 850 k voxels. Finally, 300 pairs were randomly selected and compared.

Fig. 9 shows that the proposed algorithm outperforms the ITK algorithm and has no significant runtime increase with increasing the set size.

4.4 Increasing the Grid Size

This experiment tests how the runtime and the needed memory behave when the volume grid size is increased.

2. MICCAI 2012 Challenge on Multimodal Brain Tumor Segmentation, www2.imm.dtu.dk/projects/BRATS2012

3. National Library of Medicine Insight Segmentation and Registration Toolkit www.itk.org

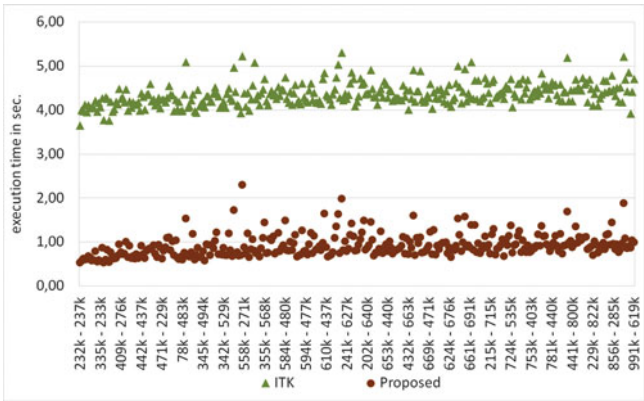

Fig. 9. Performance comparison between the proposed algorithm and the ITK algorithm in comparing enlarged volumes. The set size in kilo voxels is on the horizontal axis and the runtime in seconds on the vertical axis. All volumes have a unified grid size of 250 ×250 × 250 voxels.

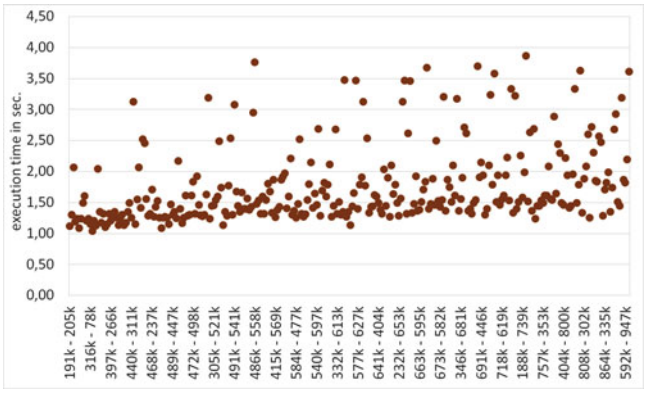

Fig. 10. The performance of the proposed algorithm in comparing volumes with grid size increased to 350 × 350 × 350 voxels. No runtime plot is shown for the ITK algorithm because it failed in all cases with a memory allocation error.

The aim is to simulate real cases with larger grids such as whole body volumes [17]. To this end, we increased both the grid size and the set size. This was done by multiple copying of randomly selected volumes into larger grids: instead of 250 × 250 × 250, we used 350 × 350 × 350 voxels which means a total number of 43 Million voxels including background. The resulting volumes span a set size range from 200 k to 950 k voxels. None of the pairs was compared successfully by the ITK algorithm, because in each pair the ITK algorithm broke down with a memory allocation error. On the contrary, the proposed algorithm computed all pairs successfully in an average time of 1.7 seconds as shown in Fig. 10.

The result of this experiment can be explained by the fact that distance transform based algorithms are sensitive to increasing grid size because all the background voxels should be labeled. On the contrary, the proposed algorithm is not sensitive to grid size increase because the background is not involved in the computation at all.

The results of all experiments with MRI segmentations against the ITK HD algorithm are summarized in Table 1.

4.5 Testing the Effect of Random Sampling

This experiment is to show the contribution of the random sampling to the efficiency of the proposed algorithm. The same data and configuration of the experiment in Section 4.1 is used except that the random sampling is replaced by direct scanning. In particular, Lines 3 and 4 in Algorithm 2 are omitted and the sets E and B are used instead of the sets E_r and B_r respectively. The results in Fig. 11 show that the random sampling is strongly related with the performance of the algorithm and has a significant contribution to the efficiency. Note that 8 instances are removed to improve the visibility of the plot because they have an execution time exceeding 100 seconds with direct scanning. The contribution of the random sampling is a factor of 36.8 measured as the ratio between the two execution times averaged over all pairs.

4.6 Testing with Random Gaussians

To rule out that the efficiency is dependent on the point distribution of medical volumes, the proposed algorithm was tested against random Gaussians. 300 point clouds were generated; each of them consisting of 50 thousand to 0.5 million points; the points in each cloud are normally distributed and satisfy a random Gaussian (i.e., the point coordinates x, y and z are generated according to three different Gaussians each with a random μ and a random σ) selected so that the points fit in a grid of 250 × 250 × 250 voxels. From these point clouds, 300 pairs were randomly selected. The HD distance between the point sets in each pair was measured by the proposed algorithm and the ITK algorithm. The results in Fig. 12 show that the proposed algorithm still outperforms the ITK algorithm with a factor of about 4.35. The experiment shows that the proposed algorithm replicates its performance with normally distributed point sets.

We analysed the few data points in Fig. 12 where the proposed algorithm required a computation time of more than 4 seconds. We found that the relatively long runtime is not related to a particular point set, but rather to a combination between two particular point set configurations. This observation conforms to the runtime analysis in Section 3.4, i.e., that the runtime is dependent on the value of the HD

TABLE 1
Result Summary for Experiments on Medical Images of Varying Sizes and Characteristics Where n1...n2 Is the Size Range of the Compared Point Sets, L, B, H Are the Grid Dimensions for Medical Volumes and the Time Values Are the Average Execution Time for Calculating the HD

Testing with ..	L × B × H (grid size)	n1..n2 (set size)	proposed algorithm	ITK algorithm
ground truth	125 to 250	2 k..350 k	0.26 sec.	2.09 sec.
non-overlapping point sets	250 × 250 × 250	2 k..350 k	0.51 sec.	3.82 sec.
merged volumes	250 × 250 × 250	207 k..1,100 k	0.93 sec.	3.45 sec.
increased grid size	350 × 350 × 350	280 k..1,470 k	1.70 sec.	allocation Error

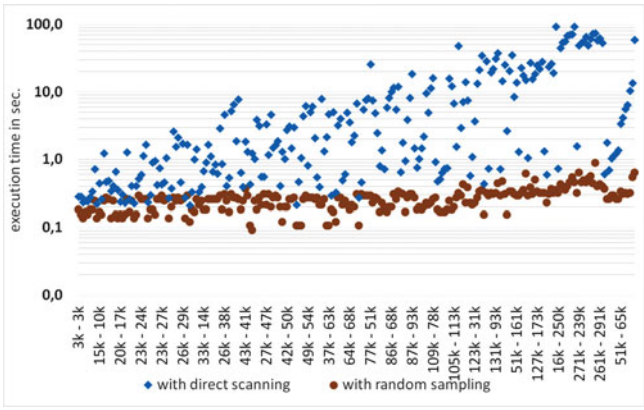

Fig. 11. Contribution of random sampling: Comparison between the efficiency of the proposed algorithm when using random sampling and direct scanning. The same data as in the experiment in Section 4.1 is used. The size of the compared images in kilo voxel is on the *x*-axis and the execution time in seconds, scaled logarithmically, is on the *y*-axis.

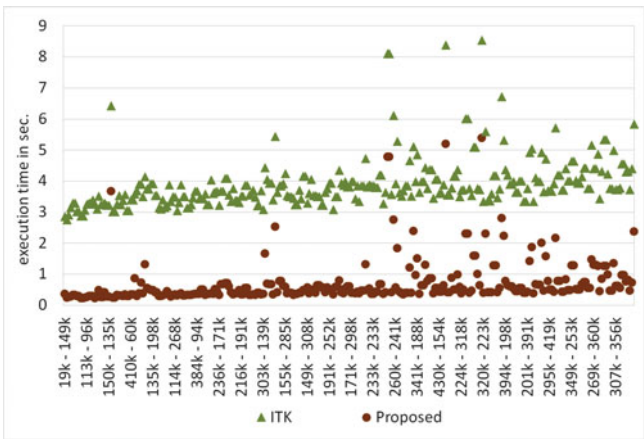

Fig. 12. Comparison between the performance of the proposed algorithm and the ITK algorithm in measuring the HD of 300 pairs of Gaussians generated by randomly selected means and standard deviations for each of the three dimensions. The size of the compared point sets in kilo voxels is on the *x*-axis and the execution time in seconds is on the *y*-axis.

relative to the distribution of the pairwise distances between the compared point sets, as illustrated in Fig. 3.

4.7 Testing Against Incremental Hausdorff Distance

In this experiment, the proposed method was tested against the incremental Hausdorff distance calculation algorithm (INC) proposed by Nutanong et al. [21]. To this end, we tested the proposed algorithm with the same data, the same setting, and on hardware of identical specification as described in [21], Section 7.1 (Hausdorff Distance Calculation). As point sets, we used trajectories generated from the Oldenburg (OL) road network[4] so that each trajectory is the shortest path between two randomly selected points in the network with a length of 2,000 units. The points on each trajectory were sampled in different resolutions, i.e., the path was truncated into chunks with different lengths. Five

4. City of Oldenburg Road Network http://www.cs.fsu.edu/ lifeifei/SpatialDataset.htm

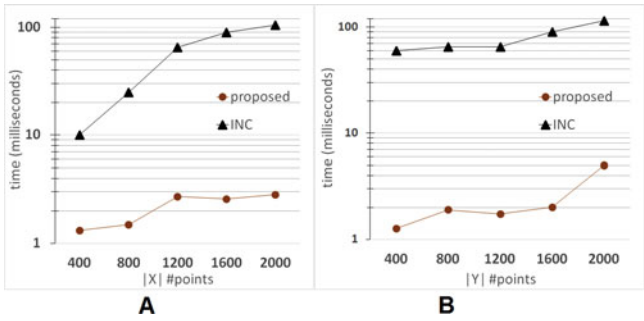

Fig. 13. Comparison of the execution time of calculating the Hausdorff distance HD(X,Y) by the proposed algorithm and the incremental Hausdorff calculation (INC) [21]. In A, the size of X is fixed and the size of Y varies and conversely in B. Each data point is the average of 200 pairs of trajectories. The size of the point set is on *x*-axis and the execution time in milliseconds on the *y*-axis.

groups of trajectories (G1 .. G5) were constructed so that each group contains trajectories sampled in a different resolution. G1, G2, G3, G4, G5 have 400, 800, 1,200, 1,600, 2,000 sampled points respectively. The HD(X,Y) was calculated between trajectories by varying the point set size, i.e., selecting trajectories from different groups. In a first experiment set, the size of X was fixed and the size of Y was varied, and in a second experiment set the size of Y was fixed and the size of X was varied. The execution times of these experiments are compared with the execution times published in [21], Section 7.1, Fig. 8. Fig. 13 shows the execution time where each data point is the average of 200 different pairs of trajectories. The results show that the proposed algorithm outperforms the INC algorithm by about 30 times.

5 Conclusion

We propose an efficient algorithm for computing the exact Hausdorff distance. We formally show that the proposed algorithm has a nearly-linear runtime in the average case. The proposed algorithm combines early breaking and randomization optimizations to achieve a significant increase in speed over other algorithms that do not use this combination. The proposed algorithm does not impose any restrictions on the input data, and is hence generalizable to all applications. Moreover, it does not require a complex setup phase needing high computational effort and extensive storage space.

We experimentally show a 36-fold increase in speed over an HD algorithm with only early breaking included i.e., without using the randomization. We also show experimentally that the proposed algorithm significantly outperforms in terms of speed the standard HD algorithm of the ITK Library in comparing medical volumes and the incremental HD algorithm in comparing trajectories generated from a road network. Moreover, the proposed algorithm is shown to work even when comparing volumes with extremely high dimensions (grid size). An implementation of the proposed algorithm is available as part of the EvaluateSegmentation software.[5]

5. EvaluateSegmentation is an open source project for evaluating medical volume segmentations available for download from http://github/codalab/EvaluateSegmentation

ACKNOWLEDGMENTS

The authors would like to thank Bjoern Menze, Computer Aided Medical Procedures, Technical University of Munich for providing the MRI brain segmentations from MICCAI 12 BRATS challenge to be used as test data. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement 318068 (VISCERAL). Abdel Aziz Taha is the corresponding author.

REFERENCES

[1] H. Alt, B. Behrends, and B. Johannes, "Approximate matching of polygonal shapes (extended abstract)," in *Proc. 7th Annu. Symp. Comput. Geometry*, 1991, pp. 186–193.
[2] M. J. Atallah, "A linear time algorithm for the Hausdorff distance between convex polygons," *Inf. Process. Lett.*, vol. 17, no. 4, pp. 207–209, 1983.
[3] K. O. Babalola, B. Patenaude, P. Aljabar, J. Schnabel, D. Kennedy, W. Crum, S. Smith, T. F. Cootes, M. Jenkinson, and D. Rueckert, "Comparison and evaluation of segmentation techniques for subcortical structures in brain MRI," *Med. Image Comput. Comput.-Assisted Intervention*, vol. 11, no. Pt 1, pp. 409–416, 2008.
[4] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
[5] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Comput. Vis. Image Understanding*, vol. 89, no. 2, pp. 114–141, 2003.
[6] K. Ciesielski, X. Chen, J. Udupa, and G. Grevera, "Linear time algorithms for exact distance transform," *J. Math. Imaging Vis.*, vol. 39, no. 3, pp. 193–209, 2011.
[7] B. Eric, F. Andriy, and C. Nikos, "The use of robust local Hausdorff distances in accuracy assessment for image alignment of brain MRI," *Insight J.*, pp. 1–14, May 2008.
[8] G. Gerig, M. Jomier, and M. Chakos, "Valmet: A new validation tool for assessing and improving 3D object segmentation," in *Proc. 4th Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2001, pp. 516–523.
[9] E. Groeller and W. Purgathofer, "Coherence in computer graphics," in *Proc. Vis. Intell. Design Eng. Archit.*, 1993, vol. 5, pp. 61–76.
[10] M. Guthe, P. Borodin, and R. Klein, "Fast and accurate Hausdorff distance calculation between meshes," *J. WSCG*, vol. 13, no. 2, pp. 41–48, 2005.
[11] M. Hossain, M. Dewan, K. Ahn, and O. Chae, "A linear time algorithm of computing Hausdorff distance for content-based image analysis," *Circuits, Syst. Signal Process.*, vol. 14, pp. 389–399, 2012.
[12] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 9, pp. 850–863, Sep. 1993.
[13] P. Indyk and S. Venkatasubramanian, "Approximate congruence in nearly linear time," *Comput. Geometry Theory Appl.*, vol. 24, no. 2, pp. 115–128, 2003.
[14] J. Bing and B. C. Vemuri, "Robust point set registration using gaussian mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 8, pp. 1633–1645, Aug. 2011.
[15] H. Khotanlou, O. Colliot, J. Atif, and I. Bloch, "3D brain tumor segmentation in MRI using fuzzy classification, symmetry analysis and spatially constrained deformable models," *Fuzzy Sets Syst.* vol. 14, no. 10, pp. 1457–1473, May 2009.
[16] H. Kim, S. Park, S. Lo, J. Monroe, and J. Sohn, "Bidirectional local distance measure for comparing segmentations," *Med. Phys.* vol. 14, pp. 6779–6790, 2012.
[17] G. Langs, H. Mueller, B. H. Menze, and A. Hanbury, "Visceral: Towards large data in medical imaging-Challenges and directions," in *Proc. MCBR-CDS MICCAI Workshop*, 2013, vol. 14, pp. 92–98.
[18] Jr., C. R. Maurer, R. Qi, and V. Raghavan, "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Trans. Pattern Anal. Mach. Intell.* vol. 14, no. 2, pp. 265–270, Feb. 2003.
[19] F. Morain-Nicolier, S. Lebonvallet, E. Baudrier, and S. Ruan, "Hausdorff distance based 3D quantification of brain tumor evolution from MRI images," in *Proc. 29th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2007, pp. 5597–600.
[20] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.* vol. 14, no. 12, pp. 2262–2275, Dec. 2010.
[21] S. Nutanong, E. H. Jacox, and H. Samet, "An incremental hausdorff distance calculation algorithm," *Proc. VLDB Endowment*, vol. 14, no. 8, pp. 506–517, May 2011.
[22] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *ACM Trans. Database Syst.* vol. 14, no. 2, pp. 529–576, 2005.
[23] D. B. Russakoff, C. Tomasi, T. Rohlfing, and C. R. Maurer Jr., "Image similarity using mutual information of regions," in *Proc. 8th Eur. Conf. Comput. Vis.*, 2004, pp. 596–607.
[24] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex wavelet structural similarity: A new image similarity index," *IEEE Trans. Image Process.*, vol. 14, no. 11, pp. 2385–2401, Nov. 2009.
[25] N. J. Tustison, M. Siqueira, and J. C. Gee, "N-D linear time exact signed Euclidean distance transform," *Insight J.*, pp. 1–5, 2006.

Taha Abdel Aziz received the BSc degree in computer science in 2004 and the MSc degree in computer science in 2007, all from Vienna University of Technology, Vienna, Austria. He is currently working toward the PhD degree in Information & Software Engineering Group of the Vienna University of Technology, where he is a research assistant in the field of axiometric and information retrieval evaluation. His main research interests are the evaluation and analysis of information retrieval systems, especially in medical imaging and medical retrieval systems.

Hanbury Allan received the BSc degree in physics and applied mathematics in 1994, the BSc (Honors) degree in physics in 1995, and the MSc degree in physics in 1999, all from the University of Cape Town, South Africa. He was awarded the PhD degree in mathematical morphology in 2002 from Mines ParisTech, France, and the Habilitation in Practical Informatics from the Vienna University of Technology, Austria in 2008. He is a senior researcher at the Vienna University of Technology, Austria. He is scientific coordinator of the EU-funded KHRESMOI Integrated Project on medical and health information search and analysis, coordinator of the EU-funded VISCERAL project on evaluation of algorithms on big data, and coordinator of the CHIST-ERA project MUCKE on search in multimodal data and social networks. His research interests include information retrieval, multimodal information retrieval, and the evaluation of information retrieval systems and algorithms. He is author or coauthor of more than 100 publications in refereed journals and international conferences.