Pattern Recognition 117 (2021) 107989

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

An efficient and locality-oriented Hausdorff distance algorithm: Proposal and analysis of paradigms and implementations☆

Érick Oliveira Rodrigues

Department of Academic Informatics (DAINF), Universidade Tecnologica Federal do Parana (UTFPR), Pato Branco, State of Parana, Brazil

ARTICLE INFO

Article history:
Received 6 April 2020
Revised 24 January 2021
Accepted 5 April 2021
Available online 20 April 2021

Keywords:
Hausdorff distance
Mathematical morphology
Similarity
Registration
Parallelism

ABSTRACT

Hausdorff distance (HD) is a popular similarity metric used in the comparison of images or 3D volumes. Although popular, its main weakness is computing power consumption, being one of the slowest set distances. In this work, a novel, parallel and locality-oriented Hausdorff distance implementation is proposed. Novel as it is the first time in the literature that an actual algorithmic implementation using morphological dilations is proposed and thoroughly evaluated. Parallel, as it is more robust in terms of parallelization than the state-of-the-art algorithm and local as it has an intrinsic sensitivity to voxels that are closer in space. This proposal can be faster than the state-of-the-art in several practical cases such as in medical imaging registrations (up to 8 times faster on average in one of the CPU experiments) and is faster in the worst-case (up to 22337 times faster in one of the CPU experiments). Worst-case scenarios and high resolution volumes also favor the proposed approach. Throughout the work, several sequential and parallel CPU and GPU implementations are evaluated and compared.

1. Introduction

Hausdorff distance (HD) is a set distance metric [1] popularly used to compute the similarity of binary images [2] or binary 3D volumes [3]. The directed version of the metric computes the maximum extent to which any point of a set lies to the nearest point of the other set. However, the Hausdorff distance does not rely on the identification of landmarks and their alignment. Due to that fact, this distance is far less tolerant to noise. Moreover, it favors proximity to the detriment of superposition. When the images or objects are equal and exactly superposed, the Hausdorff distance is equal to 0.

HD is widely used in the computation of image similarity [2], image or volume registration [4,5] and template-matching [6,7]. Other applications of HD include face recognition problems [8], object matching [9], classification [10], clustering techniques [11], diagnosis using image similarity [12], quantification of brain tumor [13], and many other cases.

Image registration can be described as a specific template matching problem where the template image can be distorted or transformed. In this sense, image registration is substantially more expensive than template matching per se, as the similarity must be recalculated several times. However, image registration is still paramount in medical imaging [7] to mitigate variations introduced by time series acquisitions [14].

Serra [15] mathematically proved that it is possible to compute the Hausdorff distance using image dilations from mathematical morphology. However, different morphological implementations can achieve distinct performances. Distinct morphological operators and approaches can be used to combine the images after the dilations. Approaches can use the mathematical definition of sets, they can use the image itself, they can be parallel or sequentially oriented. These facts lead to many uncertainties in run time performances.

In addition, Serra [15] does not clearly describe how structuring elements are related to different distance norms and does not provide any algorithm/implementation nor efficiency analysis. His work focuses on the mathematical aspect. No morphological implementation has ever been evaluated in the literature. The current state-of-the-art algorithm was proposed by Taha and Hanbury [16] and it does not use morphology nor it is outrightly amenable to parallelization. However, their algorithm was faster than the ITK medical framework, and this was their key performance claim when the article was published.

This work proposes a morphological implementation for the Hausdorff distance and also describes an extensive performance analysis. The proposed implementation is parallel-oriented and is faster than Taha-Hanbury (state-of-the-art) for high resolution data and high density structures that do not frequently intersect. It is plainly impossible to compute the Hausdorff distance of large

☆ Fully documented templates are available in the elsarticle package on CTAN.

E-mail addresses: erickrodrigues@utfpr.edu.br, erickr@id.uff.br

https://doi.org/10.1016/j.patcog.2021.107989
0031-3203/

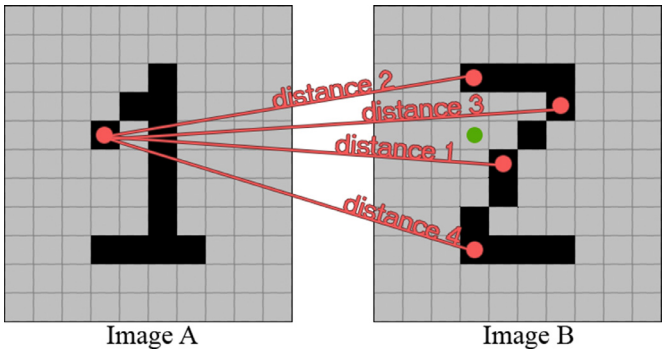


Fig. 1. Partial illustration of the Hausdorff distance computation between images A and B.

volumes using the Taha-Hanbury approach as it requires so much time to process. In one of the experiments herein (8192x8192 images), the Taha-Hanbury algorithm requires 2 days while the proposed morphological approach requires 0.6 seconds using the GPU. The improvement is very expressive. Moreover, substantial improvements also occur in the CPU.

The major contributions of this work are: (1) proposing a novel implementation of the HD using morphological techniques and (2) proposing a locality-sensitive parallel-driven approach. Minor contributions include: (1) implementing and investigating the performance of the Taha-Hanbury algorithm in the GPU and (2) proposing a faster implementation in the worst case **regardless** of the processing paradigm (CPU or GPU).

2. Literature review

Given two finite non-empty sets $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_m\}$, their Hausdorff distance is informally defined as the maximum distance any element of one of the sets is required to travel to the other set as fast as possible. The mathematical formulation is given by $H(A, B)$ in Eq. (1):

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

where,

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

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

and d stands for the distance between the points of the sets (e.g., Euclidean distance, Chebyshev distance, etc). Functions $h(A, B)$ and $h(B, A)$ are respectively called directed forward and backward Hausdorff distances. When $h(A, B) = s$, the distances from all points in A to B do not exceed s. In other words, all points in A are at distance s or less from the nearest point in B [17].

Let us suppose that we would like to compute the Hausdorff distance of images A and B shown in Fig. 1. At first, a distance has to be defined as input parameter. In this example, the sup norm (or Chebyshev distance) is chosen. Next, the distances from all black pixels in A to all black pixels in B are computed, and the minimal distance is stored.

Fig. 1 illustrates an occasion where the distance of a black pixel in image A is being calculated. For this pixel, the minimal distance is selected, which is distance 1 (sup metric). The red lines, or distances, are traced for all pixels in B (just a few of these distances are highlighted in the figure). The green circle represents the pixel in image A whose distances are being computed. In other words, the red lines actually represent the distances from the green circle to the corresponding black pixel in B.

This entire process is repeated for every pixel in image A. After obtaining all the distances for all pixels in A, the largest distance is stored. This stored value is the first directed Hausdorff distance. Later, images A and B switch places and the process is repeated. Finally, the Hausdorff distance is equal to the largest directed Hausdorff distance between the two directed Hausdorff distances (Eq. (1)).

The Hausdorff distance is highly sensitive to outliers. Let us suppose the presence of well-packed points in an arbitrary region of the space. A single point positioned very far from the rest results in a large Hausdorff distance, even if most of the content in both images are equal. It is therefore preferred to start with a noise removal filter before the calculation of the HD when it comes to real world data.

In terms of efficiency, the main weakness of the Hausdorff distance is the amount of combinations to be evaluated. Every combination of points must be checked twice and this requires more time to process when compared to other popular point or set distances that are used for the same purpose such as Mean Difference, Normalized Correlation, Euclidean, Manhattan, Chebyshev, Rodrigues [18], Mutual Information, and even Chamfer Distance [7]. Although this seems obvious for point-wise distances, HD is still substantially slower than Chamfer [7], which also calculates distances between sets.

In terms of previous work, Krishnamurthy et al. [3] and Kim et al. [19] proposed a GPU-accelerated Hausdorff distance computation for NURBS surfaces but the authors do not employ mathematical morphology. Atallah [20] proposed a linear time algorithm for the Hausdorff distance but it only works with convex polygons (case restriction) and is sequential, built for the CPU. What draws the most attention is the algorithm proposed by Taha and Hanbury [16], which is the state-of-the-art with no restrictions and is much faster than the naive (or brute force) Hausdorff distance computation.

2.1. Taha and Hanbury algorithm

In 2015, Taha and Hanbury [16] proposed an exact and general (unlike other proposals, with no restrictions or approximations) computation of the Hausdorff distance using very simple yet robust ideas such as an early break and randomization. Their proposal outperformed the ITK implementation and is the fastest implementation in the current literature.

Their fast implementation is based on three principles. At first, (1) the authors discard every voxel that is redundant. Overlapping voxels in sets A and B are removed from set A. In some occasions, this step alone can reduce the required processing time dramatically.

The (2) second principle is a randomization (pseudo-randomized) in the order of the voxels, which is performed to reduce the chance of producing a worst-case scenario. The authors perform a linear randomization with no significant increase in processing time.

The (3) third and key principle consists of an early loop break. Let us assume we want to compute the distance of a point a that belongs to A to all points in B and that variable c_{max} represents the maximal global distance found up to the current moment. At the moment the statement $||a - b_j|| < c_{max} \forall j$ becomes true, the distance computation of a is skipped (early break). In other words, if any distance from a to any other point in B is already less than the current global, there is no requirement to check for the rest of the combinations of point a to points in B. That simple rule improves the processing time of HD dramatically. Algorithm 1 depicts the overall directed computation proposed by Taha and Hanbury.

Although Taha and Hanbury propose an overall very efficient implementation, their algorithm still does not perform efficiently

Algorithm 1: Taha-Hanbury directed Hausdorff distance computation.

```
Delete or disregard the useless voxels (A = A − (A ∩ B));
Randomize the ordering of the voxels;
cmax = 0;
for a ∈ A do
    cmin = ∞;
    for b ∈ B do
        d = ||a − b||;
        if d < cmax then // Early break
            break;
        end
        if d < cmin then
            cmin = d;
        end
    end
    if cmin > cmax then
        cmax = cmin;
    end
    return cmax;
end
```

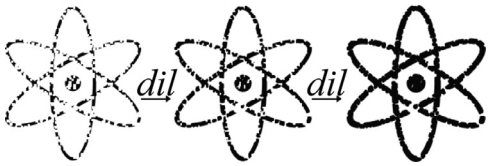


Fig. 2. Two subsequent image dilations.

in certain scenarios such as in the presence of noise. As previously mentioned, voxels are shuffled and the locality information is lost. Furthermore, their implementation consumes more memory when compared to a morphological approach. This work also indicates, in practice, that the proposed approach is faster than the Taha-Hanbury algorithm in the worst-case, for high resolution volumes.

2.2. Mathematical morphology

Mathematical Morphology (MM) [21] studies the geometrical structures in images using mathematics. Its main principle is extracting information related to the geometry and topology of subsets of an image using structuring elements [22]. The structuring element is often defined as a set of elements that represents pixels. Dilations and erosions are the principles and the two main operations in MM [10,23,24]. A binary dilation of image I by a structuring element (an image, eventually called kernel) E is given by Eq. (4):

$$dil^E(I) = \bigcup_{e \in E} I_{-e} \tag{4}$$

Where I_{-e} represents the elements of I translated by $-e$. In other words, the dilation is given by the union of I_{-e} for every element e in E. Fig. 2 illustrates a dilation performed twice on an image using a cross-shaped structuring element $E = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}$.

In practice, the dilation marks the surrounding pixels of previously marked pixels. The surrounding is marked according to the chosen structuring element. For instance, Fig. 3 shows a dilation using the structuring element $E_1 = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}$. Fig. 4 illustrates a growth using the structuring element $E_2 = \{(0,0),(0,1),(0,-1),(1,0),(-1,0),(-1,-1),(1,-1),\$ - 1,1),(1,1)\}$.

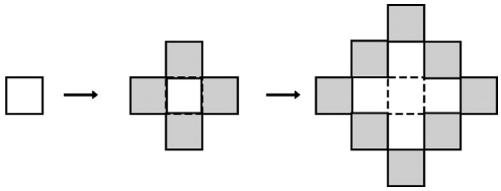


Fig. 3. Pixel growth according to structuring element E_1.

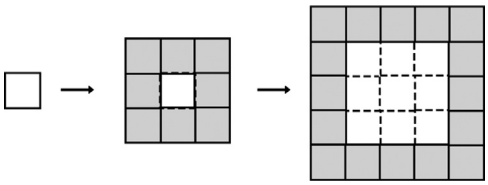


Fig. 4. Pixel growth according to structuring element E_2.

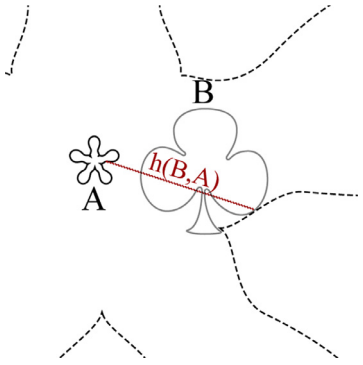


Fig. 5. Dilation of A until it covers B.

In contrast to the methodology proposed by Taha and Hanbury, the proposal in this work uses these dilations from mathematical morphology to compute the HD, which ends up being better suited for parallelization and outperforms the state-of-the-art in certain scenarios.

3. Proposed methodology

Image dilations offer an alternative data-parallel approach for computing the Hausdorff distance. Set A is dilated until it covers set B entirely. The amount of applied dilations is equal to the discrete Hausdorff distance and the shape of the structuring element indicates the metric or distance being used. The proposed morphological algorithm is illustrated in Eq. (5), where $HD^E(A,B)$ represents the discrete Hausdorff distance of sets A and B respecting a structuring element E for discrete rectangular lattices.

$$HD^E(A,B) = q \mid B \subseteq dil^E(A,q), A \subseteq dil^E(B,q), q \in \mathbb{N}_0 \tag{5}$$

where $dil^E(A,q)$ and $dil^E(B,q)$ indicate images or objects A and B dilated q times, in this case using structuring element E, as shown below using image or set of voxels P:

$$\begin{aligned} dil^E(P,n) &= P \cup dil^E(P,n-1) \mid n \in \mathbb{N}_0 \\ &\vdots \\ dil^E(P,0) &= P \end{aligned} \tag{6}$$

and P is the original input image or set of voxels.

The value q is equal to the Hausdorff distance. In other words, q indicates how many dilations are required for set A to cover B and for set B to cover A. Fig. 5 illustrates an image A being dilated until its dilated version covers image B, where the final dilated version of A is shown in dashed lines. In this illustration, $h(B,A)$ is equal

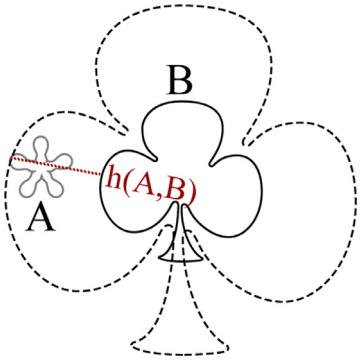


Fig. 6. Dilation of B until it covers A.

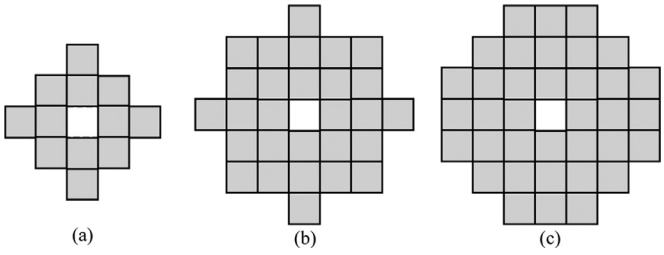


Fig. 7. Potential structuring elements for an approximate Euclidean norm.

to the directed Hausdorff distance using the Euclidean norm. It is interesting to note that the final HD distance matches the point where the covering converges in Figs. 5 and 6.

Similarly, image B is dilated until it covers A, which is shown in Fig. 6. In this case, $h(A, B)$ represents the max distance from A to the nearest point of B. Finally, the Hausdorff distance is given by $max(h(B, A), h(A, B))$, which is also equal to q.

The Euclidean and Supremum norms are the most popularly used in conjunction with HD. However, the Hausdorff distance supports different norms and distances. A previous work [18] indicates that the Euclidean and Supremum norms are equally powerful over 33 different classification problems using the k-Nearest Neighbours classifier [25], which also supports the adoption of the Supremum norm in the experiments herein.

Implementing the Hausdorff distance using morphological operators restricts the selection of the norm due to the discrete or quantized nature of images and 3D objects. It is possible to use a structuring element such as the one shown in Fig. 7-a to simulate the Euclidean norm. As the structuring element is increased (Fig. 7-b and c), a higher precision in relation to the Euclidean norm is obtained. However, this would require a proportionate resizing of the image or object in order to accommodate for the increase of the structuring element, and this impacts the processing time.

In other words, it is possible to use the Euclidean norm with the morphological approach proposed in this work. However, the distance results are approximations. These approximations could be very precise, depending on the resolution and usage of the structuring element. The precision is increased as the structuring element tends to resemble a perfect circle. Fig. 7-c, for instance, is still considerably far from a perfect circle.

The Supremum (L_∞) norm, however, is achieved using the structuring element shown in Fig. 4. The Supremum norm is used throughout this work in order to avoid Euclidean approximations. However, other norms and distances can be used. A possible example for further evaluation is the Rodrigues distance [18], which resembles an octagon and is something between the Euclidean and the Supremum norms. Besides, this distance is very efficient in terms of parallel discrete neighborhood interations, way more than the Euclidean. Therefore, it can be efficiently implemented and further evaluated using the morphological approach proposed herein.

Algorithm 2: Proposed morphological algorithm for the computation of the Hausdorff distance.

Data: Binary sets A and B where D, H and W are the depth, height and width of the 3D environment. A_{aux}, $A_{original}$ and B_{aux}, $B_{original}$ are auxiliary sets of equal dimension to that of A and B and start off as empty matrices.

```
1  A_original = A; B_original = B;
2  q = 0;
3  while true do
4      finished = true;
5      for d = 0; d < D; d++ do
6          for i = 0; i < H; i++ do
7              for j = 0; j < W; j++ do
8                  A_aux[d,i,j] = A[d,i,j] ;          // updating new dilations
9                  B_aux[d,i,j] = B[d,i,j] ;          // updating new dilations
10                 finished = finished &&
                   (B[d,i,j]||!A_original[d,i,j]) &&
                   (A[d,i,j]||!B_original[d,i,j]);
11             end
12         end
13     end
14     if (finished) return q;
15     for d = 0; d < D; d++ do
16         for i = 0; i < H; i++ do
17             for j = 0; j < W; j++ do
18                 if(A_aux[d,i,j]) dilate voxel A[d,i,j];
19                 if(B_aux[d,i,j]) dilate voxel B[d,i,j];
20             end
21         end
22     end
23     q++;
24 end
```

At last, Algorithm 2 describes the proposed morphological algorithm. The algorithm must be computed in the presented order, otherwise the results will not be exact. The duplicate matrices are also a strict requirement.

3.1. Implementation and GPU aspects

Graphical processing units (GPUs) are very popular processors based on the Single Instruction Multiple Data (SIMD) paradigm, usually containing a high number of processing cores, which makes them faster and well-suited for data-parallel tasks in comparison to traditional sequentially-oriented CPUs.

In this work, three GPU implementations are proposed and investigated along with sequential CPU approaches: (1) an implementation of the MM algorithm using global memory, (2) an implementation of the MM algorithm using texture memory and (3) an implementation of the Taha-Hanbury algorithm using global memory. Taha and Hanbury propose and provide a sequential CPU implementation using C/C++. Their algorithm was translated to the GPU paradigm in order to evaluate its parallel robustness. The source code for this implementation, along with datasets and binaries, can be found at [26].

3.1.1. Proposed Taha and Hanbury GPU implementation

The Taha-Hanbury algorithm was implemented as a kernel that computes all the distances from a single point of one of the sets to all the points of the other set, in parallel. The images are later swapped, and the kernel is triggered again. This proposed implementation contains all the three key aspects of their algorithm: (1) the exclusion of redundant voxels, (2) data randomization and (3) early loop break. Steps (1) and (2) are performed as a preprocessing phase on the CPU before data is copied to GPU memory.

The amount of memory space, in bits, required for processing this algorithm on the GPU is shown in Eq. (7):

$$m_{Taha}(v_A, v_B) = (16 \times 3) \times (v_A + v_B) + 32 \tag{7}$$

where v_A and v_B represent the amount of voxels in A and B, respectively. The value 16 stands for the bits of *unsigned shorts*, which are multiplied by 3 axes (x, y and z). The value 32 stands for a global *unsigned integer* that stores the final result.

3.1.2. Proposed GPU MM implementation

Shifting the attention to the proposed parallel implementation, the idea is to theoretically dilate all voxels concurrently, where each GPU thread is responsible for a voxel and mutually checks (using the operation *atomic and*) whether dilated versions covered both original images or volumes.

In general, GPUs work with two types of memory (obviously excluding smaller shared and register memory). Sets A and B can be stored as textures or in global memory. Texture memory is slightly faster than global memory when locality is exploited such as in image convolutions, improving coalescence. However, texture memory consumes more space as sets A and B can be stored using Boolean arrays. Besides, memory costs of the proposed algorithm must be multiplied by 3 due to image convolutions and binary pixel-wise checks that verify if the dilation covered the original image.

The function $m_{Global}(w_A, h_A, d_A, w_B, h_B, d_B)$ in Eq. 8 represents the amount of memory in bits required to process the proposed algorithm using global memory. The cost of the texture memory version is given in Eq. 9. Variables w_A, h_A, d_A, w_B, h_B and d_B represent the width, height and depth of sets, volumes or images A and B, respectively.

$$m_{Global}(w_A, h_A, d_A, w_B, h_B, d_B) = 3 \times (w_A \times h_A \times d_A + w_B \times h_B \times d_B) + 32 \tag{8}$$

$$m_{Texture}(w_A, h_A, d_A, w_B, h_B, d_B) = (3 \times 8) \times (w_A \times h_A \times d_A + w_B \times h_B \times d_B) + 32 \tag{9}$$

Eq. 9 contains 8 extra bits per voxel that are wasted due to the usage of texture memory. The value 32 represents the cost of an *integer* variable that is atomically operated and indicates when the processing halts. The value 3 stands for the (1) input A and B, (2) two sets that are used to read and (3) two sets that are used to write the new dilations.

Algorithms 3 and 4 represent in more detail the proposed computation of the HD on GPUs. The code that runs on the CPU (Algorithm 4) calls the GPU kernel shown in Algorithm 3. The returned d in this case is the Hausdorff distance. The *atomicAnd* function places at variable *dFinished* the result of the verification (B_1[id] || !A_1[id]) && (A_1[id] || !B_1[id]), which ensures that the data is visible to all threads.

In summary, the Taha-Hanbury algorithm consumes up to 16 times more memory than the global memory approach and up to 2 times more in relation to the texture approach. Both texture and global memory approaches are mathematical morphology approaches.

Memory consumption is a crucial concept, specially for the GPU, as high resolution volumes must be allocated at once in GPU memory before starting with the computation of the HD (at least for the current implementations). If the global GPU memory is insufficient, it is simply not possible to store and consequently to compute the Hausdorff distance. In some specific cases of the experiments herein, the Taha-Hanbury GPU approach was not able to compute the distance due to memory overflow, while the morphology approach succeeded.

The same reasoning applies to the CPU, the difference is that the distance can still be computed but the completion will be tremendously delayed as memory data will be repeatedly written to and read from the hard disk usually by the operational system due to the lack of space.

Algorithm 3: GPU kernel that dilates both images and halts the computation.

Data: Binary sets A and B and two copies of A: A_1, A_2, as well as two copies of B: B_1 and B_2. The variable *dFinished* indicates the convergence of the device.

```
// id is a unique voxel index;
int id = blockDim.x * blockIdx.x + threadIdx.x;
If a voxel exists at position id in A1, dilate the position and write to A2;
If a voxel exists at position id in B1, dilate the position and write to B2;
atomicAnd(dFinished, (B1[id] || !A1[id]) && (A1[id] || !B1[id]));
```

Algorithm 4: CPU part of GPU code that computes the Hausdorff distance.

Data: Binary sets A and B.

```
d = 0;
A1 = A; A2 = A;
B1 = B; B2 = B;
Allocate and copy A, A1, A2, B, B1 and B2 to the GPU;
Allocate dFinished in the GPU;
dFinished = false;
finished = false;
while !finished do
    Kernel call (Algorithm 3);
    finished = dFinished;
    A1 = A2;
    B1 = B2;
    d = d + 1;
end
return d;
```

4. Experimental results

Several 2D and 3D experiments are described in this section. The distance parameter used throughout this section is the sup norm as previously mentioned. This applies for Taha-Hanbury and also for the MM approach in order to provide a fair comparison.

The first group of 2D experiments consists of a noise test, one image contains noise and the other is entirely populated with white pixels (white equals valid pixels). This group represents a near worst-case scenario for the Taha-Hanbury algorithm.

The second group of 2D experiments employ small displacements in one of the images in order to emulate image registrations. As the proposed methodology is aware of locality, it is natural to expect that it ends up consuming less processing power in contrast to Taha-Hanbury.

The third group of 2D experiments considers sparse images and a substantial displacement between the objects, forcing a near worst case scenario for the MM approach. In this case, the Taha-Hanbury algorithm outperforms the proposed approach.

Table 1
Run time (s) - average of 10 runs - for the noise experiment (Fig. 8).

Algorithm	64×64	128×128	256×256	512×512	1024×1024	2048×2048	4096×4096	8192×8192
Taha and Hanbury (C+)[16]	0.002998	0.024523	0.353799	3.4383	54.0448	857.837	13785.5	153461.1
Taha and Hanbury (CUDA)†	0.408599	0.40649	0.434575	0.611139	3.89732	42.9724	744.021	a
Naive (C+)	0.037027	0.67297	10.6756	168.573	2951.67	41618.4	771960	b
Morphology (C++)*	**0.000502**	**0.001001**	**0.004002**	**0.019979**	**0.082601**	**0.333781**	1.37698	6.87304
Naive (Java)	0.096209	0.899875	14.07467	366.557	6364.611	81358.71	1623174	b
Morphology (Java)*	0.012128	0.017561	0.025339	0.055646	0.146806	0.34243	1.300063	5.851308
Naive (Java JET)	0.038257	0.664049	12.18436	174.1085	2865.031	49428.67	718384.7	b
Morphology (Java JET)*	0.001424	0.002929	0.006933	0.030656	0.120513	0.471395	1.955404	10.54566
Morphology Boolean (CUDA)*	0.369811	0.367287	0.371267	0.377776	0.415853	0.422825	**0.425893**	**0.693568**
Morphology Texture (CUDA)*	0.380295	0.366826	0.360454	0.373763	0.378281	0.379847	0.446315	a

* Denotes different implementations of the algorithm proposed in this work. † Denotes a GPU implementation of the algorithm proposed by Taha et al. [a] Memory overflow. [b] Exceeded the stipulated time limit for processing (2,000,000s).

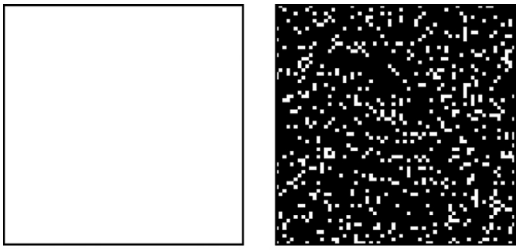


Fig. 8. Images *A* and *B* for the first group of experiments.

Later, a trade-off experiment is performed. The performance of the algorithms is investigated as pixels are introduced to both images in real time.

The fourth and last group of experiments uses 3D datasets and compares both approaches (Taha-Hanbury and MM). The first dataset contains computed tomography images of the thorax and the second contains hepatocyte cells.

4.1. Noise experiment

Fig. 8 shows images *A* and *B* for this first group of experiments. The white dots in image *B* were generated according to the rule shown in Eq. 10, where 1 represents the white color and 0 represents black. The symbol % represents division remainder and *rand* a random number generator.

$$I_{y,x} = \begin{cases} 1, & if\ rand()\%7 = 0 \\ 0, & otherwise \end{cases} \tag{10}$$

The Taha-Hanbury algorithm does not work particularly well with noisy sets because intersections do no occur very frequently and their method is blind in terms of locality. Contrariwise, their approach performs a randomization and shuffles voxels, discarding their order. Table 1 shows the results obtained with different algorithms and implementations using the images shown in Fig. 8.

Table 1 includes a Naive (or brute force) implementation that is essentially the simplest and most trivial way of computing the Hausdorff distance. In this approach, all possible combinations of voxels are naively checked. The Excelsior JET Ahead-of-Time (AOT) compiler, shown as "Java JET" in Table 1, is a native bytecode Java compiler. In other words, it translates Java code to native binary code (e.g. it generates .exe files in Windows). In contrast to the Hotspot JVM, which implements a Just-in-Time (JIT) compiler, the AOT compiler can perform optimizations that are similar to the ones applied in C/C++ code. Furthermore, CUDA is a proprietary general purpose programming language from Nvidia for the GPU.

Table 1 shows that results obtained with the proposed morphological approach are far better than the ones obtained with the Taha-Hanbury algorithm. In fact, the morphological approach is up to 22,327 times faster than the Taha-Hanbury algorithm using CPU implementations (Morphology C++ vs Taha and Hanbury C++ using 8192×8192 images), and up to 1,700 times faster using GPU implementations (Morphology Boolean CUDA vs Taha and Hanbury CUDA using 4096×4096 images). For this experiment, the morphological approach was substantially faster than the proposal of Taha and Hanbury.

In addition, the Java JET implementation improved run times of the Naive approach in relation to Java code that runs on top of the Java Virtual Machine (Table 1). Run times obtained with the JET version were similar to the ones obtained with C++. Besides, it is also of interest noting that the morphological version completes the computation faster using Java in contrast to C++ in two occasions. This improvement can be observed as the size of the images is increased, which leads to more accurate JIT optimizations.

4.2. Registration experiment

Time series images can suffer from lack of alignment between frame acquisitions. Image registration fixes this problem by transforming or distorting one of the images in order to fit into or resemble the other.

In theory, computing the Hausdorff distance using the morphological approach is faster when it comes to image registration because patient movements are influenced by locality, e.g., a person in a frame of a time series acquisition is displaced by a small margin in relation to its previous frame. In other words, every pixel of its silhouette, for instance, is usually close to its correspondent pixel in a subsequent frame in time. As images are dilated, the dilated version rapidly engulfs the other image (e.g., the silhouette rapidly engulfs other silhouettes of other frames), converging very fast. The standard approach for computing the Hausdorff distance (including Taha and Hanbury) is not sensible to this locality feature. As previously mentioned, it shuffles the voxels and computes the distances individually in no particular intelligent order.

Fig. 9 shows sets *A* and *B* used in this experiment. Some of the shapes in these images are considered hard to compute in terms of the Hausdorff distance (e.g., the spiral). However, the aim of this experiment is to compare translated versions of the same image, or very similar ones. Thus, the second image is a translated version of the first (130 pixels of difference). This mimicks the registration operation used to align medical images in computed tomography, magnetic resonance imaging and other types of imaging modalities [27]. A superposition of these images is shown in image (c), where the white color represents their intersection.

Table 2 shows the obtained results for this experiment. Once again, the proposed algorithm outperformed the state-of-the-art

Table 2
Run time (s) - average of 10 runs - for the registration experiment (Fig. 9).

Algorithm	64×64	128×128	256×256	512×512	1024×1024	2048×2048	4096×4096	8192×8192
Taha and Hanbury (C+)[16]	0.0050027	0.041518	0.179235	1.405045	10.22517	79.88444	279.6418	1116.724
Taha and Hanbury (CUDA)†	0.3883912	0.394526	0.39196	0.496835	1.412638	14.88276	92.98898	455.2397
Morphology (C++)*	**0.0000032**	**0.000501**	**0.002001**	**0.011004**	**0.085561**	0.587431	4.945524	41.55378
Morphology (Java)*	0.0111961	0.012775	0.22365	0.052863	0.209393	0.957885	7.392813	58.65267
Morphology (Java JET)*	0.0015751	0.001397	0.003867	0.019434	0.138745	1.051424	8.748907	70.90767
Morphology Boolean (CUDA)*	0.3811541	0.388424	0.377013	0.389778	0.399346	0.432474	**0.717577**	**2.689651**
Morphology Texture (CUDA)*	0.3895547	0.378583	0.388856	0.388288	0.390921	**0.431387**	0.726946	a

* Denotes different implementations of the algorithm proposed in this work. † Denotes a GPU implementation of the algorithm proposed by Taha et al.
[a] Memory overflow.

Fig. 9. Images *A* and *B* for the image registration experiment. The image (c) represents the superposition of images (a) and (b).

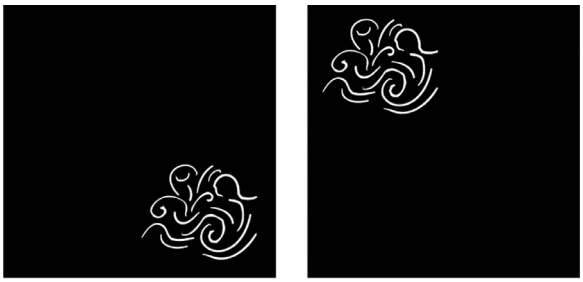


Fig. 10. Images *A* and *B* for the accentuated displacement experiment.

algorithm substantially. The morphological approach is approximately 27 times faster than the Taha-Hanbury algorithm when it comes to the CPU (Morphology C++ vs Taha and Hanbury C++ using 8192×8192 images), and up to 170 times faster when it comes to GPU implementations (Morphology Boolean CUDA vs Taha and Hanbury CUDA using 8192×8192 images).

The implementation using texture memory is slightly faster than the one using global boolean memory (2048×2048 column). However, it is not that interesting to use texture memory rather than global boolean arrays as it consumes more space and does not deliver a huge efficiency trade in comparison.

4.3. Accentuated displacement experiment

The third group of experiments considers less pixels (sparse images) and expressive translations between objects of images (a) and (b). Figure 10 shows the two images used in this third experiment. The objects in these images are separated by an expressive translation, the first object is in the bottom-right edge and the second in the top-left edge. This configuration requires several iterations, or dilations, for the morphological approach.

This fact, along with the sparse property, favors the approach of Taha-Hanbury. The Taha-Hanbury approach “erases” or does not consider black pixels, accelerating the process. Large distances between objects along with sparsity favors them as valid pixels are inserted into an array and the distances are computed using pure mathematical distance calculations (not based on iterative dilations). Table 3 shows the results obtained in this occasion.

As expected, the Taha-Hanbury algorithm is faster than the proposed approach in this case (Table 3). When it comes to the 8192×8192 column, the Taha-Hanbury algorithm is up to 1,147 faster regarding CPU implementations (Morphology C++ vs Taha and Hanbury C++) and up to 52 times faster using GPU implementations (Morphology Boolean CUDA vs Taha and Hanbury CUDA).

4.4. Worst-case analysis

The worst-case complexity for the Taha-Hanbury algorithm is $O(n * m)$ where n and m are the numbers of white points (valid pixels) in images A and B, respectively. In the worst-case no early break occurs and therefore every distance for each point in A to B is computed, resulting in $n * m$ steps.

In terms of the morphological approach, the worst-case occurs when sets A and B contain pixels at their extremes. In other words, the worst-case is reached when image A contains one pixel at the top-left and B contains one pixel at the bottom-right position. In this case, both pixels (or images) need to be dilated the maximal amount to cover the other set. Therefore, the distance in this case is equal to the diagonal of the image (assuming the sup norm). The diagonal is written in terms of width and height. The worst-case complexity is then given by $O((w + h))$ where w and h are the width and height of the grid, respectively. If we consider the width to increase along with increases in height and vice-versa (which is something that happens with real world data), we can assume a complexity of $O(w*h)$.

In summary, when we analyse the complexity of Taha-Hanbury we consider the points in the space (valid white pixels). However, our complexity is described in terms of the grid. If we consider n and m to be equal to the amount of spaces in the grid (or written in terms of the grid - again, which is something that happens with real world data), then it is possible to conclude that the worst-case complexities of both approaches are mathematically equivalent.

Although mathematically equivalent, run time is a whole different story. Certain computation patterns are favored in certain occasions. Furthermore, mathematical complexity will not always be able to properly estimate algorithmic run time in practical scenarios. In this sense, Table 4 shows the run times for the described worst-case of the proposed approach, where each image has to be fully dilated.

The practical worst-case of the algorithm proposed in this work is already substantially faster than a near worst-case scenario of the Taha-Hanbury algorithm for images or grids of equal size. In fact, for 8192×8192 images, the proposed algorithm is approximately 35 times faster when comparing CPU implementations (run times in Table 4 compared to the ones obtained with the Taha-Hanbury algorithm in Table 1). The proposed algorithm is approximately 25 times faster for 4096×4096 images using GPU implementations, also in the worst-case.

Table 3
Run time (s) - average of 10 runs - for the accentuated displacement experiment (Fig. 10).

Algorithm	64×64	128×128	256×256	512×512	1024×1024	2048×2048	4096×4096	8192×8192
Taha and Hanbury (C+) [16]	**0.0004989**	**0.001001**	**0.002001**	**0.007499**	**0.029021**	**0.125088**	**0.596422**	2.70062
Taha and Hanbury (CUDA)†	0.431381	0.450688	0.421374	0.443881	0.419944	0.50206	0.815873	**2.34358**
Morphology (C++)*	0.0015191	0.007997	0.084830	0.697414	5.61795	50.2108	383.501	3098.9
Morphology (Java)*	0.020383	0.045346	0.157808	0.854481	9.566188	85.02071	742.4422	5269.256
Morphology (Java JET)*	0.003104	0.017457	0.139878	1.077426	9.050213	81.0148	695.722	4942.925
Morphology Boolean (CUDA)*	0.393354	0.406813	0.456598	0.453802	0.732357	2.48971	16.0314	122.688
Morphology Texture (CUDA)*	0.401651	0.419039	0.409417	0.492438	0.790893	2.92891	19.2741	a

* Denotes different implementations of the algorithm proposed in this work. † Denotes a GPU implementation of the algorithm proposed by Taha et al.
[a] Memory overflow.

Table 4
Run time (s) - average of 10 runs - for the worst case of the proposed morphological algorithm.

Algorithm	64×64	128×128	256×256	512×512	1024×1024	2048×2048	4096×4096	8192×8192
Morphology (C++)*	**0.002001**	**0.01452**	**0.11307**	0.909646	7.89133	62.8916	533.216	4347.62
Morphology (Java)*	0.020033	0.044321	0.195341	1.678677	14.82279	110.1916	1022.649	7212.093
Morphology (Java JET)*	0.004371	0.02474	0.194065	1.807945	14.62916	118.3291	908.1043	8067.729
Morphology Boolean (CUDA)*	0.399017	0.403596	0.425439	**0.492488**	**0.939122**	**4.15689**	**28.9213**	**224.43**
Morphology Texture (CUDA)*	0.396506	0.405052	0.429441	0.531527	1.10393	5.12952	36.4861	a

* Denotes different implementations of the algorithm proposed in this work. [a] Memory overflow.

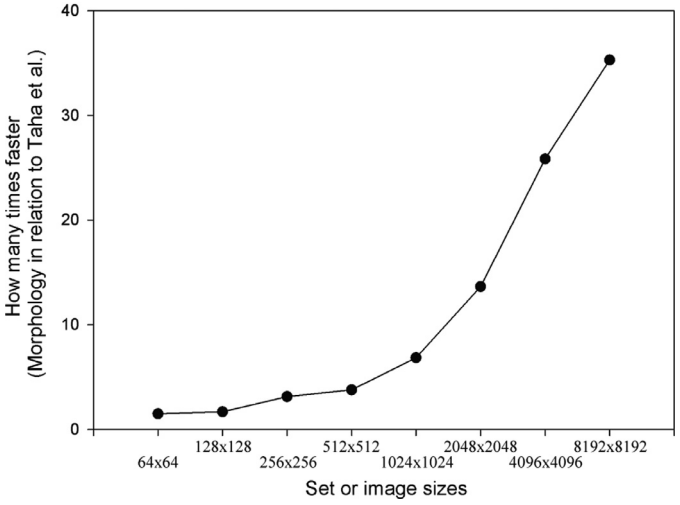


Fig. 11. How many times faster the worst case of the proposed morphological algorithm is in regard to a near worst case scenario of the Taha-Hanbury algorithm (CPU implementations only).

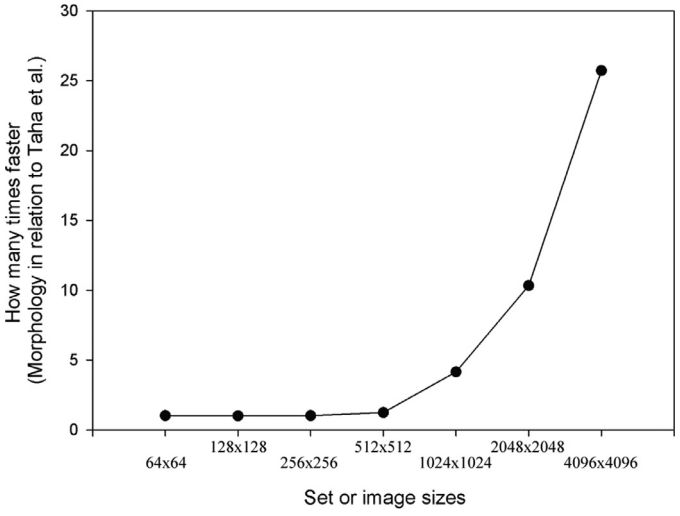


Fig. 12. How many times faster the worst case of the proposed morphological algorithm is in regard to a near worst case scenario of the Taha-Hanbury algorithm (GPU implementations only).

Figures 11 and 12 illustrate how many times faster the worst-case of the proposed morphological algorithm is in relation to a near worst-case of Taha-Hanbury algorithm comparing implementations for the CPU and GPU, respectively. The implementation proposed in this work is faster in the worst-case for all the investigated image or grid sizes. Furthermore, the improvement appears to be exponential in the worst-case, as shown by those two figures.

4.5. Tradeoff experiment

At last, this section investigates the algorithmic behavior as the amount of pixels in the grid is increased. As these experiments require a significant amount of processing power, this case is restricted to grid sizes of 1024×1024 and to the C++ sequential CPU implementation.

At first, we perform an experiment increasing the amount of pixels in an image starting from the worst-case scenario of the MM approach. That is, image A starts with a pixel in the top-left position while B starts with a pixel at the bottom-right. Pixels are inserted respecting this direction (from the edges towards the center of the image) in further iterations. Fig. 13 shows the run times obtained in this experiment.

The morphological algorithm starts off almost 100 times slower than the Taha-Hanbury algorithm (Fig. 13). However, as the amount of pixels in the image increases, the MM proposal starts to outperform Taha-Hanbury after slightly more than half of the space in the grid is populated. This moment is illustrated by a red line in Fig. 13. It is of importance noting that worst-cases are not being forced for the Taha-Hanbury approach in this case, just for the proposed MM algorithm.

In addition, Fig. 14 shows run times when pixels are randomly inserted in the image grid. This case forces a near worst-case scenario for the Taha-Hanbury approach, as opposed to the previous case, which forced a worst-case for the MM approach. In this case, the MM approach outperforms the Taha-Hanbury algorithm much earlier, illustrated by the red line.

At last, an "average" case that is not the worst for MM neither for Taha-Hanbury is presented. In this last 2D experiment the image grid is populated from the center of the image instead of start-

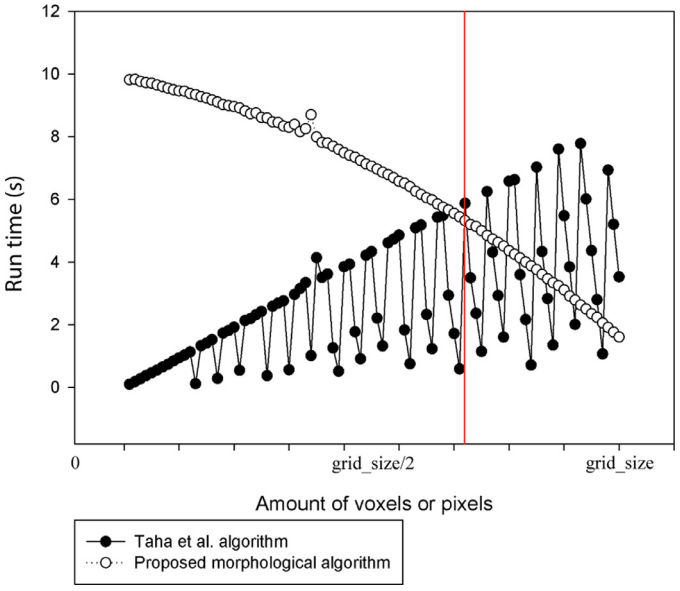


Fig. 13. Comparison of run time (s) as the amount of pixels is increased in both images, starting with the worst case for the proposed Mophological algorithm (sequential C++ CPU approaches).

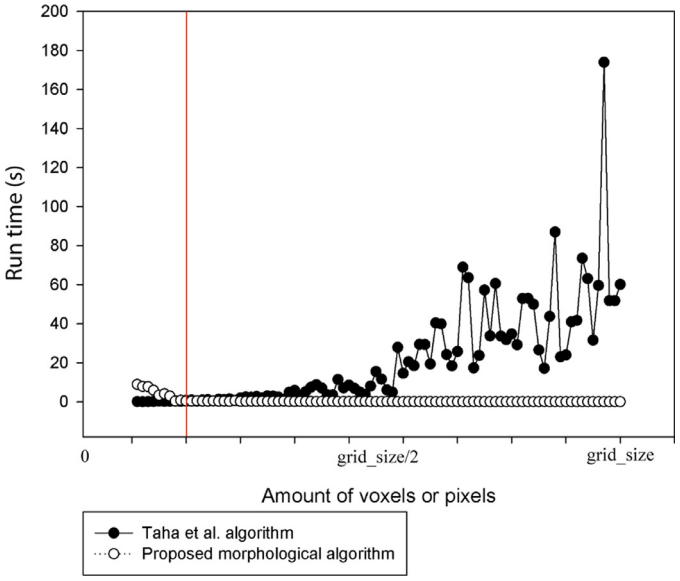


Fig. 14. Comparison of run times (s) as the amount of pixels is increased in both images starting with a near worst case scenario for the Taha-Hanbury algorithm (sequential C++ CPU approaches).

ing from the edges. This leads to a very clear picture of where and when the MM approach starts to outperform Taha-Hanbury. This occurs as soon as slightly more than half of the grid space is populated by white pixels. This moment is illustrated by the red line in Fig. 15. By the end of the analysis, the MM approach ends up being almost 16 times faster.

4.6. 3D Experiments

This section presents two sets of experiments using 3D medical datasets and CPU implementations. Both approaches were compiled and run using C/C++ and optimization flags. GPU versions of the algorithms were discarded as these are low resolution volumes and the benefit would be minimal. However, GPU approaches are still very important, especially for the case of high resolution volumes.

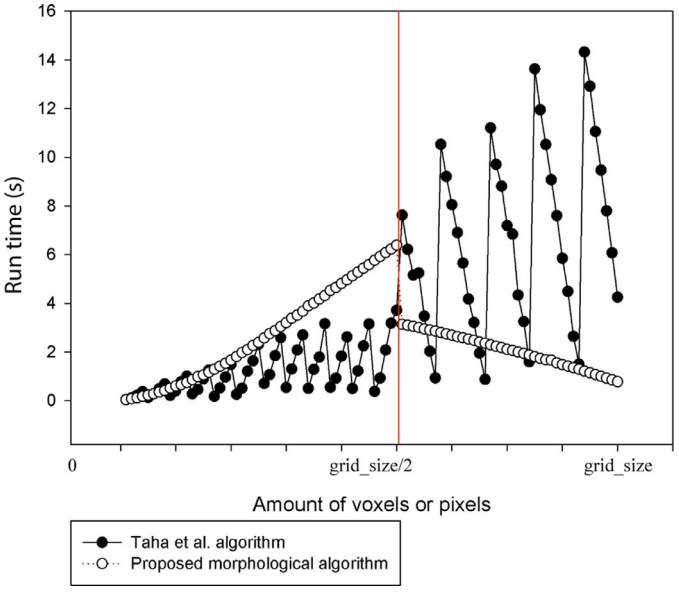


Fig. 15. Comparison of run times (s) as the amount of pixels is increased in both images respecting a balanced scenario (sequential C++ CPU approaches).

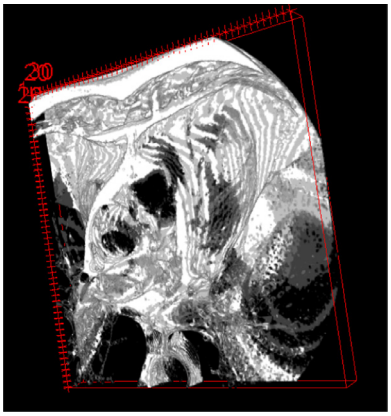


Fig. 16. 3D representation of fat tissues in a computed tomography exam of the thorax (512x512x41 voxels of space).

4.6.1. Cardiac computed tomography volume

The first dataset refers to a previous work that consisted of segmenting fat tissues of the outer and inner surfaces of the human heart [7,28,29]. A scanned patient contains 41 slices (or images), as shown in Fig. 16. Each image has its width and height equal to 512 pixels.

A total of 5 randomly chosen patients were selected from this computed tomography dataset, which is available at [7]. The thoracic scans are thresholded to the fat tissue range, generating binary images. In this experiment, different patients are compared to each other using the Hausdorff distance as if we were to perform an intersubject image registration in order to align their volumes to a common position. Table 5 shows, in seconds, run times obtained using the Taha-Hanbury algorithm while Table 6 shows the run times for the MM approach.

This set of experiments yields balanced results. The proposed MM approach was better in 8 cases and the Taha-Hanbury approach was better in 7 cases. This occurs because of the low resolution volumes and also because the scans are substantially sparse, which favors Taha-Hanbury. Nowadays, it is already possible to ob-

Table 5
Run time (s) obtained with Taha-Hanbury algorithm using 5 computed tomography scans of the thorax.

3D Patients	Acel	Aedu	Afre	Amar	Axav
Acel	0.7866	2.04	4.68	9.54	46.55
Aedu		0.51269	2.39	5.6	42.75
Afre			0.6853	11.6284	36.99
Amar				0.6154	3.97
Axav					0.65

Results are the average of 10 runs in seconds.

Table 6
Run time (s) obtained with morphological proposed approach using 5 computed tomography scans of the thorax.

3D Patients	Acel	Aedu	Afre	Amar	Axav
Acel	**0.2557**	20.34	17.19	41.25	**26.84**
Aedu		**0.2025**	29.84	29.79	**26.44**
Afre			**0.2289**	29.391	**28.74**
Amar				**0.233**	40.37
Axav					**0.22**

Bold denotes occasions where the proposed method outperformed the Taha-Hanbury algorithm. Results are the average of 10 runs in seconds.

Table 7
Comparison of run time (s) considering computed tomography registrations.

	Acel	Aedu	Afre	Amar	Axav
Taha-Hanbury [16]	63.47	33.18	44.75	78.96	94.95
Rodrigues (Proposal)	**24.86**	**25.24**	**26.69**	**24.19**	**25.68**

Bold denotes occasions where the proposed method outperforms the Taha-Hanbury algorithm. Results are the average of 10 runs in seconds.

tain computed tomography scans that contain up to thousands of images where each image resolution axis is also in its thousands. As the resolution increases and sparsity decreases, the proposed MM approach is favored.

The scenario changes when we compute the HD between shifted or translated CT exams. The morphological approach is favored when a similar scan has its distance computed in relation to its translated version. In fact, an experiment where these scans are translated by 50 voxels on the x and y axes is shown in Table 7. This analysis mimics a registration scenario, where the HD is required in order to align the acquired medical data. Table 7 confirms that the proposed methology can be up to four times faster than the state-of-the-art proposal by Taha and Hanbury, even for mildly sparse and low resolution volumes, which is this case.

4.6.2. Hepatocyte cells dataset

In this section, experiments using the Hepatocyte dataset provided by [30] are described. Each volume packs a total of 45 images, each being 1392x1040. The first 6 volumes were used: A0, A1, C0, C1, E0 and E1, in no particular biased reasoning. In this case, we are comparing the images to check by how much they resemble each other using the Hausdorff distance. In order to do that, we perform threshold operations, highlighting the cells. The thresholded version can be found at [26]. Fig. 17 illustrates the 3D representation of one of these volumes.

The content in these images is close to a random placement of cells, in a noise-like pattern. This fact already indicates that the proposed approach is favored. Table 8 shows the run times obtained using the Taha-Hanbury algorithm. In contrast, Table 9 shows the ones obtained with the proposed approach. As expected, the proposed approach is much faster than Taha-Hanbury.

Two factors lead to these observations. First, these are higher resolution volumes in relation to the previous thorax exams. The

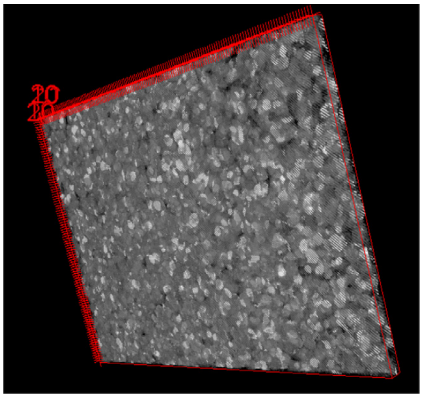


Fig. 17. 3D representation of hepatocyte cells (1392x1040x45 voxels).

Table 8
Comparison of run time (s) on hepatocyte cells using the Taha-Hanbury algorithm.

Volumes	A0	A1	C0	C1	E0	E1
A0	4.89	171.79	291.32	242.01	173.18	174.82
A1		7.37	452.97	742.33	300.08	734.19
C0			5.54	425.18	424.67	417.03
C1				7.1	353.55	684.86
E0					5.84	336.72
E1						6.32

Results are the average of 10 runs in seconds.

Table 9
Comparison of run times (s) on hepatocyte cells using the proposed morphological algorithm.

Volumes	A0	A1	C0	C1	E0	E1
A0	**1.38**	**56.59**	**45.42**	**54.66**	**56.61**	**55.71**
A1		**1.64**	**43.74**	**37.08**	**49.72**	**36.34**
C0			**1.46**	**42.06**	**41.86**	**45.55**
C1				**1.45**	**49.61**	**41.1**
E0					**1.49**	**47.82**
E1						**1.56**

Results are the average of 10 runs in seconds.

second aspect relates to the contents of the space leaving less room for sparsity. These facts favor the proposed approach, as previously discussed. In this last experiment, the proposed approach is, on average, 8.36 times faster than the Taha-Hanbury algorithm, and outperformed its counterpart in every occasion. Improvements can be much greater as high resolution volumes and GPU approaches are used.

As a final remark, all the experiments performed in this work were run using an Intel i7-4720HQ CPU, clocked at 2.60Ghz (no turbo boost) using 16gb of DDR3 memory and a Nvidia GTX 960M (4gb of global memory) under controlled and supervised conditions (including temperature in order to avoid throttling). All the used source code, binaries and datasets are available at [26].

5. Conclusion

This work proposes an algorithm based on mathematical morphology for calculating the computationally expensive Hausdorff distance. One of the key aspects of the proposal is that it is indirectly capable of exploiting locality and hence is faster in tasks such as image registrations. The locality feature is not included in the current state-of-the-art approach as Taha and Hanbury shuffle the voxels and discard order information. This is one of the key aspects of their approach, which improves their efficiency. Furthermore, the MM proposal also computes the exact Hausdorff distance when it comes to the sup metric and rectangular lattices, with no

dataset or case restriction. It is parallel-oriented and more efficient than the state-of-the-art for a bunch of cases, including high resolution volumes.

Overall, performed experiments indicate that the MM approach outperforms Taha-Hanbury in worst-case scenarios, and that the improvement exponentially increases with the linear increase of the grid size. Two out of three 2D experiments indicate better performances for the MM approach and one out of two 3D experiments indicates better performances for the MM approach in contrast to the Taha-Hanbury algorithm.

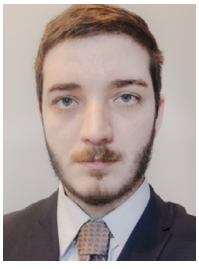
The first conclusion of the individual benefits of the evaluated algorithms is that the proposed MM approach is better suited for high resolution data irrespective of the platform (CPU or GPU), though the performance improvement is even more expressive in terms of GPUs. Taha and Hanbury usually outperform the proposed MM approach when it comes to low resolution volumes and/or when they are sparse and/or when the images contain lots of intersections, which is the second conclusion. The third major conclusion refers to the fact that the proposed morphological approach is better suited for image registrations as long as the volumes are not very sparse, which would probably never occur in medical real world data. This novel MM implementation has tremendous benefits and lots of room for practical applicability.

Declaration of competing interest

The author declares no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. Takacs, Comparing face images using the modified Hausdorff distance, Pattern Recognit. 31 (1998) 1873–1881.
[2] D. Huttenlocher, G. Klanderman, W. Rucklidge, Comparing images using the Hausdorff distance, IEEE Trans. Pattern Anal. Mach. Intel. 15 (1993) 850–863.
[3] A. Krishnamurthy, S. McMains, I. Hanniel, Gpu-accelerated Hausdorff distance computation between dynamic deformable nurbs surfaces, Comput.-Aided Des. 43 (2011) 1370–1379.
[4] T. Hrkac, Z. Kalafatic, J. Krapac, Infrared-visual image registration based on corners and Hausdorff distance, Lect. Notes Comput. Sci. 4522 (2007) 383–392.
[5] A. Fedorov, Evaluation of brain MRI alignment with the robust Hausdorff distance measures, Adv. Vis. Comput. ISVC 2008. Lect. Notes Comput. Sci. 5358 (2008).
[6] M. Lalonde, M. Beaulieu, L. Gagnon, Fast and robust optic disc detection using pyramidal decomposition and Hausdorff-based template matching, IEEE Trans. Med. Imaging (2001) 1193–1200.
[7] E. Rodrigues, F. Morais, N. Morais, L. Conci, L. Neto, A. Conci, A novel approach for the automated segmentation and volume quantification of cardiac fats on computed tomography, Comput. Methods Programs Biomed. (2015).
[8] E. Vivek, N. Sudha, Gray Hausdorff distance measure for comparing face images, IEEE Trans. Inf. Forensics Secur. 1 (2006) 342–349.
[9] W.J. Rucklidge, Efficiently locating objects using the Hausdorff distance, Int. J. Comput. Vis. 24 (1997) 251–270.
[10] E.O. Rodrigues, A. Conci, P. Liatsis, Morphological classifiers, Pattern Recognit. 84 (2018) 82–96.
[11] E.O. Rodrigues, L. Torok, P.L.J. Viterbo, A. Conci, k-ms: a novel clustering algorithm based on morphological reconstruction, Pattern Recognit. 66 (2017) 392–403.
[12] P. Spyridonos, G. Gaitanis, I.D. Bassukas, M. Tzaphlidou, Gray hausdorff distance measure for medical image comparison in dermatology: Evaluation of treatment effectiveness by image similarity, Skin Res. Technology (2012).
[13] F. Morain-Nicolier, S. Lebonvallet, E. Baudrier, S. Ruan, Hausdorff distance based 3d quantification of brain tumor evolution from mri images, in: Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007.
[14] L.F. Silva, D.C. Saade, G.O. Sequeiros, A.C. Silva, A.C. Paiva, R.S. Bravo, A. Conci, A new database for breast research with infrared image, J. Med. Imaging Health Inform. 4 (2014) 92–100.
[15] J. Serra, Hausdorff distances and interpolations, in: Proceedings of the Fourth International Symposium on Mathematical Morphology and Its Applications to Image and Signal Processing, 1998.
[16] A.A. Taha, A. Hanbury, An efficient algorithm for calculating the exact Hausdorff distance, IEEE Trans. Pattern Anal. Mach. Intel. 37 (2015) 2153–2163.
[17] N. Li-pi, J. Xiu-hua, Z. Wen-hui, S. Dong-xin, Image registration based on Hausdorff distance, in: International Conference on Networking and Information Technology, 2010, pp. 252–256.
[18] E.O. Rodrigues, Combining Minkowski and Chebyshev: new distance proposal and survey of distance metrics using k-nearest neighbours classifier, Pattern Recognit. Lett. 110 (2018) 66–71.
[19] Y.-J. Kim, Y.-T. Oh, S.-H. Yoon, M.-S. Kim, G. Elber, Efficient Hausdorff distance computation for free form geometric models in close proximity, Comput.-Aided Des. 45 (2013) 270–276.
[20] M.J. Atallah, A linear time algorithm for the Hausdorff distance between convex polygons, Inf. Process. Lett. 17 (1983) 207–209.
[21] G. Matheron, J. Serra, The birth of mathematical morphology, 2002, (http://cmm.mines-paristech.fr/~serra/communications_pdf/C-72.pdf).
[22] J. Crespo, J. Serra, R.W. Schafer, Theoretical aspects of morphological filters by reconstruction, Signal Process 47 (1995) 201–225.
[23] J. Serra, P. Soille, Mathematical morphology and its applications to image processing, Comput. Imaging Vis., 1994.
[24] F.Y. Shih, Image Processing and Mathematical Morphology: Fundamentals and Applications, 1, CRC Press, 2009.
[25] E.O. Rodrigues, A. Conci, P. Liatsis, Element: multi-modal retinal vessel segmentation based on a coupled region growing and machine learning approach, IEEE J. Biomed. Health Inform. 24 (2020) 3507–3519.
[26] É.O. Rodrigues, Mathematical morphology Hausdorff distance source code, 2015, (https://github.com/Oyatsumi/HausdorffDistanceComparison).
[27] L.F. Silva, A.A.S.M.D. Santos, R.S. Bravo, A.C. Silva, D.C. Muchaluat-Saade, A. Conci, Hybrid analysis for indicating patients with breast cancer using temperature time series, Comput. Methods Programs Biomed. 130 (2016) 142–153.
[28] E.O. Rodrigues, L.O. Rodrigues, L.S.N. Oliveira, A. Conci, P. Liatsis, Automated recognition of the pericardium contour on processed ct images using genetic algorithms, Comput. Biol. Med. 87 (1) (2017) 38–45.
[29] E.O. Rodrigues, V.H.A. Pinheiro, P. Liatsis, A. Conci, Machine learning in the prediction of cardiac epicardial and mediastinal fat volumes, Comput. Biol. Med. 89 (1) (2017) 520–529.
[30] D.J. Logan, J. Shan, S.N. Bhatia, A.E. Carpenter, Quantifying co-cultured cell phenotypes in high-throughput using pixel-based classification, Methods 96 (2016) 5–11.



Érick O. Rodrigues is a professor of computer science at the Department of Academic Informatics (DAINF) in Universidade Tecnológica Federal do Paraná (UTFPR), Brazil. He received a couple of awards during his graduation which includes 2 awards for the “best PhD thesis in exact sciences”, an honorable mention from the Brazilian government (CAPES) for his PhD thesis, being one of the top three best PhD thesis in computer science in Brazil and a “best M.Sc. dissertation in exact sciences”. His main topics of interest are computer vision, machine learning, optimization, GPU parallelism and biological/medical informatics.