Applied Mathematical Sciences, Vol. 1, 2007, no. 4, 145 - 164

On the Computation of the Hausdorff Metric between Digitised Images in Three Dimensions

V. Drakopoulos

Department of Informatics & Telecommunications
Theoretical Informatics, University of Athens
Panepistimioupolis 157 84, Athens, Hellas
vasilios@di.uoa.gr

N. Nikolaou

Division of Information Systems
Bank of Greece
341 Mesogeion Ave., 152 32, Cholargos, Hellas
npnikolaou@bankofgreece.gr

Abstract

Efficient and fast solutions to the problem of computing the Hausdorff metric between two arbitrary grey-scale, digitised images are considered. The more efficient of the techniques proposed here considers the grey-scale attribute of the images as the third dimension. To test the performance of our methods, we apply them to compare pairs of monochrome fractal objects as well as to compare real-world images with the corresponding reconstructed ones.

Mathematics Subject Classification: 62H35

Keywords: fractals, Hausdorff metric, image comparison, computational geometry

1 Introduction

An interesting class of problems reduces to finding the distance between two sets of points. The manifestations of this class demonstrate the significance of the procedure. For finite sets the "closest-pair-between-sets" problem is solved with the aid of the Voronoi diagram in optimal $\Theta(n \log n)$ time. When the points are the vertices either of a simple or of a convex polygon faster

solutions are expected. Atallah in [2] discusses the Hausdorff distance, which is a well known non-linear operator for comparing discrete sets of points, between two convex polygons and gives an $O(n)$ time algorithm.

What about real-world images? Neither do they have straight-line edges, nor are they convex sets or even Euclidean objects. A useful approach to model these images is using fractal techniques. With these, images are stored as contraction maps, of which the images are approximate fixed points. Images are decoded by iterating these mappings to their fixed points. Subsequently, the problem provides, for example, a means of testing the efficiency of fractal-drawing algorithms by comparing the attractor produced with the actual attractor. Another significant application is in the solution of the so-called "inverse problem" [5] or "image encoding problem" [10], that is, when a given image has to be approximated by a suitable *iterated function system* (IFS) whose number of affine transformations is the least possible and the attractor produced is an acceptable approximation of the given image. An efficient method for comparing digitised images can be applied in pattern recognition, a scientific area where systematic research has lately brought into light a lot of promising results; see for example [8] and the references therein. The interest in the field is huge because of the outstanding applications of pattern recognition in industry, commerce, security, even in everyday life.

The Pythagorean and the city-block metrics have become the two most widely used distances for the specific comparison of digitised images, even though the Hausdorff metric is the most natural in comparing objects (see [6]) in an ideal geometric space. Note that the city-block metric is usually much faster to compute than the Pythagorean and so is sometimes used where speed is critical but accuracy is not too important. The algorithms of Rosenfeld & Pfaltz [14] renewed by Gunilla Borgefors [7] work as approximations of the Euclidean distance (not of the Hausdorff distance) by integer metrics. In particular, they involve local operations that are performed repeatedly on every picture element and its immediate neighbors. Huttenlocher et al. in [13], Rucklidge in [15] and Takács in [20] find the Hausdorff metric attractive for model-based recognition. In the following, we shall describe two new methods for the computation of the Hausdorff metric with respect to the Pythagorean metric between two arbitrary digitised images, one of which is based on the observation that grey-scale images can be viewed as three-dimensional objects. The algorithm of Shonkwiler [17] applies only for black-and-white images with preprocessing using the city-block metric. A linear time minimal memory algorithm using the city-block metric is given in [18]. One should also note that according to J. Stark in [19], where he studies the efficiency of an algorithm for drawing fractal objects, the Hausdorff metric was " . . . not feasible to be evaluated . . ." .

Firstly, after outlining some useful notions, we describe a computational

technique and implement two algorithms for computing the Hausdorff metric; one for black-and-white images and one threshold algorithm, a generalisation of the first one, to include also grey-scale images. To illustrate the performance of the first algorithm, we study its behaviour on monochrome fractal objects. The IFS code used for rendering this class of images is given as well. Next, we address the issue of computing the Hausdorff metric between grey-scale digitised images in an efficient manner, which is based on the observation that grey-scale images can be viewed as three-dimensional objects with their grey scale as the third dimension. Then, to demonstrate the effectiveness of the threshold algorithm and of the proposed 3D technique in their entirety, we apply these methods to find the proximity of an original image to the corresponding filtered or compressed one. Finally, some conclusions are drawn along with a discussion of implementation issues.

2 Objective Quality Measurement

A *metric space* is a set X with a global distance function (the metric ρ) that, for every two points x, y in X, gives the distance between them as a nonnegative real number $\rho(x, y)$. A metric space must also satisfy

$$\begin{aligned} \rho(x,y) &= 0 \text{ if and only if } x = y; \\ \rho(x,y) &= \rho(y,x); \\ \rho(x,z) &\leq \rho(x,y) + \rho(y,z), \end{aligned}$$

where x, y, z are any three elements of X. The nonnegative real number $\rho(x, y)$ is called the *distance* between x and y. The function ρ itself is called a *metric* on the set X. A metric space may be written as a pair (X, ρ), but if the metric is understood, it will be referred to simply as X.

The most important space for us is the familiar n-dimensional Euclidean space $\mathbb{R}^n = \{(x_1, x_2, \ldots, x_n) : x_i \in \mathbb{R}, i = 1, 2, \ldots, n\}$ with the *Pythagorean* or *root mean square error metric* defined by

$$\rho_2(x,y) = \sqrt{\sum_i (x_i - y_i)^2},$$

where $x = (x_1, x_2, \ldots, x_n)$ denotes a typical point defined by its coordinates x_i. Meaningful distances can be calculated in other ways. For instance, the shortest distance from one intersection to another along city streets, assuming a rectangular grid of two-way streets, is a valid distance measure, that is

$$\rho_1(x,y) = \sum_i |(x_i - y_i)|.$$

It is sometimes called the *box*, *city-block* or more justly should be known as *Hippodamean metric*, in honour of Hippodamus (5th cen. B.C.), the Militian architect and town planner, who was the creator of the town-planning system, in which the construction space is subdivided into lattices. Once we have a digitised image (made of pixels), our space (of images) is not $\mathbb{R}^2$ any more and one has to identify a new space X which includes all digitised screens of resolution $M \times M$ pixels.

Let μ be an image and μ_{ij} be the weight of the (i,j)-th pixel, i.e., for a black-and-white image $\mu_{ij} \in \{0,1\}$ whereas in a grey-level one $\mu_{ij} \in \{0,1,\ldots,$ $maxcolours-1\}$, where $maxcolours$ is the number of available colours. The simplest approach is to treat the value μ_{ij} as coordinates of points in $\mathbb{R}^{M^2}$ and to take the distance between two images μ and ν using either the Pythagorean or the city-block metric. This gives respectively:

$$\rho_2(\mu,\nu) = \sqrt{\sum_{(i,j)} (\mu_{ij}-\nu_{ij})^2} = \sqrt{\sum_i \sum_j (\mu_{ij}-\nu_{ij})^2}$$

and

$$\rho_1(\mu,\nu) = \sum_{(i,j)} |\mu_{ij}-\nu_{ij}| = \sum_i \sum_j |\mu_{ij}-\nu_{ij}|.$$

These two metrics have the advantage that, given μ and ν, they are relatively easy to compute and they are applicable to all kinds of images regardless of the application. However, it is well known that they often fall short in predicting the visual (perceptual) quality of an image. By the definition of these two metrics, one may easily deduce that they can be employed both in the case of monochrome and coloured images. A black-and-white image can be viewed as a compact subset in the plane $\mathbb{R}^2$. Yet grey-scale and colour images are more interesting. Colour images are typical extensions of the grey-scale representation of images, since a colour image can be viewed as several grey-scale images, e.g., as a decomposition of *red*, *green* and *blue* channels. Therefore, only grey-scale images will be discussed.

Definition 2.1 *If $\mathcal{H}(X)$ denotes the space whose points are the compact subsets of X other than the empty set, i.e.,*

$$\mathcal{H}(X) = \{\emptyset \neq A \subset X : A \text{ compact}\},$$

then the distance between a point x and a subset $B \in \mathcal{H}(X)$ *is given by*

$$d(x,B) = \min\{\rho(x,b) : b \in B\}. \tag{1}$$

The validity of the above definition depends upon the cited minimum actually existing, but it does, due to compactness. $\mathcal{H}(X)$ is called the "space of fractals

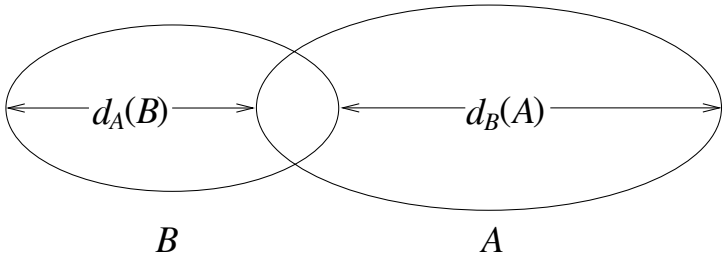

Figure 1: The difference between $d_A(B)$ and $d_B(A)$.

in X" by Barnsley (but note that not all members of $\mathcal{H}(X)$ are fractals). The *Hausdorff distance* between the points A and B of $\mathcal{H}(X)$ is given by

$$h(A, B) = \max\{d_A(B), d_B(A)\},$$

where $d_B(A) = \max\{d(x, B) : x \in A\}$ and $d_A(B) = \max\{d(x, A) : x \in B\}$ (Figure 1). The function $d_A(B)$ sometimes is called the *directed Hausdorff distance* from A to B.

Theorem 2.2 ([3]) *The Hausdorff distance is a metric on $\mathcal{H}(X)$.*

The following theorem is fundamental in the theory of fractals since one might then approximate every fractal object through a sequence of compact subsets of (X, ρ).

Theorem 2.3 ([3]) *(X, ρ) is a complete metric space if and only if $(\mathcal{H}(X), h)$ is a complete metric space.*

Because of the sensitivity of the Hausdorff metric to degradation such as noise and occlusions, some Hausdorff-like metrics have been proposed, such as the *modified Hausdorff distance* (MHD); see [22] and the references therein. The directed MHD from B to A is defined by

$$h_{MHD}(A, B) = \frac{1}{N_a} \sum_{a \in A} d(a, B),$$

where N_a denotes the number of points in A.

3 The 2D Algorithmic Approaches

The use of the Hausdorff metric for binary image comparison and computer vision was originally proposed by Huttenlocher et al. in [13]. As we saw, it is defined upon the set of all nonempty compact subsets of a metric space

(X, ρ) which is denoted by $\mathcal{H}(X)$. In our case, X is the set of the screen pixels and when its resolution is $M \times M$ pixels then X has M^2 elements. On $X = \{(i, j) : i, j \in \{1, 2, \ldots, M\}\}$ the following distance $d_2 \colon X \times X \to \mathbb{R}$ can be defined:

$$d_2((i, j), (i', j')) = \sqrt{(i - i')^2 + (j - j')^2},$$

for all pixels $(i, j), (i', j') \in X$ and $i, j, i', j' \in \{1, 2, \ldots, M\}$.

The distance function d_2 is a metric on X since it is the restriction of the Pythagorean metric on the discrete space X of the screen pixels. Every subset of X consists of a finite collection of pixels and hence it is a compact set. Since the point metric space (X, d_2) is well-defined, the Hausdorff metric is well-defined on the discrete space of screen pixels and, therefore, it suffices to find a way of computing it.

3.1 The Core Algorithm

If a pixel belongs to both objects, then it naturally does not contribute to any distance between our objects μ and ν. Let p be a pixel that belongs to the object ν but does not belong to the object μ. For every such pixel p one should compute the distance of the pixel from the image μ as in Eq. 1; in order to accomplish this we initially take the smallest (digitised) square which is centred at p (see Figure 2) and we examine whether any of the pixels

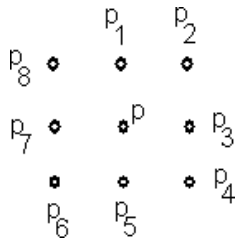

Figure 2: The smallest (digitised) square which is centred at p.

p_1, p_3, p_5 or p_7 belong to the image μ. If this is the case then $d(\{p\}, \mu) = 1$, since $d(\{p\}, A) = \min\{d(\{p\}, y) : y \in A\}$. If this is not the case we go on and examine whether any of p_2, p_4, p_6 or p_8 belong to μ. In such a case $d(\{p\}, \mu) = \sqrt{2}$; this is Stage 1. If the intersection of μ and the smallest square centred at p is the empty set then we focus our attention on the next greater square (always centred at p) which contains 16 pixels (see Figure 3). Again, if any of p_1, p_5, p_9 or p_{13} belong to the image μ, then the distance $d(\{p\}, \mu) = 2$; if none of these belong to μ, then we examine whether this is possible for any of $p_2, p_4, p_6, p_8, p_{10}, p_{12}, p_{14}$ or p_{16}. If one of them is a pixel of μ, then the distance of p from μ is $\sqrt{2^2 + 1}$; otherwise we examine the same hypothesis for p_3, p_7, p_{11} or p_{15} whose distance from p (i.e. $d(\{p\}, \mu)$) is $\sqrt{2^2 + 2^2}$; this is Stage 2. The algorithmic procedure is simplified by the following

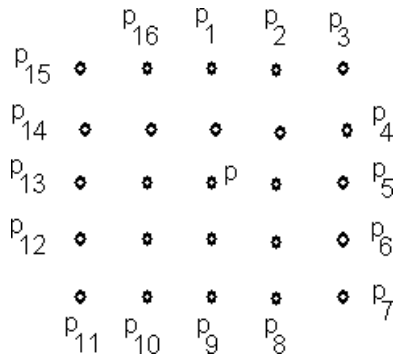

Figure 3: The next greater square centred at p used in Stage 2.

Lemma 3.1 *If* $stage \leq 2$*, then* $stage\sqrt{2} \leq stage + 1$*.*

That is, if a pixel of μ is found in Stage 1 or Stage 2, it is not necessary to look at the next stage for a pixel of μ closer to p, because in Stages $n = 1, 2$ the distance is at most $n\sqrt{2}$, while in the next stage it is at least $n + 1$, which is greater than $n\sqrt{2}$.

In the unfortunate case where none of the above mentioned cases is true, we take the next greater square (centred at p) which corresponds to Stage 3. If we find a pixel that belongs to μ its distance from ν is a candidate *hdist*. The above mentioned process continues to the next stage until the candidate distance computed so far is less than the number of $stage + 1$. This is because the next stage does not contribute towards a better *hdist* than the one already computed. In some cases one might be obliged to take rectangles instead of squares. This happens when one side of the square examined in the previous stage has reached the boundaries of the screen and, hence, cannot become larger towards that side; however, this does not affect the algorithm since there are other directions that the (former) square can be expanded and, hence, become a rectangle.

We follow the same procedure for every pixel p' which belongs to the image μ but does not belong to the image ν. Since we are always going to compare nonempty finite sets, the algorithm is bound to terminate in a finite number of steps.

There are two major factors which affect the number of steps executed before the termination of the algorithm (and, hence, the time requirements of the algorithm): the first one is the size of the images compared; the "smaller" the objects the faster the comparison takes place. The second one is the similarity between the images compared; the more similar the images are, the faster the algorithm terminates. This happens because pixels of the screen that do not belong to either image do not affect the Hausdorff metric between the objects and therefore are not involved in the process.

The theoretical basis behind the proposed algorithm is that the Hausdorff metric can be equivalently defined as

$$d_H(\mu,\nu) = \max\{\max\{d(\{p\},\nu) : p \in \mu\}, \max\{d(\{p\},\mu) : p \in \nu\}\}.$$

Since we found a way to calculate the distances $d(\{p\},\nu), p \in \mu$ and $d(\{p\},\mu), p \in \nu$, then the Hausdorff distance is easily computed.

An analytic formula for the computation of $d(\{p\},\mu)$ is the following:

$$d(\{p\},\mu) = \begin{cases} 0, & \text{if } p \in \mu \\ 1, & \text{if } (i_p \pm 1, j_p) \text{ or } (i_p, j_p \pm 1) \in \mu \text{ and } p \notin \mu \\ \sqrt{2}, & \text{if } (i_p \pm 1, j_p \pm 1) \in \mu \text{ and all the above are false} \\ 2, & \text{if } (i_p \pm 2, j_p) \text{ or } (i_p, j_p \pm 2) \in \mu \text{ and all the above are false} \\ \sqrt{5}, & \text{if } (i_p \pm 2, j_p \pm 1) \text{ or } (i_p \pm 1, j_p \pm 2) \in \mu \text{ and all the above are false} \\ \sqrt{8}, & \text{if } (i_p \pm 2, j_p \pm 2) \in \mu \text{ and all the above are false} \\ 3, & \text{if } (i_p \pm 3, j_p) \text{ or } (i_p, j_p \pm 3) \in \mu \text{ and all the above are false} \\ \sqrt{10}, & \text{if } (i_p \pm 3, j_p \pm 1) \text{ or } (i_p \pm 1, j_p \pm 3) \in \mu \text{ and all the above are false} \\ \sqrt{13}, & \text{if } (i_p \pm 3, j_p \pm 2) \text{ or } (i_p \pm 2, j_p \pm 3) \in \mu \text{ and all the above are false} \\ \sqrt{18}, & \text{if } (i_p \pm 3, j_p \pm 3) \in \mu \text{ and all the above are false} \\ \vdots & \end{cases}$$

or, more compactly,

$$d(\{p\},\mu) = \begin{cases} 0, & \text{if } p \in \mu \\ \sqrt{k^2+l^2}, & \text{otherwise, where } k \geq 1 \text{ and } 0 \leq l \leq k \text{ are the smallest natural} \\ & \text{numbers such that } (i_p \pm k, j_p \pm l) \text{ or } (i_p \pm l, j_p \pm k) \in \mu. \end{cases}$$

This algorithm involves $O(h^2 M \cdot N)$ operations for processing $M \times N$ images which turn out to have Hausdorff distance h.

3.2 A Threshold Algorithmic Extension for Grey-scale Images

By its definition, the Hausdorff metric can be used only for binary image comparison. Now that black and white binary images have been compared, this subsection will address the issue of computing the Hausdorff metric between grey-scale images. It is well-known that a grey-scale image can be modelled as a function $f(x,y)$ that gives the grey level at each point (x,y). In accordance to the previous black and white image representation as a compact subset of the plane $\mathbb{R}^2$, a third dimension is introduced for the colour information. Hence X can now be viewed as a compact subset of $\mathbb{R}^3$.

This generalisation is not as easy as it seems. It is impossible to expect an exact colour matching between the original and the compressed picture. In order for our algorithm to represent the human perception of visual similarity a tolerance must be introduced, since the human eye is not that sensitive to slight color variations. Another issue is the termination of our algorithm. It is

possible that no colour matching may be found inside the predefined tolerance; thus, the design of the algorithm must cater for this case after the termination of the search. Also, we must account for possible degenerate cases, such as to compare a black with a white image, or objects with only one pixel.

To identify the nearest pixel (colour) matching the original one, we assume that a pixel $p = (x, y, t) \in \mathbb{Z}^3$ belongs to both objects, if $|t - t'| \leq \tau$, for some preset threshold τ, where $p' = (x, y, t')$ is the corresponding pixel of the second image. Whilst scanning the square, each time the current directed Hausdorff distance is stored. So, in the unfortunate case where the appropriate approximate match is not found, the Hausdorff metric is the Haudorff distance from the farthest (border) pixel. Note that, for $\tau = 0$, we have the Hausdorff metric for black-and-white images addressed in Subsection 3.1.

4 Application on Fractal Sets

Within Fractal Geometry, the method of *iterated function systems* introduced by Hutchinson [12], and popularised by Barnsley [4] and Demko et al. [9], provides a framework for encoding and generating a large class of fractal images.

The easiest method for rendering a fractal image is with the aid of the *Random Iteration Algorithm*, or *RIA* for short. In the RIA we calculate at each stage just one new point x_{n+1} from its predecessor x_n, by $x_{n+1} = w_i(x_n)$ with a randomly chosen i from $1, 2, \ldots, N$. Given a large number of iterations—say 100,000, it works. We refer the interested reader to [3] or [11].

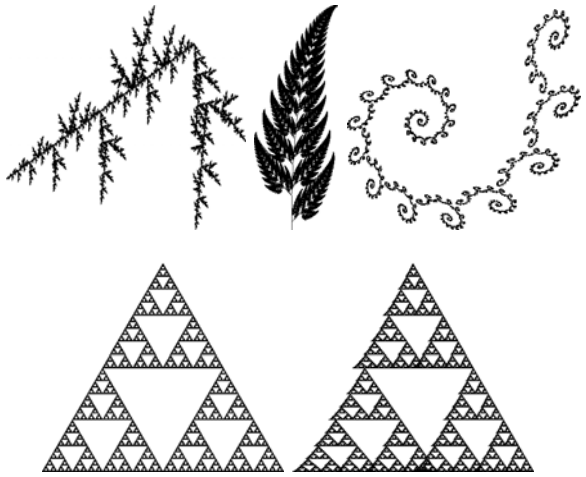

Figure 4: A dendrite, a fern leaf, a spiral, the Sierpiński triangle and a transformed version of it.

The IFS codes for the under-examination fractal objects are presented in Tables 5–8. The corresponding fractal images are illustrated in Figure 4. The

size of the fractal images considered is 256×256 pixels and the number of iterations is 100,000. Table 1 shows the results of our algorithm with regard to the city-block, the Pythagorean, the MHD and the Hausdorff distance between the fractal attractors, as well as the CPU (computation) time t_H in min:sec format of the fourth distance. The computation time of the city-block and the Pythagorean distance is negligible.

Table 1: The city-block, Pythagorean, MHD and Hausdorff distance between the fractal objects and the computation times in min:sec format of the fourth distance.

μ, ν	d_1	d_2	d_{MHD}	d_H	t_H	t'_H
dendrite, spiral	1,719,054	21,060	0.9378	56	00:31	00:10
leaf, spiral	2,286,396	24,287	2.0271	60	01:32	00:30
dendrite, leaf	2,352,186	24,635	1.6558	70	01:16	00:26
dendrite, triangle	3,141,150	28,468	3.0325	82	01:38	00:33
triangle, spiral	2,686,554	26,327	3.3501	90	01:43	00:35
leaf, triangle	3,583,104	30,405	4.5930	121	02:42	00:56

Looking at the values of all the distances we observe that the most dissimilar objects are the {leaf, triangle} pair, whereas the most similar ones are the {dendrite, spiral} pair. The former pair has the slowest computation time, while the latter the fastest. This confirms our observations in Subsection 3.1, that the more similar the images are the faster our algorithm terminates. The strength of the Hausdorff metric versus the other metrics for comparing objects can be seen in the comparison of the pairs {dendrite, triangle} and {triangle, spiral}. Initially looking at their city-block and Pythagorean distances, one can be easily led into misperceiving that the latter pair contains the more similar objects; but their Hausdorff distance is smaller for the first pair, which shows the correct discrepancy between the two objects. Compare also the distances between the Sierpiński triangle and the transformed version of it that arises if we slightly alter its IFS code; taking for example $a_1 = 0.6$, then $d_1 = 2,206,416$, $d_2 = 23,859$, $d_{MHD} = 0.3610$ and $d_H = 18$. Although d_1 and d_2 suggest, after looking at Table 1, that the two triangles are very different, d_H suggests that they are very close to each other, which is the case.

Another thing that must be mentioned is that the Hausdorff distance is more stable as compared with the other three metrics. The reason for this is because its value is independent of the random distribution of the points which form the attractor. In contrast, the other two metrics depend on the density of the points which form the invariant measure of the distribution. So, it is possible two consecutive runnings of the RIA to give us different values for the d_1, d_2 and d_{MHD} (but consistent concerning the order), whereas the d_H will be almost the same, with deviation ± 1 unit.

5 The 3D Approach: A Generalised Algorithm for Grey-scale Images

In this section we present a new method for computing the Hausdorff metric between two grey-scale digitised images, by considering them as 3-dimensional objects. In our case, X is the set of the screen pixels together with their grey levels and, when its size is $M \times N \times P$ bits, X has $M \cdot N \cdot P$ elements, where P is the maximum grey level; hence $X = \{(i,j,k) : i \in \{1,2,\ldots,M\}, j \in \{1,2,\ldots,N\}, k \in \{0,1,\ldots,P-1\}\}$. On X the following distance d_2: $X \times X \to \mathbb{R}$ can be defined:

$$d_2((i,j,k),(i',j',k')) = \sqrt{(i-i')^2 + (j-j')^2 + (k-k')^2}, \tag{2}$$

for all pixels $(i,j,k),(i',j',k') \in X$.

The distance function d_2 is a metric on $X \subset \mathbb{R}^3$ since it is the restriction of the Pythagorean metric on the discrete space X of the screen pixels. Every subset of X consists of a finite collection of pixels and hence it is a compact set. Since any discrete metric space is complete, the Hausdorff metric is well defined on the discrete space of screen pixels. The algorithm we propose for computing it is the following.

Let $p' = (i,j,k') \in \mu$ and $p = (i,j,k) \in \nu$. If a pixel belongs to both objects, i.e., if $k = k'$, then it naturally does not contribute to any distance between our objects μ and ν. Let p be a pixel that belongs to the object ν but does not belong to the object μ, i.e., $k \neq k'$. Then $|k-k'|$ is stored as a candidate distance, i.e., we set as *TempHausdorff* $= |k-k'|$. For every such pixel p one should compute the distance of the pixel from the image μ according to Eq. (1); in order to accomplish this we initially take the smallest (digitised) square which is centred at p and we examine whether its distance from any of its eight neighbouring pixels is less than *TempHausdorff*; in such a case we assign the new value to *TempHausdorff* and the first examination level is completed. Furthermore, we take the next greater square (centred at p) and we continue until either we reach the boundary of the image or, at some examination level, $d_2((i,j,k),(i',j',k')) >$ *TempHausdorff*. We follow the same procedure for every pixel p' which belongs to the image μ but does not belong to the image ν. The maximum value of *TempHausdorff* that results from both procedures mentioned above is the actual Hausdorff distance. Since we are always going to compare nonempty compact sets, the algorithm is bound to terminate in a finite number of steps. Table 1 also shows the CPU (computation) time t'_H in min:sec format of the 3D Hausdorff distance mentioned above; it is almost one third of t_H!

When X is a bounded subset of $\mathbb{R}^3$, it has an obvious interpretation as a grey-scale image and can be used to yield colour images either by a suitable colour map coding or by the superimposition of three such measures. We use

the Hausdorff distance to examine the effect of a compression method on the original image.

6 Application on Grey-scale Images

Most digital-image-compression methods required in real-world applications are lossy compression methods, i.e., the decompressed images must be visually similar, not necessarilly identical, to the original images. So, there is always a tradeoff between the compression ratio and the reconstruction error. We now present typical results from the application of our algorithm to real imagery, aiming to demonstrate its applicability to the demanding problems inherent in the image compression area and its performance. The original images used as our reference point in the experiments presented here are the $256 \times 256 \times 8$ bits Lena and Barbara images shown in Figure 5.

Figure 5: The original images of Lena (left) and Barbara (right) used in our experiments ($256 \times 256 \times 8$ bits).

We examine for each original image how close it is to a filtered or compressed replica of it. In other words we seek to measure the difference (i.e. the error) between two images by computing the Hausdorff distance between the original image and each of the associated filtered ones.

The filter banks used in our simulations include the *lifting scheme* (Figure 6), which represents a generic and efficient solution to the perfect inversion problem, the Haar filter as well as the (9,7) pair developed by Antonini et al. [1] (Figure 7). We have also tested the performance of our scheme to a Joint Photographic Experts Group (JPEG) codec in its Corel 7 implementation, as applied to our test images and to images compressed with the EZW method [16]. The *Embedded Zerotree Wavelet*, or *EZW* for short, is a reasonable algorithm in modern image compression to use for experiments (see

Figure 6: $256 \times 256 \times 8$ bits test image used in our experiments (lifting scheme is used).

[21]). The EZW algorithm is a particularly effective approach to the following two-fold problem: achieving the best image quality for a given compression ratio (bit-rate) and encode the image in such a way that all lower bit-rate encodings are embedded in the beginning of the final bit stream. The symbol stream generated by EZW is entropy (losslessly) encoded to achieve further compression.

Table 2: The city-block, Pythagorean and Hausdorff distance between the $256 \times 256 \times 8$ bits 3D real-world images and the computation time in min:sec format of the third distance.

	d_1	d_2	d_H	t'_H
μ, μ_1	0	0	0	00:00
μ, μ_2	1,082	34	2	00:00
μ, μ_3	8,657	95	2	00:00
μ, μ_4	300,320	1,669	27	00:04
μ, μ_5	366,095	2,002	31	00:06
ν, ν_1	694,055	4,229	46	00:09
ν, ν_2	893,458	4,701	55	00:19

For the image of Lena we used the filter banks as well as the compression schemes mentioned above. For the image of Barbara we used only the compression schemes. Figure 8 shows compressed images of Lena at a ratio of 32:1 using (9,7) Discrete Wavelet Transform (DWT) combined with Run-Length Encoder (RLE) and JPEG coding respectively. Figure 9 shows compressed images of Barbara at a ratio of 64:1 using (9,7) DWT combined with RLE and

(a) (b)

Figure 7: $256 \times 256 \times 8$ bits test images used in our experiments ((a) Haar and (b) Antonini (9,7) filters are used).

Table 3: The Hausdorff distance d_H between the $256 \times 256 \times 8$ bits real-world images with respect to different values of the threshold τ.

$d_H \backslash \tau$	0	5	10	15	20	25	30
μ, μ_1	0	0	0	0	0	0	0
μ, μ_2	56	0	0	0	0	0	0
μ, μ_3	160	0	0	0	0	0	0
μ, μ_4	362	362	60	26	25	24	24
μ, μ_5	362	362	362	362	362	362	25
ν, ν_1	362	362	362	362	96	75	75
ν, ν_2	362	362	362	362	362	109	106

JPEG coding respectively.

Table 2 shows the city-block and the Pythagorean distances between the real-world images. Table 3 shows the results of the threshold algorithm with regard to the Hausdorff distance between the real-world images with respect to the threshold value. Table 4 shows the CPU (computation) time of the third distance in min:sec format with respect to different values of the threshold. The correspondence between the images of Lena and the indices is the following: μ = original image, μ_1 = lifting scheme, μ_2 = Antonini filter, μ_3 = Haar transform, μ_4 = 32:1 compression and μ_5 = JPEG compression. The correspondence between the images of Barbara and the indices is the following: ν = original image, ν_1 = 64:1 compression and ν_2 = JPEG compression.

There are two ways to examine Tables 2–4: vertically and horizontally. Looking at Tables 2 and 3 from top to bottom we can see, which of the im-

(a) (b)

Figure 8: $256 \times 256 \times 8$ bits test images used in our experiments ((a) EZW Shapiro (9,7) and (b) JPEG compression are used).

Table 4: The computation time t_H in min:sec format of the Hausdorff distances between the $256 \times 256 \times 8$ bits real-world images in relation with different values of the threshold τ.

$t_H \backslash \tau$	0	5	10	15	20	25	30
μ, μ_1	00:00	00:00	00:00	00:00	00:00	00:00	00:00
μ, μ_2	00:00	00:00	00:00	00:00	00:00	00:00	00:00
μ, μ_3	00:11	00:00	00:00	00:00	00:00	00:00	00:00
μ, μ_4	01:30	00:07	00:02	00:01	00:01	00:01	00:01
μ, μ_5	05:22	00:39	00:15	00:06	00:02	00:01	00:01
ν, ν_1	03:17	00:44	00:24	00:13	00:08	00:05	00:03
ν, ν_2	21:27	03:05	01:41	01:07	00:33	00:05	00:03

ages are closer to the originals. Some extreme cases are possible because, by definition, the Hausdorff metric is not robust under noise. A single "wrong" pixel can make it very high. In this case, it is better to view the table from left to right in order to conceive the total behaviour of the distance via the increasing value of the threshold used for these degenerate cases. Table 4, which shows the computation times, is more consistent because the extreme cases are not capable of affecting the whole time spent for computing the particularly distances. Hence, the computation time can be considered as an empirical measure for the similarity of the compared images.

Table 2 also shows the results of the generalised algorithm with regard to the Hausdorff distance between the real-world images and its CPU (computation) time in min:sec format. Looking at all the pictures of Lena and Barbara it is not possible to observe at once all of their differences or defects, except for the figures compressed using JPEG. With the help of the Hausdorff distance it

(a) (b)

Figure 9: $256 \times 256 \times 8$ bits test images used in our experiments ((a) EZW Shapiro (9,7) and (b) JPEG compression are used).

is now possible to look "behind" these images and unveil their imperfections. Looking at Table 2 from top to bottom we can see, which of the images are closer to the originals. Compare also the computation times in Table 4 (for $\tau = 0$) with those in Table 2. The generalised algorithm is really faster!

7 Conclusions and Extensions

The current implementation of our algorithms is developed using Microsoft Visual Basic 6.0 and is capable of drawing two fractal, or open two bitmapped, images as well as to compute their city-block, Pythagorean, Hausdorff and modified Hausdorff metrics. The CPU time needed for their computation is also displayed. There exist also a window in which one can see the discrepancy between the two under comparison images [1]. The whole algorithm was finally tested and rated by computing the distance between attractors produced using the RIA, as well as between grey-scale original and transformed real-world images. Time results are given in CPU minutes on an Intel(R) Pentium(R) M processor 1700 MHz 1.69 GHz, 512 MB of RAM running Microsoft Windows XP Professional Version 2002 SP 2.

In the case of the fractal images the non-blank (the non background color) pixels (of the fractal image) are *known* due the method used for their construction out of the IFS code. This observation, which holds for both images, heavily simplifies the number of calculation required for the computation of the Hausdorff distance. More analytically, for the computation of $d_A(B)$ we take into account only the non-blank points of image A. Moreover, during the 'spiral'

[1] Please visit cgi.di.uoa.gr/∼vasilios/Hausdorff.exe

process for the computation of $d(x, B)$ we only take into account the non-blank pixels of image B. Conversely, during the computation of $d_B(A), d(y, A)$ we disregard the pixels of B and A, respectively, that are blank (i.e. not colored). As it is evident, the aforementioned procedure can be applied in both black & white and grey-scale fractal images.

In the case of digitized images that do not come from IFS code, (e.g. real world images), we are mainly interested in measuring their similarity. While comparing similar images, our algorithm efficiently computes their Hausdorff distance. In the opposite case of dissimilar images, our algorithm quickly yields a lower bound of their Hausdorff distance from the very first iterations; this bound can be considered as a measure of their dissimilarity helping us to quickly judge that the two pictures are not alike.

When our algorithms for the computation of the Hausdorff metric are implemented using conventional single-processor machines, their computation time is close to the one needed for the computation of the Pythagorean or the city-block distance, which is fairly satisfactory (see Table 2 or Table 4). This result holds under the proviso that relatively "close" images are compared, as in the case of the Collage Theorem for fractal objects. The same result holds also for the grey-scale real-world images and is the case for the μ_1, μ_2 and μ_3 images as compared with the original image of Lena. The proposed generalised method can be extended to images with more than one channels, such as RGB or CMYK. In each such instance, it suffices to add one extra dimension to our space X for each channel.

Appendix

A transformation w is *affine*, if it may be represented by a matrix A and translation $\mathbf{t}$ as $w(\mathbf{x}) = A\mathbf{x} + \mathbf{t}$, or (if $\mathbf{x} \in \mathbb{R}^2$)

$$w \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d \\ e \end{bmatrix}.$$

The *code* of w is the 6-tuple (a, b, c, s, d, e), and the *code of an IFS* is a table whose rows are the codes of $w_1, w_2, \ldots, w_N$. If we add one extra column with the corresponding probabilities $p_1, p_2, \ldots, p_N$, then we are talking about the *code of an IFS with probabilities*.

We list the IFS codes (see Tables 5–8) for the examples discussed in the main text.

ACKNOWLEDGEMENTS. The authors wish to thank Aliki Vassilarakou, Ph.D. student of the Department of Informatics & Telecommunications, for her help in the compressed images presented here.

Table 5: The IFS code for a dendrite.

w	a	b	c	s	d	e	p
1	0.5	0	0	0.5	0.0625	0.15	0.25
2	0.21	-0.20625	0.528	0.21	0.789	0	0.5
3	0.5	0	0	0.5	0.375	0.375	0.75
4	-0.2	0.1125	-0.288	-0.2	0.609	0.975	1

Table 6: The IFS code for a fern leaf.

w	a	b	c	s	d	e	p
1	0	0	0	0.16	0.5	0.07	0.001
2	0.2	-0.195	0.3066667	0.22	0.41625	0.045	0.071
3	-0.15	0.21	0.3466667	0.24	0.5575	-0.07333	0.141
4	0.85	0.03	-0.5333	0.85	0.07249999	0.1725	1

References

[1] M. Antonini, M. Barlaud and P. Mathieu, Image coding using wavelet transform, *IEEE Trans. Image Process.,* **1** (1992), 205–220.

[2] M.J. Atallah, A linear time algorithm for the Hausdorff distance between convex polygons, *Info. Proc. Lett.,* **8** (1983), 207–209.

[3] M.F. Barnsley, *Fractals everywhere,* 2nd ed., Academic Press Professional, San Diego, 1993.

[4] M.F. Barnsley and S. Demko, Iterated function systems and the global construction of fractals, *Proc. Roy. Soc. London, Ser. A,* **399** (1985), 243–275.

[5] M.F. Barnsley, V. Ervin, D. Hardin and J. Lancaster, Solution of an inverse problem for fractal and other sets, *Proc. Nat. Acad. Sci. U.S.A.,* **83** (1986), 1975–1977.

[6] E. Belogay, C. Cabrelli, U. Molter and R. Shonkwiler, Calculating the Hausdorff distance between curves, *Inform. Process. Lett.,* **64** (1997), 17–22.

Table 7: The IFS code for the Sierpiński triangle.

w	a	b	c	s	d	e	p
1	0.5	0	0	0.5	0	0	0.33
2	0.5	0	0	0.5	0.5	0	0.67
3	0.5	0	0	0.5	0.25	0.433	1

Table 8: The IFS code for a spiral.

w	a	b	c	s	d	e	p
1	-0.18	0.126	-0.2571	-0.18	0.815	0.8485	0.05
2	-0.8	0.4	-0.4	0.8	-0.088	0.2514	1

[7] G. Borgefors, Distance transformations in digital images, *Computer, Vision, Graphics and Image Processing,* **34** (1986), 344–371.

[8] B.B. Chandhuri and A. Rosenfeld, A modified Hausdorff distance between fuzzy sets, *Inform. Sci.,* **118** (1999), 159–171.

[9] S. Demko, L. Hodges and B. Naylor, Construction of fractal objects with iterated function systems, *Comput. Graph.,* **19** (3) (1985), 271–278.

[10] Y. Fisher, editor, *Fractal image compression,* Springer-Verlag, New York, 1995.

[11] S.G. Hoggar, *Mathematics for computer graphics,* Cambridge Univ. Press, London and New York, 1992.

[12] J.E. Hutchinson, Fractals and self similarity, *Indiana Univ. Math. J.,* **30** (1981), 713–747.

[13] D.P. Huttenlocher, G.A. Klanderman and W.J. Rucklidge, Comparing images using the Hausdorff distance, *IEEE Trans. Pattern Anal. Machine Intelligence,* **15** (1993), 850–863.

[14] A. Rosenfeld and J. Pfaltz, Distance functions in Digital Pictures, *Pattern Recognition,* **1** (1968), 33–61.

[15] W. Rucklidge, *Efficient visual recognition using the Hausdorff distance,* Springer-Verlag, Berlin and Heidelberg, 1996.

[16] J.M. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. Signal Process.,* **41** (1993), 3445–3462.

[17] R. Shonkwiler, An image algorithm for computing the Hausdorff distance efficiently in linear time, *Inform. Process. Lett.,* **30** (1989), 87–89.

[18] R. Shonkwiler, Computing the Hausdorff set distance in linear time for any L_p point distance, *Inform. Process. Lett.,* **38** (1991), 201–207.

[19] J. Stark, Iterated function systems as neural networks, *Neural Networks,* **4** (1991), 679–690.

[20] B. Takács, Comparing face images using the modified Hausdorff distance, *Pattern Recognition,* **31** (1998), 1873–1881.

[21] M. Vetterli and J. Kovačevič, *Wavelets and subband coding,* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[22] C. Zhao, W. Shi, and Y. Deng, A new Hausdorff distance for image matching, *Pattern Recognition Lett.,* **26** (2005), 581–586.

Received: July 13, 2006