FAST SHAPE MATCHING USING THE HAUSDORFF DISTANCE

[1]Paweł Rotter, [2]Andrzej M.J. Skulimowski, [3]Constantine Kotropoulos and [3]Ioannis Pitas

[1]rotter@agh.edu.pl

[1, 2]AGH-University of Science and Technology in Cracow, al. Mickiewicza 30, Kraków, Poland
[3]Aristotle University of Thessaloniki, Box 451, 54124 Thessaloniki, Greece

ABSTRACT

In this paper we provide a series of properties that simplify the computation of the Hausdorff distance between graphical image objects represented by sets of pixels. We propose a simple test which allows for a given set of pixels to check whether it is sufficient to compute the Hausdorff distance using only the boundary pixels. Next, we present a method that allows to prune a part of the contour. We discuss the possibility of combining the proposed approach with other methods to speed-up the computations The experimental results presented demonstrate a considerable reduction of the computation time.

NOTATION

Θ – family of non-empty, closed and bounded subsets of a metric space
A, B – elements of Θ
cl(A) – closure of A
∂ A – boundary of A
int(A) – interior of A
comp(A) – complement of A
$A^{D(r)}$ – dilation of A with a ball with radius r
k(x, r), K(x, r) – open and closed ball with centre x and radius r

1. INTRODUCTION

The Hausdorff distance between two sets A, B $\in \Theta$ is defined as:

$$d_H(A, B) = \max \{ d_{H+}(A,B), d_{H-}(A,B) \}, \quad (1)$$

where

$$d_{H+}(A, B) = \max \{d(a, B) : a \in A\} \quad (2)$$

$$d_{H-}(A, B) = \max \{d(b, A) : b \in B\}, \quad (3)$$

$$d(v, W) = \min \{d(v, w) : w \in W\}. \quad (4)$$

The Hausdorff distance is very useful as a dissimilarity measure between graphical objects. Unlike the feature space methods (e.g. moment invariants, shape coefficients), the Hausdorff distance between two closed sets is zero if and *only if* both sets are identical. Moreover, any object transformation (like translation, rotation, scaling, or perspective projection) can be taken into consideration by searching for the minimum of the Hausdorff distance between original object and the transformed one over the space of transformation parameters. The values of transformation parameters which ensure the best matching can then be used for high-level recognition based on the relations between the elements of an image. Another important advantage of the Hausdorff distance is the possibility of using separately d_{H+} and d_{H-} which are dissimilarity measures between one object and a part of another: $d_{H+}(A, B) = 0$ iff $A \subset B$ and $d_{H-}(A, B) = 0$ iff $B \subset A$. The separate consideration of d_{H+} and d_{H-} is very useful when the segmentation is not reliable or when the object can be partially occluded [1], [2]. Another possibility of coping with partial occlusions is the application of the Hausdorff distance fraction, i.e. substitution for supremum in (2) and (3) by a quantile. Good results for noisy images are reported in [3].

The main disadvantage of the Hausdorff distance is its computational burden. For a pair of objects represented by sets of pixels, in a naive approach, the distance between every pair of pixels $(a,b) \in A \times B$ should be computed. During the last decade several methods for speeding-up the computations have been proposed. Some of them are based on the approximation of objects by polygons. For example, a fast and simple linear-time algorithm for convex polygons is described in [4]. A more general method for arbitrary polygons is presented in [5].

The aforementioned methods give only an approximation of the Hausdorff distance but there other methods that yield an exact result. The Voronoi diagram of the set B allows to find minimal distance from any point to set B in time O(log|B|) where |B| is the

cardinality of the set B [6]. A very effective method of the Hausdorff distance calculation is the application of a distance transform, which pre-calculates the distances between every point of a specified ε-lattice and a model/query object and stores them in a distance matrix. However, this approach suffers from certain drawbacks like the long pre-computation time and the high memory requirements. Moreover, the size of distance matrix is fixed and if any part of the object lies outside of the matrix, the method does not work. A generalisation of the Hausdorff distance that exploits the edge direction information [7] requires a 3D distance transform so combining these two methods is not always profitable [8].

It is worth noting that there exist methods to speed-up the computations not only for a single pair of objects at fixed position but also at further stages of Hausdorff-distance-based matching, as shown in Table 1. When we are seeking for a minimum of the Hausdorff distance between one object and geometrical transformation of another, big parts of a transformation parameters space can be excluded from computations [1]. At the highest level of the matching process (see Table 1) metric properties of the Hausdorff distance can be used for a fast database search. Having found the distances between the model objects off-line, we can apply the triangle inequality in model object space and rule out a part of the database from computations [9]. The matrix of distances between the models can also be used for determining the order of database searching which yields quickly the model closest to the query object [10], [2].

The method proposed here aims at the reduction of computations in the stage where maximum operations occur, a point that was not addressed in the related literature. In Sec. 2 we discuss on the condition that allows to discard the interior of a set, i.e. when the value of the Hausdorff distance does not change by replacing a set by its boundary. We propose a test based on morphological operations that allows to check the necessary and sufficient condition for matching a given pixel set A with arbitrary pixel set B by employing only the boundary of A.

In Sec. 3 we propose a method for further computational complexity reductions. It is designed especially for contours of coherent 2-D objects represented by finite sets of pixels, so it can be applied as a next step of the method presented on Sec. 2. Since sets A and B are contours of graphical objects, they can be ordered so that every two consecutive pixels are neighbours in an 8-connected ε-lattice. Therefore the distance between consecutive pixels has an upper bound and cannot be greater than $\varepsilon\sqrt{2}$. The method presented in Sec. 3 allows to calculate the exact value $d_{H+}(A, B)$ without resorting to d(a, B) for every point $a \in A$. This means that some parts of set A can be discarded in computations.

Table 1. *Locating the proposed method in the hierarchy of the complexity reduction methods that are appropriate for the several stages of the matching process.*

Stage of the matching process	**Method for complexity reduction**		
Find the best model in the database	d. Pruning part of the database		high stage
Find the transformation that yields the best matching	c. Pruning part of the search space		
Computation of maximum: max {d(b, A): b∈B}	**b. Proposed method**		
Computation of minimum: min {d(b, a): a∈A}	a. Voronoi diagram	a'. Distance transform	low stage

As shown in Table 1, our approach can be combined with existing ones, since it works at a different stage of the hierarchy of computations.

2. CONTOUR-BASED MATCHING WITH THE HAUSDORFF DISTANCE

In this section we discuss when the Hausdorff distance between two sets is equal to the Hausdorff distance between their contours. As shown in [11] for non-empty and closed sets their interior can always be discarded if both sets are convex but it is not a necessary condition. In this section we present a necessary and sufficient condition when for a given set A holds: $d_{H-}(A, B) = d_{H-}(A, \partial B)$ for any set B and when for a given set B holds: $d_{H+}(A, B) = d_{H+}(\partial A, B)$ for any set A. The condition can be checked for every model object in the database at a stage of the database creation and for a query object before the process of its recognition. If the condition holds, we can discard the interior of sets and apply the method proposed in the Sec. 3 for a further computation reduction.

2.1. Necessary and sufficient condition for computing the Hausdorff distance between two sets using only their boundaries.

Property 1

A set $A \in \Theta$ satisfies *Property 1* if for any open ball $k(x_0,r)$ such that $k(x_0,r) \cap A = \varnothing$ a curve S exists such that:

P1. $x_0 \in S$

P2. $\forall \eta>0 \quad \exists\, x \in S: d(x, A) > \eta$ (i.e. S "goes to infinity")

P3. $\forall\, x \in S: k(x, r) \cap A = \varnothing$.

In other words, if *Property 1* holds for a set, it guarantees that every ball can be "taken out" from every

hollow in A as is depicted in Figure 1. Figure 2 demonstrates a set that violates *Property 1*.

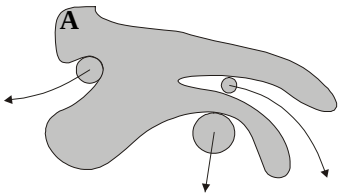


Figure 1. *Example of a set for which Property 1 holds.*

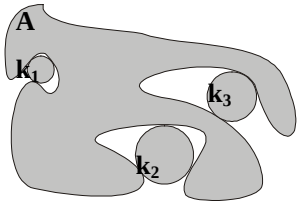


Figure 2. *Example of a set which violates Property 1.*

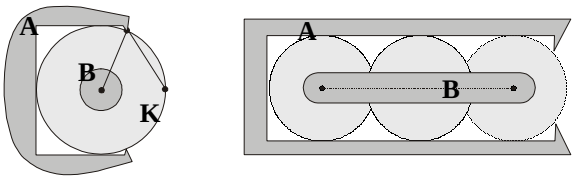
Proposition 1
For a given set $A \in \Theta$: $d_{H+}(B, A) = d_{H+}(\partial B, A)$ for every $B \in \Theta$ if and only if *Property 1* holds for A.

Proof
1. A satisfies *Property 1* $\Rightarrow \forall$ B: $d_{H+}(B, A) = d_{H+}(\partial B, A)$
   We will show that if A satisfies *Property 1*, $\forall\, b \in \text{int}(B)\ \exists\, b' \in \partial B$: $d(b', A) \geq d(b, A)$. The open ball k(b, d(b, A)) has an empty intersection with A. Based on Property 1 for the set A, there exists a curve S=S(k(b, d(b, A)) that fulfils P1-P3. The set B is bounded, while the curve S is not (as it fulfils P2), so $S \cap \partial B \neq \emptyset$. Let us denote by b' any point belonging to $S \cap \partial B$. The curve S fulfils P3, so: $d(b', A) \geq d(b, A)$.

2. A does not satisfy *Property 1* $\Rightarrow \exists$ B: $d_{H+}(B, A) \neq d_{H+}(\partial B, A)$
   If the set A does not satisfy *Property 1*, there exists a ball $k(x_0, r)$ such that no curve $S(k(x_0, r))$ fulfils P1-P3. Let us denote by X the set of all points that can be connected with x_0 by a curve that fulfils P3. For all $x' \in \partial X$: $d(x', A)=r$. Let us denote by $A^{D(\varepsilon)}$ the dilation of A with a ball with radius ε. There exists $\varepsilon > 0$ such that $\forall\, x'' \in \partial X^{D(\varepsilon)}$: $d(x'', A) < r$, so the set $X^{D(\varepsilon)}$ is an example of such a set B, that: $d_{H+}(B, A) > d_{H+}(\partial B, A)$.

♦

Let us look at Figure 2. For any closed ball $K_n = k_n \cup \partial k_n$ holds: $d_{H+}(K_n, A) \neq d_{H+}(\partial K_n, A)$ and the point argmax d(b, A): $b \in K_n$ is located just in the centre of the ball. We could ask if it is a rule, i.e. if for any set A any closed ball K "blocked" in its hollow (more precisely: closure of "blocked" open ball) can serve as an example of the set such that $d_{H+}(K, A) \neq d_{H+}(\partial K, A)$. The answer is negative, in examples shown in Figure 3 $d_{H+}(K, A)$ is equal to $d_{H+}(\partial K, A)$. However, according to Proposition 1, a set B such that $d_{H+}(B, A) \neq d_{H+}(\partial B, A)$ exists. In both cases shown in Figure 3, B is such a set.



a b

Figure 3. *For a ball K "blocked" in a hollow in A: $d_{H+}(K, A) = d_{H+}(\partial K, A)$ but for B: $d_{H+}(B, A) \neq d_{H+}(\partial B, A)$.*

2.2. Test for validation *Property 1* and its application for shape matching

Next, we prove a proposition that allows to check if the given set A satisfies *Property 1*.
Let us denote by $A^{D(r)}$ the dilation of the set A with a ball with radius r as a structuring element, i.e.:
$A^{D(r)} = \{x: K(x,r) \cap A \neq \emptyset\}$, in other words: $\text{int}(A^{D(r)}) = \{x: k(x,r) \cap A \neq \emptyset\}$.

Proposition 2
A set $A \in \Theta$ satisfies *Property 1* if and only if for any r the set $\text{comp}(A^{D(r)})$ is connected.

Proof
1. $\forall$ r: $\text{comp}(A^{D(r)})$ is connected $\Rightarrow$ A has *Property 1*
   Let us take any ball $k(x_0,r)$ such that $k(x_0,r) \cap A = \emptyset$. It means that $x_0 \notin \text{int}(A^{D(r)})$. Since the set $\text{comp}(\text{int}(A^{D(r)}))$ is connected and unbounded, there exists a curve S such that:
   P4. $x_0 \in S$,
   P5. $\forall \eta > 0\ \exists\, x \in S$: $d(x, A) > \eta$
   P6. $\forall\, x \in S$: $x \in \text{comp}(\text{int}(A^{D(r)}))$.

   Properties P4 and P5 are identical with P1 and P2 and P3 is equivalent to P6.

2. $\exists$ r: $\text{comp}(A^{D(r)})$ is not connected $\Rightarrow$ The set A does not satisfy *Property 1*
   If the set A satisfies *Property 1*, P1-P3 or equivalently P4-P6 must be fulfilled. The set A is bounded and the set $\text{comp}(A^{D(r)})$ is not connected so at least one of its connected parts is bounded. For any x_0 belonging to this part $k(x_0, r) \cap A = \emptyset$ and no curve $S(k(x_0, r))$ exists that fulfils P4-P6.

♦

3. CONTOUR PRUNING

3.1 Mathematical basis of the proposed contour pruning method

Subsequently we prove a proposition that is used in our contour pruning method. The proof is based on triangle inequalities.

Proposition 3
Given are two sets A and B and two points k, l in A. Let

$$\delta = d(k, l) \tag{5}$$

$$d_1 = d(k, B) = \min_{p \in B} d(k, p), \tag{6}$$

$$d_2 = d(l, B) = \min_{p \in B} d(l, p). \tag{7}$$

We claim that:

$$|d_1 - d_2| \le \delta. \tag{8}$$

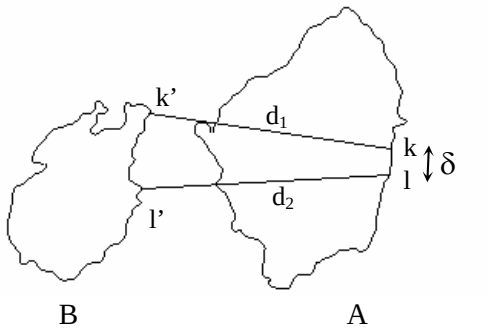


Figure 4. *Notation for Proposition 3.*

Proof

Eq. (6) implies:

$$d_1 \le d(k, l') \quad \forall\, l' \in B. \tag{9}$$

Eq. (7) implies:

$$d_2 \le d(l, k') \quad \forall\, k' \in B. \tag{10}$$

The triangle inequality implies:

$$d(k, l') \le \delta + d_2 \tag{11}$$

$$d(l, k') \le \delta + d_1 \tag{12}$$

From (9) and (11) we have:

$$d_1 \le \delta + d_2 \equiv d_1 - d_2 \le \delta \tag{13}$$

and from (10) and (12):

$$d_2 \le \delta + d_1 \equiv d_2 - d_1 \le \delta \tag{14}$$

From (13) and (14) it follows that:

$$\mathbf{|d_1 - d_2| \le \delta}$$

♦

Let us assume that we have checked the distance to the set B for two points a_m and a_n that belong to the set A (see Figure 5) and let us denote:

$$d(a_m, B) - d(a_n, B) = r > 0. \tag{15}$$

On the basis of Proposition 3 it is known that for every element a_i that belongs to *r*-neighbourhood of a_n:

$$| d(a_n, B) - d(a_i, B) | \le r, \tag{16}$$

From (15) and (16) follows that:

$$d(a_i, B) \le d(a_m, B). \tag{17}$$

As a consequence of (17) all points lying within the *r*-neighbourhood of a_n can be eliminated from computations.

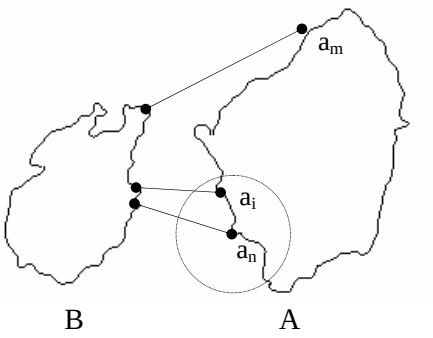


Figure 5. *The knowledge of distances $d(a_m, B)$ and $d(a_n, B)$ that satisfy (15) enables the elimination of Hausdorff distance calculation for pixels in a ball $K(a_n, d(a_m, B) - d(a_n, B))$.*

3.2. An application to contour recognition

In the pre-processing stage both contours to be compared must be ordered so that consecutive pixels $a_{i,}$, a_{i+1} are neighbours on an 8-connected $\mathcal{E}$-grid. In a typical application the query contour is compared with many contours stored in a database which can be ordered off-line during database creation. Since the contours are closed, for *i* greater that the list size |A|: $a_i = a_{rem(i, |A|)}$, where *rem* is remainder after division. The algorithm works as follows:

Algorithm 1

Step 1. Choose an arbitrary integer N > 1. The influence of N on the algorithm performance will be discussed in the next section.

Step 2. Pick up N pixels $a_{p(1)},\ldots,a_{p(N)}$ placed evenly along the contour A and calculate their distances to the contour B. The largest distance will be referred to as d_{max}, as shown in Figure 6.

Step 3. For each pixel $a_{p(i)}$, $i \in 1, \ldots, N$ rule out from further calculations the pixels with subscripts

between p(i)-floor($r_i / \sqrt{2}$) and p(i)+floor($r_i / \sqrt{2}$), where $r_i = d_{max} - d(a_{p(i)}, B)$. It is obvious that all these pixels belong to r_i-neighbourhood of $a_{p(i)}$.

Step 4. Calculate the distance to contour B for all the reminding pixels of contour A. The largest of them is $d_{H+}(A, B)$.

$d_{H-}(A, B)$ is calculated in the same way.

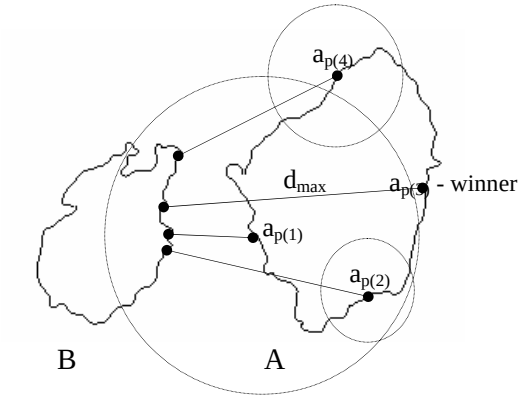


Figure 6. *Illustration of Algorithm 1 for N=4.*

3.3. Efficiency

The efficiency of the method depends on several factors:

1. **The value of N** (i.e. number of pixels taken from the contour A for which the distance from contour B is calculated in the preliminary stage). By increasing N we reduce the computation burden in Step 4 but Steps 2 and 3 become more time-consuming (see also next section).
2. **Shape and relative location of objects.** In the situation depicted in Figure 7b, the calculation time cannot be reduced because for every point of ∂A the distance to contour ∂B is identical. Since the proposed method is used for finding the location which gives a minimal Hausdorff distance, the efficiency changes during the optimisation process.

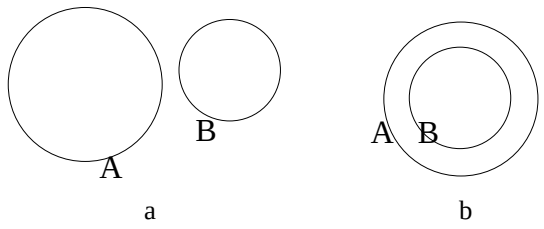


Figure 7. *For different locations of the same objects the efficiency of the proposed method may vary considerably.*

3. **Number of contour pixels.** The method is effective if the cost of distance computations from a single pixel in A to contour B is high, in other words if contour B consists of many pixels. In the case of long contours, it is recommended for every pixel $a_{p(i)}$ to rule out *all* pixels lying is its r_i-neighbourhood by checking $d(a_k, a_{p(i)})$ for every pixel $a_k \in$ A (modification of Step 3). In this case the contour ordering would be used only for placing evenly pixels $a_{p(i)}$. However, random placing could also be applied, thus contour pixels can be given in any order.
4. **Choice of pixels $a_{p(i)}$.** We recommend the choice of $a_{p(i)}$ in equal intervals. However, also the selection of the starting pixel has an influence on the computation time. In the situation depicted in Figure 8a, the method will work very effectively but if pixels $a_{p(1)}$ and $a_{p(2)}$ will be chosen as shown in Figure 8b, no pixel of contour B can be pruned. In the proposed algorithm, by default, $a_{p(1)}=a_1$, that is we start form the first contour pixel. When N increases, the choice of the starting pixel becomes less important because varying the starting pixel causes only small fluctuations of the computation time.

Figure 8. The c*omputation time depends on the choice of pre-selected pixels.*

4. EXPERIMENTS

The test proposed in Sec. 2 has been implemented in Matlab. In case of test, the speed is not very important as it is designed for processing a model object database off-line and making a single query processing before recognition. The method described in Sec. 3 has been implemented in C++ as a Matlab MEX-file.

For experiments we used a database of 70 objects that correspond to the shape of Greek islands. 10 among the 70 objects were chosen randomly and slightly deformed in order to serve as query objects. Examples of the objects included in the database of Greek islands are shown in Figure 9.

Figure 9. *For these six islands considering only contour may lead to an error. All database consists of 70 objects.*

The number of pixels of objects varies from 202 to 28638 (4333 in average) and the number of pixels of contours – from 60 to 1116 (270 in average).

The first series of experiments aims at determining the best value of the parameter N in Algorithm 1. From the time needed to recognize query objects for various values of N, we have found that the distance calculation for N=40 pixels in the preliminary stage of the algorithm is appropriate. In Figure 10 an exemplary relationship is plotted between N and time needed to perform recognition for the first query object (q_1). Recognition means matching a query object with every object in the model object database whereas matching aims to find the smallest Hausdorff distance between all possible transformations of the query object and a model object. The following transformations were considered: translation, rotation, and scaling. Horizontal line indicates computation time without speeding-up. From the inspection of Figure 10 one can see that the application of proposed method can reduce the computation time 3-3.5 times.

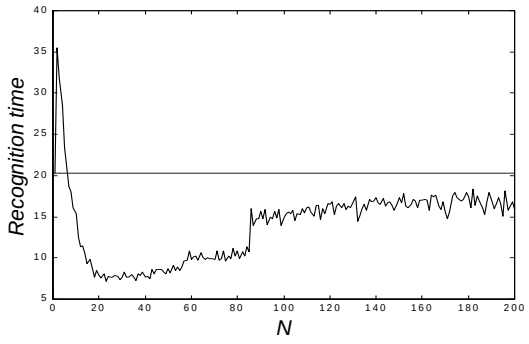


Figure 10. *Recognition time as a function of N for query object q1. The horizontal line indicates computation time for all contour pixels.*

In the subsequent experiments we try to estimate reduction of the computation time offered by our method. For 10 query objects we have recorded time of recognition process.

In Figure 11 for every query three values in logarithmic scale are depicted. The smallest value is the recognition time when Algorithm 1 is applied regardless if objects possess *Property 1*. This approach may lead to errors if the Hausdorff distance is realised on an interior point of one of sets being matched. The second value is the time with application of Algorithm 1 only if it can be done without any risk of error, i.e. if *Property 1* holds. It is worth noting that for 6 among the 70 objects in the database and for 1 among 10 query objects *Property 1* does not hold. Consequently, while matching these objects with another object B, all pixels of the object B should be considered. The third, largest value is the time needed when the Hausdorff distances are computed in a straightforward manner by considering all the pixels in calculations. The values presented in histogram are listed in columns: F, G and C of Table 2. In the latter table, recognition time values when calculations are restricted to contours but without application of Algorithm 1 are listed in columns D and E.

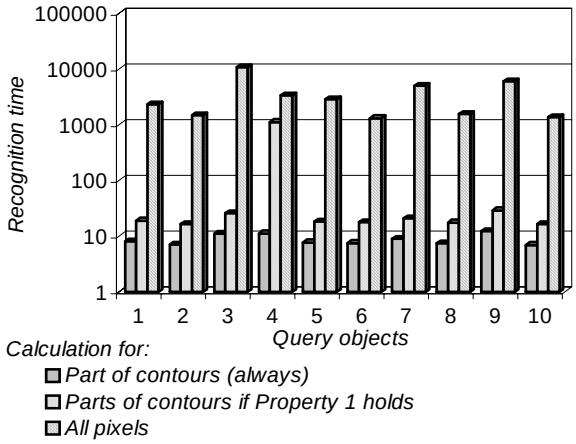


Figure 11. *Recognition time for 10 query objects.*

Table 2 *Time of recognition for different speeding-up actions.*

Query		Overall time for recognition [sec]; calculations:				
			restricted to contours:		restricted to parts of contours (Algor. 1):	
A number	B name	C for all pixels	D always – with error risk	E if Property 1 holds – without error risk	F always – with error risk	G if Property 1 holds – without error risk
q_1 (4)	c_antiparos	2291.1	20.3	68.0	8.0	26.9
q_2 (7)	c_giaros	1470.5	16.1	53.9	7.0	23.4
q_3 (17)	c_rithnos	10614.9	34.2	114.6	10.9	36.7
q_4 (31)	d_leros	3343.2	*33.8*	1119.3	*11.1*	1119.3
q_5 (34)	d_nissiros	2862.5	16.8	56.3	7.7	25.8
q_6 (43)	i_meganisi	1303.8	20.0	67.0	7.5	25.1
q_7 (53)	s_skiathos	4991.4	23.3	78.2	8.8	29.5
q_8 (62)	sa_poros	1555.4	16.5	55.3	7.4	24.8
q_9 (63)	sa_salamina	5974.7	39.9	133.9	12.2	41.0
q_{10} (66)	sg_elafonisos	1364.4	15.1	50.5	6.9	23.2

5. CONCLUSION

In this paper we have presented a method for speeding-up the exact computation of Hausdorff distance between a pair of graphical objects, represented by finite sets of pixels. The method can be combined with existing ones, which are applicable independently at lower or higher stages of matching process (cf. Table 1) in order to speed-up the time needed to perform shape recognition. We have formulated and proved several propositions that enable testing whether we can consider only the objects contours in calculation of the Hausdorff distance. We have also proposed a novel method that allows to eliminate part of contours from computations. We have demonstrated by experiments the savings offered by the proposed methods.

6. ACKNOWLEDGEMENT

This work has been supported by the European Union Research Training Network "Models for Unified Multimedia Information Retrieval" (MOUMIR) and by the Polish Committee for Scientific Research (KBN), project no. 8T11A04010 "Modelling of interactive image recognition process".
The authors are grateful to Professor Adam Korytowski for discursions and comments, which helped us to make some improvements.

7. REFERENCES

[1] D. P. Huttenlocher, G. A. Klanderman and W. J. Rucklidge, "Comparing images using the Hausdorff distance", *IEEE Trans. Pattern Analysis and Machine Intelligence, 15*(9), 850-863, 1993.

[2] A. M. J. Skulimowski and P. Rotter, "Hausdorff distance application to unique 2D object identification", *Working Paper of the Chair of Automatics 84/98*, AGH-University of Science and Technology in Cracow, 1998 (in Polish).

[3] D. Sim, O. Kwon and R. Park, "Object Matching Algorithms Using Robust Hausdorff Distance Measures", *IEEE Trans. Image Processing, 8*(3), 425-429, 1999.

[4] M. J. Atallah, "A linear time algorithm for the Hausdorff distance between convex polygons", *Information Processing Letters, 17*, 207-209, 1983.

[5] H. Alt, B. Behrends and J. Blömer, "Approximate matching of polygonal shapes", *Annals of Mathematics and Artificial Intelligence, 13*(3-4), 251-266, 1995.

[6] F. Preparata and M. Shamos, "*Computational Geometry"* Springer-Verlag, New York, 1985.

[7] C. F. Olson and D. P. Huttenlocher, "Automatic Target Recognition by Matching Oriented Edge Pixels", *IEEE Trans. Image Processing, 6*(1), 103-113, 1997

[8] S. Eschrich, "Edge matching using the Hausdorff distance", *Computational Geometry Project - Final Report, http://morden.csee.usf.edu/~eschrich/cg/*, 2000.

[9] J. Barros, J. French, W. Martin, P. Kelly, and M. Cannon, "Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval", in *Proc. IS&T/SPIE: Storage and Retrieval for Image and Video Databases IV,* 1-2 Feb 1996, San Jose, CA, vol. 2670, pp. 392-403.

[10] P. Rotter "*Application of multicriteria optimisation methods in image interpretation*", PhD Thesis, AGH-University of Science and Technology in Cracow, 2004.

[11] A.M.J. Skulimowski "Mathematical Bases for the Numerical Evaluation of the Hausdorff Distance." *Preprints of the 9th IMACS World Congress, Oslo, August 5-9, 1985; Vol. 5, pp.343-346.*