Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing and Management 41 (2005) 1121–1139
INFORMATION PROCESSING & MANAGEMENT
www.elsevier.com/locate/infoproman

Computing, explaining and visualizing shape similarity in content-based image retrieval

Ioannis Andreou *, Nikitas M. Sgouros

Department of Technology Education and Digital Systems, University of Piraeus, Karaoli and Dimitriou 80, Piraeus, Greece

Received 2 April 2004; accepted 11 August 2004
Available online 14 October 2004

Abstract

Although there is a growing need for Content-Based Image Retrieval systems, their use is often hampered by significant computational complexity and their inability to explain to their users the reasoning behind the similarity and retrieval processes they employ. This paper introduces Turning Function Difference (TFD), an efficient novel shape-matching method, which is based on the curvature of the shape outline and is translation, rotation and scale invariant. The method produces information about the correspondence of points belonging to the compared shapes that are used during the explanation process. TFD explains its results through an alignment and a visual animation process that highlights the similarities between the model images and each one of the selected images as perceived by the method. The proposed shape-matching method is used in the G Computer Vision (GCV) library, a single-object image retrieval system that utilizes information about the objects' outlines and explains the reasoning behind the selection of similar images to the user. The implemented system is freely available for download to all interested users.

Keywords: Shape; Content-based image retrieval; Turning function difference; Computer vision library; Results visualization

1. Introduction

Recently there has been a growing need for visual information retrieval technologies, i.e. systems that allow storage and retrieval of visual media based on visual content. One of the primary research goals in this area is the creation of methods that allow the retrieval of visual content based on the similarity of the visual characteristics of input and database content (Veltkamp, 2001). For humans and animals,

* Corresponding author.
E-mail addresses: gandreou@unipi.gr (I. Andreou), sgouros@unipi.gr (N.M. Sgouros).

0306-4573/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2004.08.008

shape is a dominant characteristic for the identification of similar objects or scenes (Belongie, Malik, & Puzicha, 2002; Palmer, 1999). There are sophisticated shape-based identification methods in machine vision, which are used in several systems (Holden & Owens, 2003; Koller, Weber, & Malik, 1994; Lipson & Shpialni, 2002), but generic Shape Recognition methods, such as those needed in Content-Based Image Retrieval (CBIR) are still under development. As mentioned in Eakins (1996), although techniques for CBIR that utilize color or texture information have significantly matured, image retrieval based on shape information is not yet satisfactorily effective. Ulman (1995) explores several problems associated with shape recognition and related techniques. Furthermore, the use of CBIR systems is frequently hampered by their inability to explain to the user their reasoning during the retrieval process.

Several similarity measures and corresponding methods have been proposed and have been used for shape matching and shape-based image retrieval (Loncaric, 1998), including Turning Function (Arkin et al., 1991; Niblack & Yin, 1995), Moments, Tree Pruning, Geometric Hashing (Wolfson & Rigoutsos, 1997), Deformable Templates, Relaxation Labeling (Kwan, Kameyama, & Toraichi, 2003), Fourier Descriptors (Zhang & Lu, 2001), Curvature Scale Space (Abbasi, Mokhtarian, & Kittler, 1999), Shape Context (Belongie et al., 2002), and Shape Indexing (Del Bimbo & Pala, 1997). The measures and methods that describe a shape as an entity are referred to as *global*, whereas *local* are the ones that calculate similarity by combining the shape matching distances on different locations of the total shape, thus (possibly) providing description of the way a match was achieved.

Most shape matching and retrieval methods utilize and focus on one or few shape characteristics. For example, shape-matching methods such as Turning Function (Arkin et al., 1991; Niblack & Yin, 1995), or the simplest form of Curvature Scale Space (CSS) (Abbasi et al., 1999), are based only on the curvature of a curve. These methods are completely deterministic and lack the notion of experience; therefore they cannot model the heuristic and adaptive nature of the human vision. However such methods or a combination of such methods can have results that often seem logical to a human. But in order to explain to the user the relevance of their results it is highly important for these methods to present what information each shape matching method utilizes and what their results mean. Furthermore, since this kind of research has visual results, it is the authors' belief that a shape-matching process must also provide a visual explanation of its internal criteria, e.g. draw an appropriate ellipsis for a value of *Eccentricity*. The large number of shape-matching methods justifies the need to visualize the internal characteristics of these methods.

This paper introduces a novel method, called *Turning Function Difference* (*TFD*), for the retrieval of similar shapes and the explanation of the retrieved results, based on the comparison of polygonal curves. As will be demonstrated, TFD is capable of fully describing the outline of a shape, in a translation, rotation and scale invariant manner. The basic similarity criterion resembles the Turning Function (Arkin et al., 1991; Niblack & Yin, 1995). The *measure* presented here, Turning Function Difference (TFD) is translation, scaling and rotation invariant. It is also suitable for matching open curves and very robust in dealing with noise. The extra matching information produced by the TFD method lets us produce explanatory visualizations of the match results.

The results explanation process utilizes two methods; an *alignment* method that shows how each one of the selected images can be geometrically transformed to the model image supplied by the user and an *animation* method that shows how and where two objects that were characterized as similar are actually similar. In order to test the effectiveness of the retrieval and explanation methods we have developed a shape retrieval system, the GCV library, which uses two known image sets (see Section 6) as a search space.

The rest of this paper is organized as followed. Section 2 describes the preprocessing steps of our method, while Section 3 describes in detail the actual matching and retrieval process. Section 4 contains a thorough analysis of TFD algorithmic properties. Section 5 describes the explanation process used by the method and Section 6 gives information on the implementation of the GCV library. Section 7 contains an evaluation of the method and Section 8 draws overall conclusions.

2. Preprocessing phase

In current literature, the internal data that is used by a shape similarity method to represent a shape and compare it to other shapes is called a *Shape Descriptor*. This phase consists of the steps that must take place to produce the shape descriptors of the TFD method, provided that the original shapes are available. This means that whatever extraction process is needed to create the shape data has already been applied to the source images. As seen in Fig. 1, the input of this phase consists of the original shapes (polygons) and the output consists of the TFD shape descriptors, i.e. arrays of TFD values.

The preprocessing phase consists of the following steps:

(a) Convolving the initial polygons with a Gaussian matrix to minimize extraction errors and noise.
(b) Sampling the polygon into N equally spaced (on the initial contour) points. N will be referred to as *resolution*, from now on.
(c) Produce the TFD values.

The basic metric, TFD (at a vertex, in a polygon traversal), is defined as the angle between the edge following the vertex and the edge before it. Consequently, this metric is the *Difference* in the *Turning Function* between successive vertices. This new definition embeds rotation invariance in the metric itself. For a traditionally defined turning function, a rotation corresponds to a selection of a different starting point. Practically, this includes at least one unnecessary subtraction of a reference value to compare to a rotated version of the turning function. The TFD values are constrained inside $(-\pi, \pi]$, as this form is the most efficient one for the computations required by the system. This is also the representation that best describes a traversal from one vertex to another, through right-hand or left-hand turns, followed by straight segments, which was the basic inspiration behind this method. A histogram comparison of the above definitions is depicted in Fig. 2. For each input polygon, the output of the preprocessing phase (which is used in the retrieval process) is an array of N TFD values.

3. Shape matching and retrieval

3.1. Retrieval overview

The GCV library uses a multi-step shape retrieval process that iteratively applies pass/fail criteria to the initial database, until only the elements that pass all the tests remain. These elements are then sorted by similarity, and are ready for presentation. One efficient configuration is to apply some *Global Features* (*GF*): *Circularity*, *Convexity*, *Eccentricity* (Abbasi et al., 1999) as criteria, before the TFD criterion, as they are executed faster. This involves tests where the *distances* of the corresponding *Global Feature* values of the two compared polygons must be below a predefined threshold. This configuration leads to a significant pruning of the search space before the TFD method kicks in.

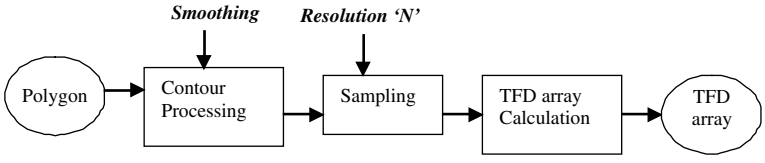


Fig. 1. Preprocessing phase.

Fig. 2. Polygonal shape (a), Turning Function (b) and TFD (c). In the histogram the turning function is in $[0, 2\pi)$, while TFD is in $(-\pi, \pi]$.

3.2. Matching with TFD

The TFD matching process accepts as input two arrays of TFD values (of size N, since each polygon is closed and all shapes are equal in size, due to the sampling process) that represent the outlines of the shapes to be compared. The method also accepts a set of (probably constant) parameters (*minScore*, *minLength*, *angleErr*, *distanceErr*, *rotationErr*), that will be explained below. The output of the method consists of (i) the correspondence between regions of the polygons that are considered similar and (ii) the matching score. The matching score could be the number of vertices contained in matched areas. Alternatively, in the global matching procedure discussed here the score can be defined as a number in the range [0, 1] that represents the proportion of the number of vertices that were matched to the total number of vertices in the shape's outline. Typical values for *minScore*, which is the score threshold, are (1/2) for strict selection of similar shapes, and (1/3) for retrieval of elements that simply contain similar regions. The experiments made on the test databases have given satisfactory results with these values. If the TFD method is used as the only criterion, the first value should be used. The overall procedure is depicted in Fig. 3.

3.3. Finding matching regions

This step seeks to find corresponding regions of similarity between two shapes. To this end, this step accepts as input a value (*angleErr*) that denotes the maximum difference (as in Fig. 3) between corresponding values of the turning function in the two shapes in order for them to be matched successfully. Furthermore, it accepts as input the minimum number of consecutive turning function values that can qualify as matched regions of the curves (*minLength*). The polygons to be matched against each other are given in the form of two TFD value arrays (*A1*, *A2*) of size N.

A sequence-checking procedure begins for each pair of elements (*a1*, *a2*), where *a1* belongs to *A1* and *a2* belongs to *A2*, and checks them and the elements that follow them in order to see if the (absolute) distance between corresponding elements of the two arrays is below *angleErr*. The engine is implemented in a way that it accepts different sequence checking modules. During our tests we have concluded that there is an additional precision benefit if we also check that the difference of the sums of the TFD values (always constrained in $(-\pi, \pi]$) is below *angleErr*. Fig. 4 explains how this affects the angle-checking process. If the length of a sequence is greater or equal to *minLength* then this becomes a candidate matching sequence,

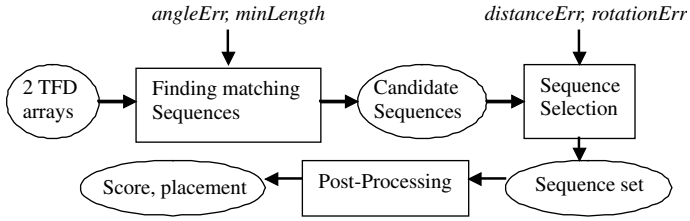


Fig. 3. Matching with TFD. Each of the three individual steps are designed to be independent, meaning that different implementations of each step can be applied, as long as they obey a set of rules (implement an appropriate Interface).

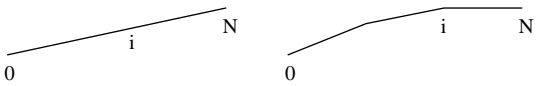


Fig. 4. If these two curves were compared, they would have passed the simple angle check process, as the individual differences in each vertex pair are small. However by TFD-comparing the sums of angles (the total change of angle), as the checking process proceeds forward, the test fails at the *i*th vertices. It is acceptable that the curve parts 0–*i* are similar but this is not the case for curves 0–*N*.

expressed in the form of: (*start1*, *start2*, *length*) where *start1*, *start2* are the corresponding starting vertices of the matching regions and *length* is the number of edges in each region. A choice of *angleErr* in the range [0.4, 0.5] radians has proven to work effectively for any (acceptable) resolution *N*. The choice of *minLength* is discussed in the following section.

3.4. Selection of matching regions

This step constructs the final set of matching regions (*matchSet*) between the two shapes that comprise the final match. *MatchSet* is the longest set (cumulative) of matching sequences that are *compatible* with each other. Two matching sequences are considered *compatible* iff they pass two tests that are based on distance and rotation. These tests are explained below and are governed by the *distanceErr* and *rotationErr* parameters. For the sake of simplicity, in this paragraph we will denote the matching sequences (or similarity pairs, from now on) as *X*–*Y*, where *X* is a region of a shape, which is found similar to a region *Y*, of another shape. An example is provided for each test to aid the readers.

The first compatibility test is that of distance. For the matching process, two matching sequences have to be placed in reasonable distances in terms of vertices (*distanceErr*). Two regions in one polygon that are 'close together' could not be matched, respectively, to two other areas in the other polygon that are quite 'far apart'. Lets consider the case of Fig. 7, where the matching sequence (or similarity pair) A1–B1, which consists of regions A1 and B1, is checked for compatibility against A2–B2, which consists of A2 and B2. By [x]s (or [x]e) we denote the start (or the end) of sequence *X* in a clock-wise traversal of the polygon. The distance between vertices (e.g. between a1e and a2s) is measured in number of vertices (on the sampled polygon, where the vertices are—almost—equally spaced). The length of the clock-wise traversal of the polygon from vertex *x* to vertex *y* will be denoted as $x \rightarrow y$. For A1–B1 to be compatible to A2–B2, the absolute difference between the a1e $\rightarrow$ a2s and b1s $\rightarrow$ b2s lengths must be below or equal to *distanceErr*. This rule also must be also applicable for a2e $\rightarrow$ a1s and b2e $\rightarrow$ b1s. In short these rules are expressed as Fig. 5.

It the case of Fig. 7, the above rules cannot be applied to the relationship of A1–B1 with A2–B2. If someone measured the distance a1e $\rightarrow$ a2s, it would be a lot longer than the distance b1e $\rightarrow$ b2s. Thus A1–B1 is not *compatible* with A2–B2. However the distance c1e $\rightarrow$ c2s is almost equal to distance a1e $\rightarrow$ a2s and c2e $\rightarrow$ c1s is almost equal to distance a2e $\rightarrow$ a1s. This means that the *similarity pair* A1–C1 is *compatible* with A2–C2, according to the first test.

The second compatibility test is the rotation test and it is governed by the parameter *rotationErr*. Two regions in one polygon that are *similarly oriented* could not be matched, respectively, to two other areas in the other polygon that are not. To define *similar orientation* we first need to define *Change of Orientation* between two vertices. Change of Orientation between vertices *x*, *y* (of the same polygon), is the sum of TFD values that are encountered in a clock-wise polygon traversal from *x* to *y*. It will be denoted *Rot* (*x*, *y*).

$$|(a1e\text{->}a2s)\text{-}(b1e\text{->}b2s)| <= distanceErr \quad \text{AND}$$

$$|(a2e\text{->}a1s)\text{-}(b2e\text{->}b1s)| <= distanceErr$$

Fig. 5. Rules for the distance compatibility test.

Using the same notation as in the distance test, two regions ($X1$, $X2$) of polygon X are similarly oriented to two regions of polygon Y, ($Y1$, $Y2$) when: Fig. 6.

It the case of Fig. 7, these rules cannot be applied to the relationship of A1–B1 with A2–B2. *Rot*(a1e, a2s) is almost zero, while *Rot*(b1e, b2s) is almost 90°. A reasonably defined value for *rotationErr* causes the above rule to fail. However the relationship between A1–C1 and A2–C2 follows the rule. Thus this relationship passes the rotation test.

The overall results for Fig. 7 would be that A1–B1 is not compatible to A2–B2, whereas A1–C1 is compatible to A2–C2. Thus when the final match set between shapes A and B is created, it contains only A1–B1 (as it is larger than A2–B2). However when A is compared to C, the final match set contains both A1–C1 and A2–C2. This results in better similarity score between A and C than between A and B.

A value of 0.4 radians for *rotationErr* is always good for any resolution N, as we want this criterion to allow an amount of variation in the rotation of the matching sequences. A proven good choice of *distanceErr* is $(1/10) \times N$, although we chose *distanceErr* always above 1, to account for small noisy regions that might have increased/decreased length. The power of the TFD matching method stems from the fact that if a larger matching sequence cannot be established because of small noisy regions (or improper scaling caused by the sampling process), then two or more candidate matching sequences will be created and they will be selected in the final matching sequence set by this step, reflecting the otherwise uncaught (but probably obvious to a human) partial similarity of the greater regions. The value of *minLength*, used in the previous step, besides ruling out possibly 'lucky' matches, has an efficiency improvement effect on this step as we have to test fewer combinations of sequences. To achieve this, we chose a value of *minLength* close to $(1/8) \times N$. Such a *minLength* selection also restricts the cardinality of the matching sequences in the final set.

After constructing the final matching sequence step, the similarity score can be calculated by summing the lengths of the matching sequences and then dividing with the minimum length of the two matched poly-

AngleCheck(Rot(x1e, x2s)-Rot(y1e, y2s)) <=*rotationErr* AND

AngleCheck(Rot(x2e, x1s)-Rot(y2e, y1s)) <=*rotationErr*

Fig. 6. Rules for the distance compatibility test. The AngleCheck function constrains the angles inside $[0, \pi]$, first by constraining them in $[-\pi, \pi]$ and then by removing the sign.

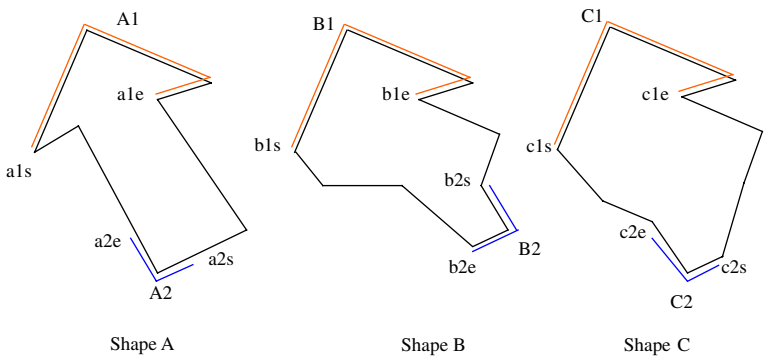


Fig. 7. Example that demonstrates the application of compatibility rules between similarity pairs. A1–C1 is compatible to A2–C2, whereas A1–B1 is not compatible to A2–B2. Thus shapes A and C are more similar then A and B are. Only the noted similarity pairs were supposedly found. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

gons (in case the lengths are not equal). It is apparent that if the all polygons in the database are sampled at the same size, the calculation of matching scores will be more reliable.

4. Analysis of the TFD algorithmic properties

4.1. General properties

The matching algorithm described has a number of desirable properties. It is rotation invariant, by definition of the metric itself, as the TFD descriptors are defined (as angles) relatively to each value's previous value. Of course the algorithm is also translation invariant as the coordinates information is eliminated in the construction of the TFD array.

As mentioned in the previous paragraph, the TFD matching method is capable of capturing the partial similarity of two large shape regions by recognizing the smaller regions of them that are actually similar. By accepting a tested and limited amount of error, in both distance and rotation, it is possible to sort out the smaller regions that are not similar. This enables an effective calculation of similarity, which is robust to noise introduced either by the initial extraction of data or the sampling process. Smoothing is generally inseparable from the sampling process, as it ensures that there is no excess noise in individual shapes. The TFD method measures length in number of TFD values. Without noise reduction, some areas of a shape that contained a lot of zigzags, could be regarded as longer than they actually are, by the matching process, making it unreliable.

The resolution parameter (N) is an easily modeled independent parameter. Increasing N, the captured detail of the shape is also increased, leading possibly to better results. However, if N is very large, noise can change the size and values of a TFD array significantly. Experiments with the tested shape sets have revealed that it is best to sacrifice some detail to achieve more robust results. Nevertheless, if detail is prioritized in an application (which means that the initial specimens are exact and do not require smoothing), it is possible to increase N. It is proposed that the other parameter values remain the same (or change slightly). The region selection process will again compensate for any noise. The outcome will be more precise, although more expensive, as the complexities of all steps depend on N.

4.2. Scale invariance

Scale invariance is easily achieved in the case of global (for complete shapes) matching, which was described in detail above. By sampling all shapes with the same resolution (after excess noise has been removed by the smoothing process), the shapes are represented as equal in outline length. One could argue that even though some noise has been removed, still there is enough of it to affect both the number of TFD values created from a region of the initial shape and the TFD values themselves. However, the TFD method can handle this, as it can reject the initial larger region and select two or more pairs of smaller sub-regions that have actually similar elements. To achieve this, the threshold values that are chosen for distance and rotation errors must be very carefully picked. The paper contains recommendation on all the parameter values of the proposed method.

Scale invariance can be emulated in the case of partial matching of shapes, i.e. the case where at least one of the shape curves is not closed. This is vital in cases where objects overlap in a scene or any other case where a whole shape is unavailable. Andreou & Sgouros (2003) presents an application where a partial-matching version of TFD was used to retrieve from a database shapes that are similar to what a user draws on a screen. Most aspects of the shape-matching process remain unchanged. Still one of the two elements resides in the database and is sampled using a predefined resolution; what needs to be determined in this case is the resolution that the second shape (query shape) has to be sampled with. Remember that TFD

measures length in number of vertices of the sampled polygons. Thus, if a complete database shape is sampled to a resolution of X, then an open curve that is supposed to cover one half of the complete shape (for simplicity, we consider the lengths of two 'complete' shapes equal) to which it belongs, it has to be sampled with a resolution close to $X/2$. If the coverage of the complete shape of the model curve can be constrained inside limits, a number or increasing resolutions inside these bounds and appropriate TFD descriptors can be constructed to query the database with. The TFD method will again handle the inaccuracy introduced by the sampling process, thanks to its ability to capture the partial similarity of larger regions. The best results from all queries are then retrieved.

4.3. Modularity

Another aspect of the TFD method is modularity. The shapes are initially represented as coordinates of vertices. The transformation into TFD values, after the sampling process, only removes the translation and scale information that exists in the initial information, if the sampling effect is ignored. Thus, although the data are converted to a form more convenient to the TFD method, more tests can be added, as long as the transformation from the initial state to the TFD arrays state is kept in mind. As seen above, the total rotation change test, added in the Matching Regions Finding step improved the reliability of the method. Furthermore, the two tests of the selection process are independent, meaning that one on of them can be disabled or that a new test can be added. In fact, different configurations had been tested before concluding to this two-step configuration, with the distance step being executed first. The above facts demonstrate the capability of the method to be extended and configured; an ability that stems from splitting the method to independent and well-defined intermediate steps.

4.4. Complexity and speed

The algorithm itself is probably one of the fastest local matching methods. It has a complexity of $O(N^4)$ but the witnessed average is much smaller. As complexity can be calculated, the algorithm is dependable for real-time applications, as N is a constant value, thus making the maximum computation time a computable, constant value for a predefined system. The sampling process complexity is $O(N)$ and the actual computation time will not be regarded as part of the matching time as it can take place before it. Region finding has a theoretical complexity of $O(N^3)$, as for each angle of the first shape and each angle of the second, starts a process that has at most N steps (comparing and checking TFD values), each of constant complexity. In order to reach this maximum, each angle of the first shape must be equal to each vertex of the other shape. The only case where this it true, is when two circular shapes are compared. Special code has been added to cut off this case. Simply, when a pair of similar regions covers a large percentage (about 90%) of each shape, the shapes are considered *exact* and the shape-matching process ends with only this region pair in the final match set, making this case the fastest executed one. Fig. 8 explains this case.

The last phase, which is Region Selection, is theoretically the slowest one. Since a matching pair can start from each pair of angles, the maximum number of such pairs is N^2. However, because of the optimization technique used, the actual maximum number of pairs is much smaller, and will now be called M. A *minLength* value of $(1/8)N$ further upper bounds M, by excluding the smaller region pairs, while the average speed gained is a lot more significant. This process has to check any combination of these elements, which would normally lead to an enormous complexity (2^M). However, it is possible to break this process down to two steps. The first step has $O(M^2)$ complexity and the second has a theoretical complexity of $O(2^M)$. It will be proven that the second step has a lot smaller complexity than 2^M and it will be demonstrated that the second step is always faster than the first. Consequently, the practical complexity that must be calculated is M^2.

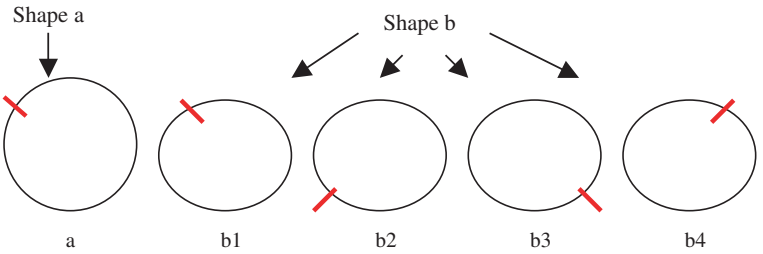


Fig. 8. Shape *a* is compared against shape *b*. Both shapes are circular. This means that a matching sequence starting from the vertex of *a* which is at the red line can have corresponding sequences in *b* that start at the red lines that are at b1, b2, b3 or b4, or actually any vertex, as long as the TFD angles are the same (having the same resolution and being 360° each, will make all the TFD values equal to $360/N°$, for each shape). This will create N matching pairs for each of a's vertices and N^2 total pairs. So the first time a pairing such as that between a and b1 is found (which covers all—or almost all—the length of the shapes) the process ends with only this pair found (the other pairs are similar, since the shapes are circular). This works correctly also in the case of exactly similar shapes, since if a large coverage is found in a similarity pair, the process should end this way. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

The first step of Region Selection is building a *compatibility map*. This map is a 2D array, let it be named C, where $C[i][j]$, $0 < i < M$, $0 < j < M$ is true iff the ith similarity pair is *compatible* with the jth similarity pair. *Compatibility* is defined as in Section 3.4. Creating this map has a complexity of $O(M^2)$, which theoretically is $O(N^4)$, but in TFD is much smaller, as it is explained below. After the *compatibility map* is created, the possible combinations can be computed faster, and only the best combination is kept. The last step still has the same complexity as finding all the combinations between M elements, but it is much faster than checking *compatibility* each time it is needed.

Seemingly, the complexity of the second step should be $2^M - 1$. However, it could never come close to that. The complexity of the implemented best combination finding algorithm *depends on the number of compatibilities* in the map. It is impossible to create 2^N compatible similarity pairs; in fact, the largest group of compatible pairs contains less then N elements. To make a group of N compatible similarity pairs, all the regions in the group should have a length of 1 and be placed one after the other on the curve, which is impossible. We will name the maximum number of similarity pairs in a match set, for a resolution N, a number of eligible pairs M and a predefined set of TFD method parameters, as R. This constrains the complexity as in Eq. (1) (complexity of finding the best combination for the match set):

$$T = \sum_{1}^{R} \binom{M}{i} \tag{1}$$

From Eq. (1) it is obvious that if $R < M$, the complexity decreases. As mentioned above, $R < N$ but the TFD method imposes further restrictions on both M and R. First of all the value of *minLength* restricts R with an upper bound of the integer part of (N/*minLength*). For example by setting *minLength* to $N/8$, R is upper bounded by 7 (pieces cannot be glued together). Also, $M < N^2/4$, as the maximum number of pairs per vertex is $N/2$. This results from the angle-checking step, since a similarity region cannot start from a vertex whose previous vertex was already included in a similarity pair for the current polygon traversal. So a shape comparison with *minLength* = $N/8$ cannot reach the complexity in Eq. (2) (theoretical complexity for a shape comparison, when *minLength* = $N/8$). Furthermore, several optimizing checks, such as the one that checks for exactly similar (>90% coverage) shapes (which would be activated in a comparison of two circular shapes), described above, do not allow the blow up of combinations that leads up to this number.

$$T = \sum_{1}^{7} \binom{N^2/4}{i} \tag{2}$$

The implemented computation method is based on a custom execution stack for the combinations and intermediate results. Thus it is possible to take advantage of the restrictions on R described above. Practically, finding the best combination is not the slowest step of the method, because N never gets large enough to make the execution of the simple *if* checks in it slower then M^2 *compatibility* checks. Values of resolution up to 256 have been tested and this step was faster in all match tries than building the compatibility map. Using resolutions larger than 256 is improbable, at least for global shape retrieval; the initial shapes have to be very detailed, otherwise a larger resolution will create a lot of sampling noise, In order to exactly calculate the complexity of the algorithm, a specific resolution has to be considered. As seen from our tests however, for reasonable resolutions ($\leqslant$256), the most demanding step is that of building the compatibility map, which is $O(M^2)$, i.e. smaller than $N^2/4$. The maximum execution time for a shape comparison can be estimated for a predefined system by adding the maximum durations of all steps. We have tested this method on a Pentium 3 running at 1 GHz, and on average it compared 1500 shapes per second, with a resolution of 64 (without the preprocessing phase). This includes the duration of all the processes that are executed from the moment the user queries the engine and it returns the results. The reader can verify that by running the test application (Section 6).

4.5. Configuration parameters

In the context of describing the TFD method, it is vital to justify the selection of the provided configuration values. The validity of the method itself, as a calculator of shape similarity is obvious, as it returns the percentage of the continuous regions of the outlines of two shapes for which the change of orientation ($d\theta$, where θ is the orientation of the current edge) are similar, in the totality of the outline of the shapes. In order to make this method a useful and efficient shape-matching tool, several constraints must be provided. For example, *angleErr* was defined as the maximum difference in the change of orientation between two pairs of edges that belong to corresponding similar regions. The proposed values (in the [0.4, 0.5] range) for this parameter, as well as other parameter values that have been provided in the course of this paper were produced after experimentation with a number of different shape sets, and have proven to be those that are most appropriate for use with all of them.

5. The explanation process

5.1. Calculation of explanation parameters

As TFD is a local similarity measure that is based on the outline of shapes, it is possible that more information can be elicited out of the matching sequence set. From the matched vertex sequences, pairs are made between corresponding vertices of the two polygons. A procedure has been developed, that maps each vertex of one of the two matched polygons to a vertex of the other matched polygon.

Let polygon A be matched with polygon B and let the result of the TFD comparison be a set M, of elements described as: (startA, startB, length). From this, a direct correspondence can be found between the vertices of A that are included in the areas $[\text{startA}_i, (\text{startA}_i + \text{length}_i) \bmod (N_A)]$, with N_A being the size of polygon A, to vertices of polygon B. For the ith element of a matched arc described by an element (startA, startB, length), belonging to M, where $(0 \leqslant i < \text{length})$, a pair (a_i, b_i) is created where $a_i = (\text{startA} + i) \bmod (N_A)$ and

$b = (\text{startB} + i) \bmod (N_b)$. Thus an array of pairs of corresponding vertices V (with elements $[a_i, b_i]$) can be elicited from the comparison process.

Now for each vertex a_i, of polygon A that is not a part of a pair that belongs to V, a corresponding vertex in polygon b can be found as follows. First we need to locate the immediately previous and the immediately next vertices (a_p, a_n, respectively) that belong to such a pair, i.e. have a corresponding vertex (for each) at polygon B, named b_p, b_n, respectively. Let us define traversal distance D of the ith and jth vertices of a polygon A as $D_{Aij} = ((j + N_a) - i) \bmod (N_A)$ where N_A is the number of vertices of the polygon. The index of the corresponding vertex for a_i will then be $p_B + D_{Aip} \times (D_{Bnp} - D_{Anp})$.

After a mapping of all vertices of A to vertices of B has been made, it is possible to obtain the rotation, scaling and translation parameters for the best geometrical transformation of B to A. The first step is to normalize the position of the polygons. Normalizing means that they are centered around [0, 0] by translating by $-C_A$, $-C_B$ respectively, where C_A, C_B are their respective centers of gravity. Let us refer to the normalized versions of the two polygons as NA and NB.

After position normalization a transformation is made on all vertices of the two polygons: for each *vertex V*, calculate the angle between the *vector V* (vector defined by (0, 0) and the vertex) and the X-axis. This provides us with rotation information for each vertex. Next we pick a vertex of polygon A that has a corresponding vertex in B (after the previous steps, every vertex of A has one). In order to make sure that no error is introduced by the mapping of vertices that were not originally matched, we select a vertex of A that *was originally matched* to a vertex of B, i.e. the first such vertex. The rotation angle (*sang*) is simply defined as the angle between the vectors defined by the vertices of NA and NB that were picked and (0, 0). This value is the rotation angle that NB must be rotated around (0, 0) to closely match NA, according to the mapping of their vertices that was produced by the previous steps.

In order to minimize the rotation error between the final versions of NA, NB we correct the value of *sang* using the following procedure:

1. Rotate NB by *sang*.
2. For each vertex of NB mapped to a vertex of NA find the angle between the vectors defined by the corresponding vertices, as in the definition of sang.
3. Calculate the average for the above quantities (*sangCorr*).

sangCorr is the correction that should be applied to *sang*, simply by setting: *sang* = *sang* + *sangCorr*. To optimize the above procedure, remember that we already have the angles of NA and NB vertices with the X-axis. This means that the angles of step 2 can be calculated immediately, without rotation, by simply subtracting *sang*, and then constraining the result in $(-\pi, \pi]$.

The definition of the scaling parameter depends on the presentation of the results. Thus we will only demonstrate one method that utilizes the special information returned by the TFD matching process and focuses on minimizing the distance between the *matched vertices* of the two polygons, after they have been normalized into NA and NB, and NB has been rotated by *sang*:

1. For each vertex of NA, mapped to a vertex of NB, calculate the distance of the two corresponding vertices from (0, 0), L_{Ai}, L_{Bi}.
2. Calculate the ratio of the result values for every execution of the above step, i.e. $\text{ratio}_i = L_{Ai}/L_{Bi}$.
3. Calculate *scale* as the average of the ratios returned by each execution of the above step.

Note, that using the above definition we can find the *scale* that best scales NB towards NA in order to minimize the distance between the *matched vertices* or to minimize the distance between *all the mapped vertices* (as we have already obtained such a mapping). Remember that NA and NB have retained their original sizes.

What are left to calculate in order to place B, on top of A, in a manner that minimizes their distance are the translation parameters, i.e. *Tx*, *Ty*, the translation in the *X* and *Y*-axis that must be applied to B to be placed on top of A. These can be found simply by subtracting the coordinates of the initial centers of A and B, C_A and C_B respectively.

5.2. Visualization methods

5.2.1. Object alignment

This and the following paragraphs describe the visualization methods that explain the search results returned by the TFD method, using the *shape matching set* and the processes that were presented in Section 3.4. Our goal is to place polygon B on top of polygon A, in a manner that minimizes the distance between corresponding matched vertices. Only geometrical transformations are used during this process. The procedure follows:

1. Produce NB from B (after the above calculation steps this already exists), by translating its center to (0, 0).
2. Rotate NB by *sang*, calculated as previously.
3. Scale NB by *scale*, calculated as previously.
4. Translate NB by C_A.

These steps will produce an affine transformed version of B, which sits on top of A, B1. B1's vertices that are mapped to vertices of A (using a mapping produced in previous steps using A and B) and have a close to minimum average distance from those vertices of A. Note that for this procedure too, one can use the mappings of *all* the vertices or the mapping of only the *matched vertices*. The process almost minimizes the distances for each set of correspondences (Fig. 9).

5.2.2. Animation

Since the average user of a CBIR system lacks the specialized knowledge to understand the inner workings of such a system, it is only by *seeing* the effects of the matching method that s/he can judge if shapes are actually similar. And it is best that not only retrieval results are shown, but to also provide an explanation of the results that highlights specific similarities, whenever this is possible.

So the user of a system might find it visually explanatory to watch an animation that shows *how* and *where* two objects that were characterized as similar, *are actually similar*. Another reason that this is nec-

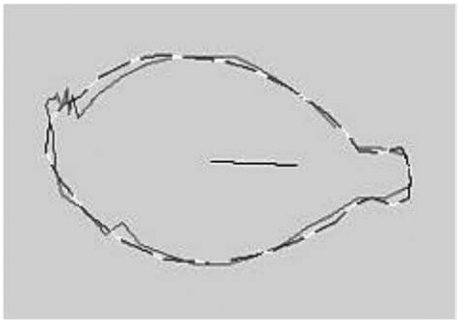


Fig. 9. Shape alignment. Two marine life specimens were placed on top of each other using the alignment procedure.

Fig. 10. Animating an object from the initial state (image shape) to the final state (model shape, apple). The green lines inside the shapes exist to show the starts of the matched sections and their transformations. For the first six positions the shape only rotates. The size of the shape changes just to fit the drawing rectangle. During the latter 6 positions, an interpolation is made between the rotated, first shape's vertices and the corresponding second shape's vertices. Note that the red dotted areas change insignificantly, showing that these areas are actually similar. The blue (not matched) areas of the shape are affected very much by the interpolation, justifying the rejection of these vertices in the final TFD match set.

essary is that it is not always apparent to a human where the similar regions exist. So a visualization of the matching process would help the user understand why the CBIR system believes that two images are similar, even if he does not apprehend this immediately, e.g. if one of the shapes is rotated by 90°. The animation process described below is based on the exact same data that were retrieved and extracted from the TFD matching process in the previous paragraphs. The above alignment procedure is one step to this direction, but we will provide a more complete visualization here: a *Match Animation*.

During the animation, the first polygon (A) remains unchanged. The second polygon (B) however, in the first n frames is slowly rotated until it is rotated by *sang* as this was calculated above. In the next m frames the vertices of B are slowly interpolated until they become equal to the vertices of A that they are mapped to (a mapping is produced as in the previous sections). One example animation is shown below. It shows the transformations applied to B during its change from its original state to a state similar to A. It is possible to access these animations and produce these images and others using the GCV C Application Programming Interface. A demonstration of the animation process is provided by the demo Java application (Andreou, 2002b). Alternatively, use one of the open-source examples to produce an image such as in Fig. 10.

6. Implementation

The image retrieval and explanation process described in this paper are part of a shape-matching system that uses a *C* interface, the *G Computer Vision* library. This library, besides shape-matching algorithms, also implements several image processing algorithms and utilities, and introduces a file format for shape input. The readers are encouraged to test the library and its output, both in terms of Shape Retrieval and in terms of Results Visualization, through some open-source applications that were developed in C++ or Java (the latter use the Java interface of the library), and are available at the GCV library site (http://thalis.cs.unipi.gr/~gandreou/gcv/). Among the demo applications is one, named *JcvRetrieve*, which demonstrates all the work presented in this paper. Fig. 11 contains a screenshot of this application.

Two image databases have been tested with the system. The first is the SQUID (Abbasi et al., 1999) database of marine life, for which an extraction process was not necessary, as text files that contain the shapes' outlines where available. The second is the "Snodgrass and Vanderwart Like" image set (downloaded from web page with several image sets for Computer Vision, http://www.cog.brown.edu/~tarr/stimuli.html) created by B. Rossion and G. Pourtois. Several shape databases were created from subsets of these shape/image databases. To extract the shape outlines from images, a simple extraction process was used, which treated the external connected area as background. The shape outlines, which came out as the border between the depicted object and the background, were stored in PDB, a novel file format supported by the GCV library (Andreou, 2002b). Appendix A explains in detail the PDB format (Fig. 12).

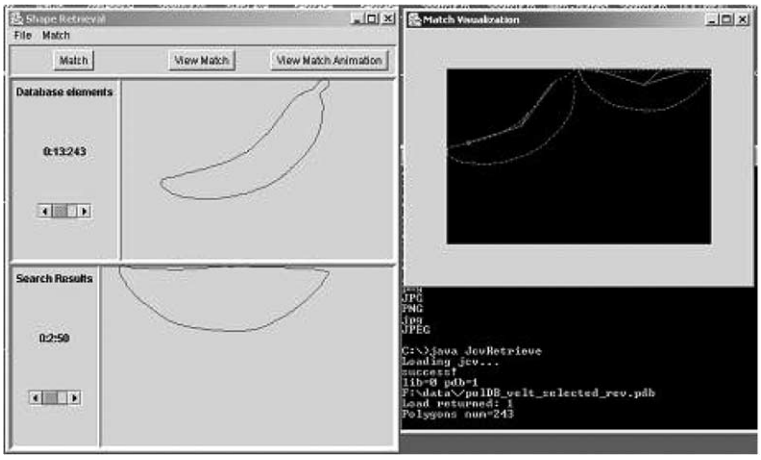


Fig. 11. The *JcvRetrieve* application. The left window depicts the selected item (upper panel—banana) and one of the relevant shapes in the database (lower panel). The right window explains where the similar regions exist between the two shapes, by painting them with the same color.

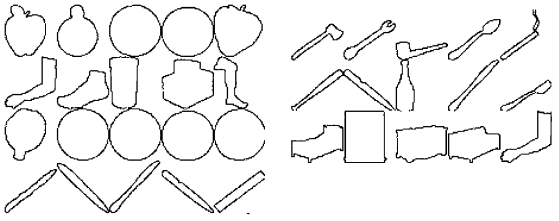


Fig. 12. Search results—top 5 for each object—for Turning Function Difference, based on the "Snodgrass and Vanderwart Like" image set. No Global Feature criterion was used in the retrieval of these results.

7. Evaluation

In order to evaluate the robustness, efficiency and applicability of the TFD method, a two-part evaluation was carried out. The first part of the evaluation was an automated evaluation that measured (a) the precision and recall of the method and (b) its speed, using a specially designed shape database. The second part of the evaluation was implemented through a user evaluation of SR-Sketch (Andreou, 2002a), a sketch creation system which is based on the TFD shape-matching method.

7.1. Automated evaluation

The first part of our evaluation consists of an automated test that measures the overall performance and efficiency of our shape retrieval method in comparing closed polygons. To this end, we employ two factors, namely precision and recall. Precision measures how well the retrieved items match the referred query and recall reveals the percentage of relevant documents that are retrieved by the module.

Out of a shape database for marine life, Squid (Abbasi et al., 1999), we chose 20 *independent* groups of shapes (each consisting of one shape). These groups have the characteristic that when querying with a group's shape the engine retrieves all the elements in it and *only them*, for a (minimum) *score* equal to

Table 1
Search engine evaluation results

DB-*score*	Precision	Recall
Indep-1/2	1.0	1.0
Indep-1/3	1.0	1.0
Noise-1/2	0.999	0.853
Noise-1/3	0.932	0.956
NoiRot-1/2	0.967	0.798
NoiRot-1/3	0.896	0.810

Table 2
Search engine performance evaluation for steps A and B of the matching process

DB-score	Step A: Angles (ms)	Step B: matchSets (ms)
Indep-20	1.5	0.8
Indep-13	1.5	0.8
Noise-20	7.87	5.01
Noise-13	7.85	5.02
NoiRot-20	8.82	4.22
NoiRot-13	8.96	5.04

Note that the first database (rows 1 and 2) contains 20 elements, while the others contain 200. These values were obtained on a Pentium 3 1 GHz machine with 128 MB RAM, running Windows 2000.

1/2, as defined in Section 3.1. The test is run twice, once with *score* = 1/2 and once with *score* = 1/3. Obviously recall and precision for any element using this database are equal to 1.0. We will use a randomization process on every element in this database that N times will select a point in the database and randomly move it in one of eight directions, to randomly introduce noise to the curve. The randomization process is repeated 10 times in order to produce a randomized database of 20 groups each containing 10 similar but 'noisy' shapes. An equally sized second database is produced by randomly rotating the noisy shapes. To simulate the noise inherent in a shape that is extracted by a digital image, the shape coordinates are always rounded to integer precision, before the creation of TFD value arrays.

Table 1, shows overall average results for each database and parameter sets.

As discussed in Section 4.4, step B of the matching process (selection of matching regions) is not the most time consuming step, although all the combinations of the matching sequences produced by step A (selection of matching regions) have to be tested. A good choice of *minLength* plays an important role in the execution of step B. In Table 2, we present the execution times for steps A and B of the matching process, for the previous evaluation, that use a value of $minLength = (1/8) \times N$.

The results of our automated evaluation are satisfying as they demonstrate the rotation invariance of the method as well as its ability of retrieve relevant results even when noise has been introduced to the search space elements. These are necessary properties for a shape retrieval method used in CBIR application, as the user expects the relevant items to be retrieved, even after geometric transformations with different parameters have been applied to them. It is also vital that the engine is able to cope with areas of error (dissimilarity) introduced by digitizing visual media or other means, something that is achieved by step B of the matching method. The second, equally important, consideration for the shape-retrieval engine is speed. The user cannot wait long for the engine to retrieve the relevant shapes. As the tests show, with a search space of 200 elements, the matching results were always produced in less than 9 ms.

7.2. User trials

The second part of our evaluation was to ask a group of fifteen users to experiment with an application that uses the TFD method (SR-Sketch, http://thalis.cs.unipi.gr/~gandreou/sr_sketch/) and record their reactions and comments. SR-Sketch is a prototype application for creating sketches, which utilizes the TFD method to retrieve shapes and add them to the currently working sketch. The user does not have to pick the shape from a list, as the program retrieves the shapes out of a preloaded database that are similar to what s/he is drawing on the screen. None of the users had any experience with a Content-Based Retrieval system before. Ten of these users had to complete a questionnaire that was handed to them. Completion of the questionnaire was followed by a short discussion with the system designers. Appendix B contains the questions and user responses that refer to the TFD method. After discussing with the users and summing the results from their responses to the questionnaire, the following conclusions were reached:

1. Although none of the users had experience with such a shape-retrieval system, all of them immediately grasped this concept and were able to utilize it. One of the reasons that allows inexperienced users to work with the system efficiently after only a small training period was the fact that, the user does not need to configure the shape-matching engine; s/he just uses its functionality through a visual, intuitive interface.
2. The actual success of a shape retrieval method cannot be measured using only precision and recall. Sometimes, a user asks for elements that are similar to a shape and gets search results that may look similar but are not what s/he had in mind. Using the system with some success has proven to be very simple, but one may gain many advantages by understanding the underlying concepts behind the search-engine. The use of the visual explanation process during user trials contributed to the success of the system because it (a) reassured inexperienced users that the engine works correctly and according to a set of rules that are based on the similarity of outline areas of the shapes and (b) helped the interested users understand the internal matching process.

8. Conclusions and future work

This paper described TFD, an efficient and robust method for computing, explaining and visualizing similarity in Content-Based Image Retrieval. All the above properties are proven and are demonstrated through a complete single-object image retrieval system. The method can be augmented with the use of several pre-computed criteria in order to improve speed and effectiveness. A visualization process was introduced, based on the matching information produced by the above method, which explains to the user the basic characteristics of the TFD method and visualizes why and how two shapes are considered similar. Future work in this area will seek to enhance and extend the proposed method, in order to apply it to object recognition in complicated and occluded scenes.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful and illuminating comments. This research was supported by the HERAKLEITOS initiative of the Hellenic Ministry of Education.

Appendix A. Polygon database (PDB) file

PDB is a novel file format for storing shapes that was created for use with the GCV library (http://thalis.cs.unipi.gr/~gandreou/gcv/). A PDB file usually contains:

(a) polygons,
(b) categories for the polygons,
(c) strings that correspond to polygons, and usually contain the path of the image from which each polygon was extracted.

Following is an example of a small PDB file, describing the basic structure of this format. This file contains:

(a) only one polygon, with four vertices;
(b) one category, named 'general', which contains this polygon. This category starts at index 0 and ends (before) index 1;
(c) one string, containing the path of the image from which the polygon was extracted.

```
PDB   //header
sampleSize-1   //if>2, the polygons are ALL pre-sampled at this resolution
polSize 1   // number of polygons in the file
pSize 4   // number of vertices for the first polygon
0 128 131   //vertices, in this and the following lines
1 129 131
2 130 131
3 131 131
catSize 1 // number of categories for shapes
general 0 1 // a shape category, named general, starting at 0, ending at 1
strSize 1 // number of strings, that contain image paths
0 F:/data/images/velt_selected/002.PCT.bmp
```

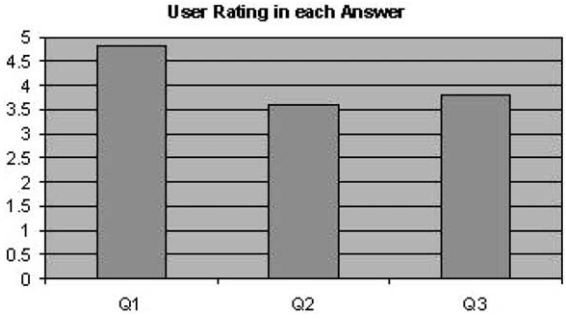
An XML-equivalent of this file exists to facilitate import/export from/to other shape file formats, yet the text structure above is most commonly used, for storage space and parsing time efficiency.

Appendix B. User questions and results

Users questionnaire
Rate the system, by answering the following questions, from 1 (lowest) to 5 (highest):

1) How easily did you learn and use the system?
2) Considering the options available from the selected database, how often do you think the system retrieves the shapes you consider more likely to the drawn shape?
3) How much did the visual explanation system help you understand the (SR-Sketch) system, during the first trials?

B.1. Results summary



The results of the users' evaluation demonstrate that it is very easy to learn to take advantage of the TFD method (Q.1). The users' opinions on the relevancy of the returned shapes (Q.2) vary. This was caused by the fact that some users had more insight of the system than others. Apart from the software that is available at the SR-Sketch site (http://thalis.cs.unipi.gr/~gandreou/sr_sketch/), there are also utilities that view/manage the shape database and other files types used by system, which some of the users had access to. The users that had access to these tools and spent more time on the system gave better ratings on its relevancy. Most users took advantage of the Visual Explanation System—a dynamic image containing a visualization of the TFD matching process—very often, and have given it a good rating (Q.3), as it helped them understand the shape-matching function, during their first trials.

References

Abbasi, S., Mokhtarian, F., & Kittler, F. (1999). Curvature scale space image in shape similarity retrieval. *Multimedia Systems: Vol. 7(6)*, Berlin: Springer.

Andreou, I., & Sgouros, N. M. (2003). Sketch creation utilizing shape matching techniques. In *IEEE international conference on multimedia and expo*. Baltimore, USA.

Andreou, I. (2002a). SR-Sketch application. Available from http://thalis.cs.unipi.gr/~gandreou/sr_sketch/.

Andreou, I. (2002b). The GCV library. Available from http://thalis.cs.unipi.gr/~gandreou/gcv/.

Arkin, E. M. et al. (1991). An efficiently computable metric for comparing polygonal shapes. *IEEE transactions on pattern analysis and machine intelligence*.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence, 24*(24), 509–522.

Del Bimbo, A., & Pala, P. (1997). Shape Indexing by structural properties. In *International conference on multimedia computing and systems, Ottawa, Canada* (pp. 370–377).

Eakins, P. (1996). Automatic image content retrieval—are we getting anywhere? In *Proceedings of third international conference on electronic library and visual information (ELVIRA3)* (pp. 123–135).

Holden, E.-J., & Owens, R. (2003). Representing the finger-only topology for hand shape recognition. *Machine Graphics and Vision Journal, 12*(2), 187–202.

Koller, D., Weber, J., & Malik, J. (1994). *Robust multiple car tracking with occlusion reasoning. ECCV, LNCS 800* (pp. 189–196). Stockholm, Sweden: Springer-Verlag.

Kwan, P. W. H., Kameyama, K., & Toraichi, K. (2003). *On a relaxation labeling algorithm for real-time contour-based image similarity retrieval*. Amsterdam: Elsevier Science.

Lipson, H., & Shpitalni, M. (2002). Correlation-based reconstruction of a 3D object from a single freehand sketch. In *AAAI spring symposium on sketch understanding* (pp. 99–104).

Loncaric, S. (1998). A survey on shape analysis techniques. *Pattern recognition: Vol. 31 (8)*. Amsterdam: Elsevier.

Niblack, W., & Yin, J. (1995). A pseudo-distance measure for 2-D shapes based on turning angle. In *IEEE international conference on image processing*, Washington DC.

Palmer, S. E. (1999). *Vision science: photons to phenomenology*. Cambridge, MA: The MIT Press.

Ulman, S. (1995). *Higher level vision*. Cambridge, MA: MIT Press.

Veltkamp, R. C. (2001). Similarity measures and algorithms. In *International conference on shape modeling and applications (SMI 2001)*, Genova, Italy.

Wolfson, H., & Rigoutsos, I. (1997). Geometric hashing: an overview. *IEEE Computation Science and Engineering*, 10–21.

Zhang, D. S., & Lu, G. (2001). A comparative study on shape retrieval using Fourier descriptors with different shape signatures. In *International conference on intelligent multimedia and distance education (ICIMADE01)* (pp. 1–9). Fargo, ND, USA.