Shape Descriptors based on Tensor Scale

Fernanda A. Andaló, Ricardo da S. Torres, and Alexandre X. Falcão
Institute of Computing – State University of Campinas (Unicamp)
CEP 13084-851 – Campinas – SP – Brazil
{feandalo,rtorres,afalcao}@ic.unicamp.br

Abstract

Tensor scale is a morphometric parameter that unifies the representation of local structure thickness, orientation, and anisotropy, which can be used in several computer vision and image processing tasks. We exploit this concept for binary images and propose two shape descriptors – Tensor Scale Descriptor with Influence Zones and Tensor Scale Contour Saliences. It also introduces a robust method to compute tensor scale, using a graph-based approach – the image foresting transform. Experimental results are provided, showing the effectiveness of the proposed methods, when compared to other relevant methods with regard to their use in content-based image retrieval tasks.

1. Introduction

The recent growth of the World Wide Web and the new technologies that became available for image acquisition and storage have increased the demand for image retrieval systems based on image properties. In content-based image retrieval (CBIR) systems, image processing techniques are used to describe the image content, encoding image properties – shape, color, or texture – that are relevant to the query. These properties are processed by image descriptors that can be characterized by two functions: a feature vector extraction function and a similarity function that computes the similarity between images based on their feature vectors [7].

The shape of an object is an important and basic visual feature for describing image content [16]. Shapes are often the archetypes of objects belonging to the same pattern class, and can be used in a wide range of practical problems, such as document analysis (optical character recognition), internet (content-based image retrieval), security (fingerprint detection), etc [6].

This work focus on shape feature extraction and description for CBIR systems and, for this purpose, we need a parameter for characterizing the structures presented in the images. In [15], Saha introduces a new concept called *tensor scale* – a local morphometric parameter that yields a unified representation of structure size, orientation, and anisotropy. We extend the application of tensor scale, proposing two shape descriptors – Tensor Scale Descriptor with Influence Zones (TSDIZ) and Tensor Scale Contour Saliences (TSCS). We also present a much faster tensor scale computation, as compared to previous methods [15, 14], by exploiting the Euclidean Image Foresting Transform (Euclidean IFT) [10]. This new tensor scale algorithm can also be applied to the solution of other problems not related to image description, such as clustering, classification, image filtering and image registration.

The complete work[1] is available at [1] and its related publications are [2, 3].

This paper briefly overviews the methods as follows. Section 2 describes the previous methods for computing tensor scale and also the new method proposed for binary images. Section 3 describes the proposed shape descriptors and Section 4 shows the related experiments and results. Finally, Section 5 concludes this paper.

2. Tensor scale

In [15], Saha introduced the *tensor scale* of a pixel p in a gray-scale image as the largest ellipse within the same homogeneous region, centered at p. The homogeneous region is defined based on small differences between the pixels' intensity.

Tensor scale defines the ellipse by three factors:

- $Orientation(p) =$ angle between $t_1(p)$ and the horizontal axis;
- $Anisotropy(p) = \sqrt{1 - \frac{|t_2(p)|^2}{|t_1(p)|^2}}$;
- $Thickness(p) = |t_2(p)|$;

1 This paper contains extracts of the M.Sc. dissertation named "Descritores de forma baseados em tensor scale" by Fernanda A. Andaló.

where $|t_1(p)|$ and $|t_2(p)|$, with $|t_1(p)| \geq |t_2(p)|$, denote the length of the two semi-axis of the ellipse centered at p. Figure 1 illustrates the components to compute each one of these factors.

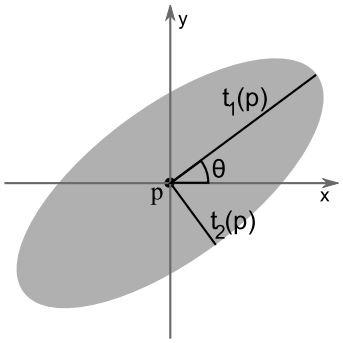


Figure 1. Tensor scale factors.

In the following subsection, we describe two previously proposed tensor scale computation methods for gray-scale images [15, 14]. After that, we provide a faster tensor scale computation for binary images.

2.1. Tensor scale for gray-scale images

In Saha's approach [15], a tensor scale ellipse is calculated from sample lines that are traced around a given pixel, from 0 to 179 degrees (Figure 2(a)). The axes of the ellipse are determined by computing the image intensities along each of the sample lines and the localization of two opposite edge points on these lines (Figure 2(b)). The next step consists of repositioning the edge locations to points equidistant to that given pixel, following the axial symmetry of the ellipse (Figure 2(c)). The computation of the best-fit ellipse to the repositioned edge locations is done by Principal Component Analysis (PCA) (Figure 2(d)).

These computations are performed for every pixel of the image and a critical drawback is that the computational cost of the algorithm makes the method quite prohibitive for more complex tasks, such as image description for content-based image retrieval. For this reason, Miranda et al. [14] proposed an efficient implementation of the original method, which differs in the following aspects.

The first change was in the localization of the edge points. Miranda's approach proposes to go along each pair of opposite segments, alternately and at the same time, instead of going along one entire segment by turn. By doing this, the reposition phase is no longer necessary. The second change was the use of two connected thresholds to improve and simplify the original method of detecting edges. The third and final change was the improvement of the ellipse computation phase. Miranda et al. proposed a function g (Equation 1) that gives the angle of the ellipse directly, instead of using PCA. The ellipse orientation is obtained from the value of γ that minimizes the function

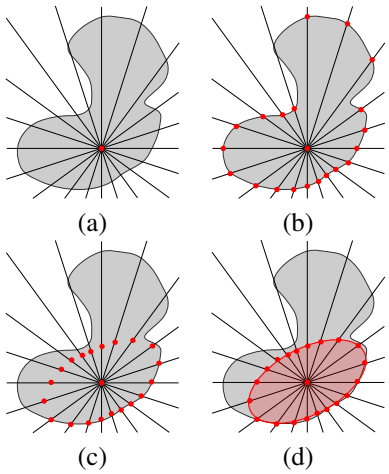


Figure 2. Tensor scale Computation.

$$g(\gamma) = \sum_{i=1,2,\ldots,2m} [x_{i_\gamma}^2 - y_{i_\gamma}^2], \qquad (1)$$

where $x_{i_\gamma} = x_i \cos(\gamma) - y_i \sin(\gamma)$, $y_{i_\gamma} = x_i \sin(\gamma) + y_i \cos(\gamma)$, (x_i, y_i) are the relative coordinates of the edge points with respect to the center pixel $p = (x_p, y_p)$ of the ellipse, and $(x_{i_\gamma}, y_{i_\gamma})$ are the new coordinates after applying a rotation by the angle γ.

In the next subsection, we provide a faster tensor scale computation, as compared to the described approaches [15, 14], by exploiting the Euclidean IFT.

2.2. Tensor scale for binary images

The improvements achieved by Miranda et al. algorithm are important for computing tensor scale in gray-scale images. However, for binary images the tensor scale computation can be further improved.

The first simple change consists of the elimination of the thresholds used for edge detecting. These thresholds are not necessary, because the images have already been segmented. Furthermore, the method can incorporate techniques to easily find the edges in the directions of the sample lines. Such techniques comprehend the use of Euclidean Distance Transform, computed by the Image Foresting Transform, that is described in the next subsection.

2.2.1. Euclidean Distance Transform via Image Foresting Transform The Image Foresting Transform (IFT) is a graph-based approach to the design of image processing operators based on connectivity, in which the images are represented by graphs – the pixels are considered as nodes and

the arcs are defined by an adjacency relation between pixels. For a given seed set (roots), the seeds compete with each other, defining influence zones. Each influence zone consists of pixels that are "more closely connected" to a seed than to any other, according to a path-cost function [10]. We use a path-cost function that assigns the closest Euclidean distance between object pixels and contour pixels to each pixel inside the object (Euclidean IFT).

In the Euclidean IFT (Algorithm 1), the path-cost function is such that the cost of a path from a seed s to a pixel t in the forest is the Euclidean distance between s and t. The algorithm also needs an Euclidean relation A defined as

$$q \in A(p) \Rightarrow (x_q - x_p)^2 + (y_q - y_p)^2 \leq \rho^2$$

where ρ is the adjacency radius and (x_i, y_i) are the coordinates of a pixel i in the image.

Algorithm 1 assigns to each object pixel p three attributes: the squared Euclidean distance $C(p)$ between p and its closest point s in the contour (forming an optimum cost map), its closest seed $R(p) = s$ (forming a root map), and the label $L(p) = L(s)$ of the segment that contains s (forming a label map).

The advantages of computing the Euclidean Distance Transform via IFT is that label propagation is executed on-the-fly and in linear time. The Euclidean IFT is used for two purposes in the proposed methods: faster tensor scale computation, that is described in the next subsection, and tensor scale orientation mapping (Section 3.1).

2.2.2. Tensor scale computation via Image Foresting Transform A considerable speed up in the computation of the tensor scale for binary images is possible by exploiting the following aspect: if we have the shortest distance between a pixel p and the contour, there is no need to search for edge points inside the circle with radius $\sqrt{C(p)}$ (Figure 3(a)). For every pixel p, this distance can be obtained from the cost attribute $C(p)$ returned by Euclidean IFT.

According to Miranda's algorithm, edge points are searched along opposite sample lines, alternately. However, in our approach, the algorithm jumps along the lines and visits the pixels q and r at the same time (Figure 3(b)). The searching for edge points continues outside the area defined by the cost $\sqrt{C(p)}$ in Figure 3(b), and the minimum between $\sqrt{C(r)}$ and $\sqrt{C(q)}$ indicates the location for the next jump. These jumps may continue iteratively until the closest edge point along the sample line is found.

In the example, the edge is found at the pixel $R(r)$ (i.e., at the contour point r' nearest to r). The algorithm defines that the two edge points in this sample line are at r' (coordinate of $R(r)$ relative to p) and at q' (coordinate of the point diametrically opposite to r', relative to p), as shown in Figure 3(c).

Algorithm 1:
Input: A binary image I, a set S of contour points or segments in I (seeds), an Euclidean adjacency relation A, and a labeling function $\lambda(p)$ that assigns a label to each point or segment p in S.
Output: The cost map C, the root map R, and the label map L.
Auxiliary data structure: A priority queue Q.

foreach $p \in I$ **do**
  $C(p) \leftarrow +\infty$;
  $R(p) \leftarrow NIL$; $L(p) \leftarrow NIL$;
foreach $p \in S$ **do**
  $C(p) \leftarrow 0$;
  $R(p) \leftarrow p$;
  $L(p) \leftarrow \lambda(p)$;
  insert p in Q;
while Q is not empty **do**
  remove from Q a pixel $p = (x_p, y_p)$ such that $C(p)$ is minimum;
  foreach $q = (x_q, y_q)$ such that $q \in A(p)$ and $C(q) > C(p)$ **do**
    $C' \leftarrow (x_q - x_{R(p)})^2 + (y_q - y_{R(p)})^2$, where $R(p) = (x_{R(p)}, y_{R(p)})$ is the root pixel of p;
    if $C' < C(q)$ **then**
      if $C(q) \neq +\infty$ **then**
        remove q from Q;
      $C(q) \leftarrow C'$;
      $R(q) \leftarrow R(p)$;
      $L(q) \leftarrow L(p)$;
      insert q in Q;

By performing this procedure for all sample lines, the algorithm defines all edge points and uses the same formula defined by Miranda et al. (Equation 1) for finding the orientation of the ellipse.

The localization of the edge points is formalized in Algorithm 2.

3. Shape descriptors

The key idea of the proposed methods is to compute the tensor scale ellipse for every object point and to map the orientations onto the object's contour. The tensor scale computation is done by the proposed method described in the previous section. The orientation mapping is described in the next subsection.

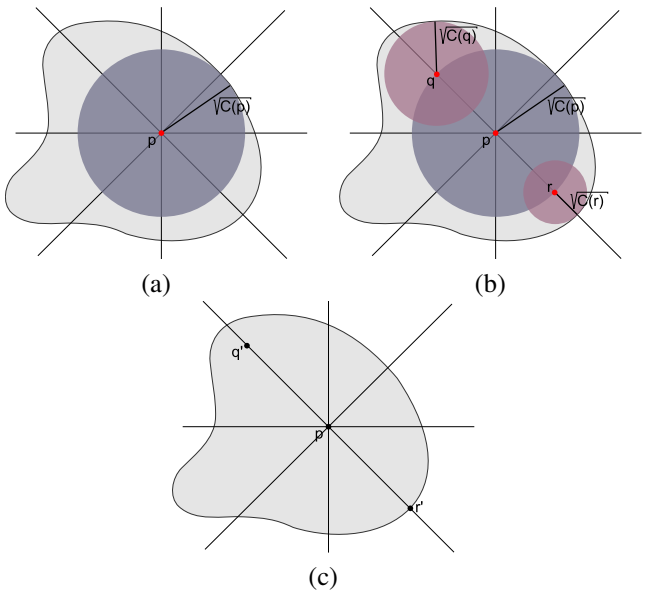


Figure 3. Example of optimization by using Euclidean IFT.

3.1. Orientation mapping

The orientation mapping (Algorithm 3) consists in mapping the tensor scale ellipse orientation (computed for all the object pixels) onto the object's contour. Although the tensor scale orientation mapping is different for each proposed method, they both exploit the discrete Voronoi regions (influence zones) of contour points inside the object. The influence zones can be efficiently obtained by label propagation using the Euclidean IFT [10].

It is necessary to compute the discrete Voronoi regions inside the object and summarize, in some way, the orientation information contained in each of these regions. The introduced methods require two different types of orientation mapping: one that assigns information to each contour point and one that assigns information to contour segments. The former can be understood as an instance of the latter with one-point length segments.

The orientation mapping (Algorithm 3) requires label map L returned by the Euclidean IFT (Algorithm 1), using the segments as seeds and a labeling function that assigns a different label to each segment. Map L groups pixels and their ellipses in the influence zone of each segment.

The meta-function $Summary(V[i])$ returns a summarization of ellipses information inside the correspondent influence zone. Each method has its own summary function and they will be discussed in the next subsections.

Algorithm 2:
Input: A pixel $p = (x_p, y_p)$, the number m of sample lines, and the cost map C returned by Algorithm 1.
Output: The vector $edge$ that contains m pairs of edge points localized at the sample lines.

for $\theta \leftarrow 0°$ **to** $179°$, with increments $\frac{180}{m}$ **do**
- $v \leftarrow \sqrt{C(p)}$;
- $p_1 \leftarrow NIL$;
- $p_2 \leftarrow NIL$;
- $q_1 \leftarrow 0$;
- $q_2 \leftarrow 0$;
- **while** $p_1 \neq 0$ and $p_2 \neq 0$ **do**
  - $x \leftarrow v * \cos(\theta)$;
  - $y \leftarrow v * \sin(\theta)$;
  - **if** $q_1 = 0$
    - $temp \leftarrow (x_p + x, y_p + y)$;
    - $p_1 \leftarrow \sqrt{C(temp)}$;
  - **if** $q_2 = 0$
    - $temp \leftarrow (x_p - x, y_p - y)$;
    - $p_2 \leftarrow \sqrt{C(temp)}$;
  - $d \leftarrow min(p_1, p_2)$;
  - $v \leftarrow v + d$;
  - $q_1 \leftarrow p_1 - d$;
  - $q_2 \leftarrow p_2 - d$;
- $edge[\theta] \leftarrow ((x, y), (x', y'))$, where (x', y') is the coordinate of the point diametrically opposite to (x, y), relative to p;

3.2. Tensor Scale Contour Saliences (TSCS)

The saliences of a shape are defined as the higher curvature points along the shape contour [9], or vertex points along the contour with first derivative discontinuity [6]. Their detection is the key to various applications in image processing (e.g., image registration, polygonal approximation, motion analysis, and shape description [8]).

The TSCS method consists of three steps: the tensor scale computation for all pixels inside a given object, followed by the mapping of such tensor scale orientations onto each contour point, as described in Algorithm 3, and finally the detection of saliences based on the mapped orientations. These saliences are used to form the feature vector of the TSCS descriptor.

In this case, the $Summary(V[i])$ function of Algorithm 3 returns the orientation of the ellipse with maximum anisotropy inside the influence zone with label i:

Algorithm 3:
Input: A binary image containing an object O, the number n_s of contour segments, the label map L returned by the Euclidean IFT, and the vectors Ani and Ori that contain the anisotropies and the orientations of the tensor scale ellipses computed for all pixels of O, respectively.
Output: A vector M that contains the mapped orientations for each contour segment.
Auxiliary data structure: A vector V of n_s lists to store ellipse information in each influence zone of segment.
foreach pixel $p \in O$ **do**

insert $(Ani[p], Ori[p])$ in list $V[L(p)]$, where $L(p)$ is the label of the influence zone in which p is contained;

foreach $i \in [1, \ldots, n_s]$ **do**

$M[i] = Summary(V[i])$;

$$Summary(V[i]) = \underset{p}{\operatorname{argmax}}\, Ani[p], (Ori[p], Ani[p]) \in V[i],$$

where the vectors Ani and Ori contain the anisotropies and the orientations of the tensor scale ellipses computed for all the object pixels.

Therefore, the vector M contains n_s values, where n_s is the number of points in the object's contour. Contour points with no influence zone inside the object borrow the orientations of the neighbors.

In order to localize the salience points, the method calculates the differences between adjacent mapped orientations in M. The difference value at $p \in M$ is

$$Difference(p) = AngularDist(M(p-1), M(p+1)),$$

where the function $AngularDist(\alpha, \beta)$ gives the smallest angle between the orientations α and β.

Now, the method uses a threshold value to eliminate low values of difference along the contour. Figure 4(a) shows the adopted threshold and the difference values computed for every contour point of the shape illustrate in Figure 4(b). Figure 4(c) shows another example of the detected saliences (dots) using threshold 16, i.e, saliences related to angle differences lower than 16° were not represented.

After the salience detection phase, the descriptor is formed by the feature extraction and metric functions used in CS [8].

3.3. Tensor Scale Descriptor with Influence Zones (TSDIZ)

The key idea of the Tensor Scale Descriptor with Influence zones (TSDIZ) is to map the tensor scale orientations

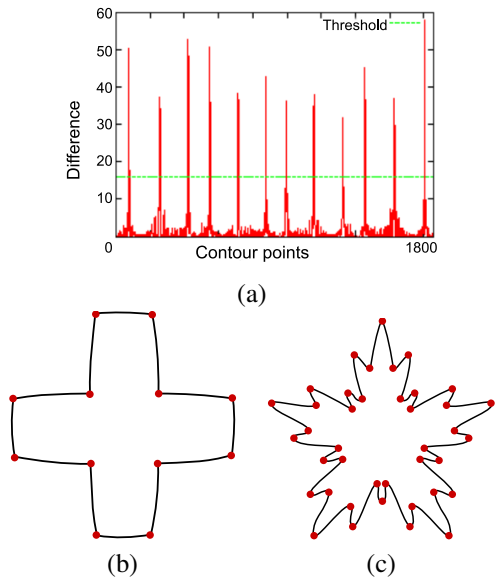


Figure 4. Difference values along a contour and detected saliences.

inside an object onto a few segments of its contour, and use this information for shape description.

First, the TSDIZ approach computes tensor scale for all pixels inside an object. Next, it divides the object's contour into segments and maps the tensor scale orientation onto each contour segment, as described in Algorithm 3.

For TSDIZ, the $Summary(V[i])$ function returns the weighted angular mean [13] of the ellipses orientations contained in the influence zone with label i, considering the anisotropies as the weights:

$$Summary(V[i]) = arctan\,(K)\,,$$

$$K = \frac{\sum_{(Ori[p], Ani[p]) \in V[i]} Ani[p] * \sin(2Ori[p])}{\sum_{(Ori[p], Ani[p]) \in V[i]} Ani[p] * \cos(2Ori[p])}.$$

The vector M returned by Algorithm 3 is used as TSDIZ feature vector and contains n_s values, where n_s is the number of contour segments.

The similarity function has to determine the rotation difference of the orientations between two TSDIZ vectors. This function also has to determine the segment in which the feature vectors must be lined up to obtain the best matching between the underlying shapes.

The exhaustive algorithm (Algorithm 4) consists of the registration between the orientation feature vectors. For this purpose, the algorithm computes, for each rotation α, where $\alpha = 0^\circ, \ldots, 179^\circ$, and for each shift j in the feature vector, where $j = 1, \ldots, n_s$ and n_s is the size of the vectors,

the difference between the vectors, after rotating all orientations of one vector by α and circular shifting the same vector by j. The minimum difference obtained corresponds to the distance between the vectors.

Algorithm 4:
Input: Two feature vectors F_A and F_B.
Output: Distance $dist$ between F_A and F_B.

$dist \leftarrow \infty$;
foreach $j \in [1, \ldots, n_s]$ **do**
  foreach $\alpha \in [0, \ldots, 179]$ **do**
    foreach $i \in [1, \ldots, n_s]$ **do**
      $dist_{aux} \leftarrow AngularDistance(\{F_B[(j - i) \bmod n_s] + \alpha\} \bmod 180, F_A[i])$;
      if $dist_{aux} < dist$ **then**
        $dist \leftarrow dist_{aux}$;

The complexity of this algorithm is $O(c * {n_s}^2)$, where c is a constant (in this case, 179°). Although it is an exhaustive search, small values of n_s (e.g., $n_s < 70$) makes it still fast. Figure 5 illustrates the registration between two TSDIZ vectors. An orientation curve is computed for each object and then, applying the matching algorithm, these curves can be matched.

4. Experimental results

Experiments were conducted using two databases: Fish-shape[2] and MPEG-7[3] part B.

4.1. TSCS results

The experiments are based on comparisons between the TSCS and the Contour Salience (CS) method [8], because of its interesting previous results. But, before comparing the different approaches, we need to find the best threshold for our method. For this purpose, we constructed a database consisting of 42 shapes of the Fish-shape database and 112 shapes of the MPEG-7 Part B database, resulting in 2835 saliences. The images were chosen by taking into account the obviousness of the contour salience points location. Then, a set of ground truth images were constructed with the location of the salience points.

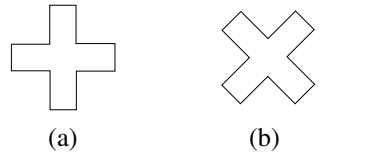
This experiment relies on counting the true positive saliences (T_+) and false positive saliences (F_+) for the ground truth images. After this counting, three effectiveness measures were calculated: recall, precision, and accuracy. Recall (Rec) and precision (Prec) are computed as

$$Rec = \frac{T_+}{T_+ + T_-}$$

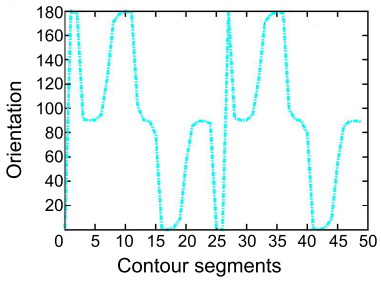
and

$$Prec = \frac{T_+}{T_+ + F_+}$$

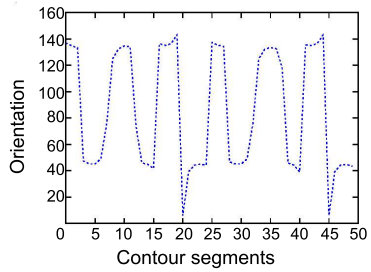
where T_- is the number of true negatives, and (T_+ + T_-) represents the total number of points. The accuracy (Acc) is calculated as



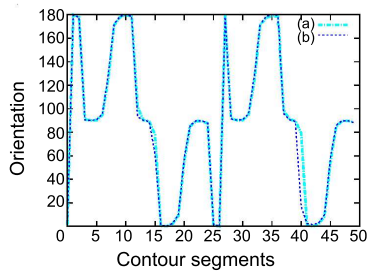
(a) (b)



(c) Orientation curve for (a).



(d) Orientation curve for (b).



(e) Curve matching.

Figure 5. Examples of TSDIZ curves and registration.

2 http://www.ee.surrey.ac.uk/research/vssp/imagedb/demo.html
3 http://www.chiariglione.org/mpeg/

Measures	10	12	14	16	18
Recall	0.964	0.964	0.963	**0.963**	0.962
Precision	0.889	0.923	0.946	**0.963**	0.968
Accuracy	0.840	0.862	0.874	**0.875**	0.867

Table 1. Effectiveness measures.

$$Acc = \frac{T_+ + T_-}{T_+ + T_- + F_+ + F_-}$$

where F_- (false negatives) represents the number of miss-detections.

The results with different threshold values (from 10 to 18) are presented in Table 1. Note that the method is robust to the choice of the threshold. However, the accuracy was maximized with threshold value 16 and this is the value adopted for this method in further experiments.

The first consideration made between the approaches was related to performance issues. Our method was twice faster (speedup of 2.04), on average, than the CS approach, when executed for the entire Fish database (on an AMD 64 3000+ with 1GB of memory).

The second consideration is that our method is computed locally, looking for each mapped orientation and for its neighbors along the contour. The CS method is more global, because it uses the internal and external skeletons of the whole shape for salience detection. This difference in granularity also makes the detection of saliences less robust in the CS approach, because the multiscale skeletons have to be thresholded to obtain salience points. This threshold represents a smoothing of the contour and, consequently, loss of some important saliences. In order to detect these saliences, we would have to reduce the threshold. Our method is also dependent of a threshold, but it is much easier to fix a single threshold for the entire database, which is the case of our approach, than to find the best threshold for every single image in the database, which is the case of the CS approach.

The last consideration is about the impact of a better salience estimation in shape description. Corners and high curvature points concentrate more information than other points of the shape [5]. For this reason, it is intuitive to conceive that curvature is an important key for the identification of many geometric aspects. Based on this, we use the saliences as key points for shape description.

We compared the descriptors using the multiscale separability (MS separability) effectiveness measure. Separability indicates the discriminatory ability between objects that belong to distinct classes [8]. The TSCS and CS descriptors were computed for Fish-shape database and the MS separability curves for the descriptors are shown in Figure 6. Higher is the curve, better is the method.

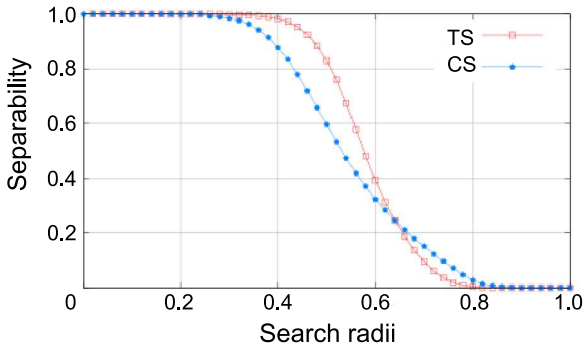


Figure 6. Multiscale separability curve for Fish database.

By analyzing Figure 6, we observe that TSCS is more effective or equal to CS in 80% of the search radii.

4.2. TSDIZ experiments

In [8], Torres et al. showed that MS separability represents better than precision vs. recall (PR) curves the separation among clusters (groups of relevant images) in the feature space. However, PR is still the most popular effectiveness measure in CBIR. For this reason, we present the results with both measures.

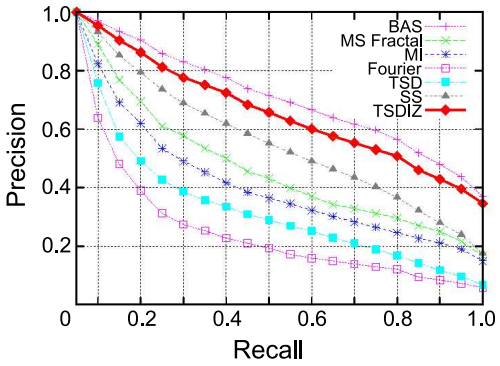
Precision is defined as the fraction of retrieved images that are relevant to the query. In contrast, recall measures the proportion of relevant images among the retrieved images. The Precision vs. Recall curve, or PR curve, indicates the commitment between the two measures and, generally, the highest curve in the graph indicates better effectiveness.

In this experiment, TSDIZ is compared with the following shape descriptors: Beam Angle Statistics [4] (BAS), Multiscale Fractal Dimension [9] (MS Fractal), Moment Invariants [12] (MI), Fourier Descriptor [11] (Fourier), Tensor Scale Descriptor [14] (TSD), and Segment Saliences [8] (SS).

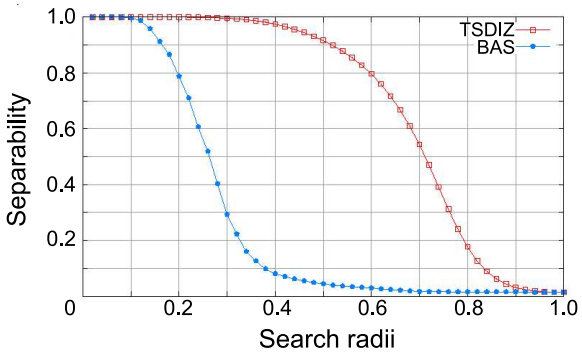
Figure 7(a) presents the PR curves for the evaluated descriptors and TSDIZ with 60 contour segments. We have tested different number of segments (30 to 120) and the method is quite robust to this choice.

TSDIZ descriptor has the second better PR curve among the tested descriptors. As TSDIZ has outperformed all other descriptors for MS separability as well, we show in Figure 7(b) the MS separability curves of TSDIZ and BAS only.

TSDIZ and BAS present equivalent effectiveness for search radii less than 10% of their maximum distance. From this point on, the BAS separability curve decreases quickly,



(a) PR curve.



(b) MS separability.

Figure 7. TSDIZ experiments.

indicating that this descriptor is neither robust nor effective for search radii greater than 20%.

Table 2 shows a visual CBIR example for a query image. The images that are not in the same class of the query image and should not be returned by the query are shown with a border around them.

5. Conclusions

This paper describes a faster algorithm for tensor scale computation in binary images using Image Foresting Transform (IFT), and two shape descriptors based on tensor scale.

For the TSCS descriptor, the experimental results showed that it is faster and more robust than the CS descriptor. The experiments for TSDIZ indicate that this descriptor has better PR curve than all relevant shape descriptors (except BAS) and the best separability among them, making it the most robust and effective according to this metric.

Acknowledgments

Authors are grateful to FAEPEX, FAPESP, CAPES, CNPq, and Microsoft.

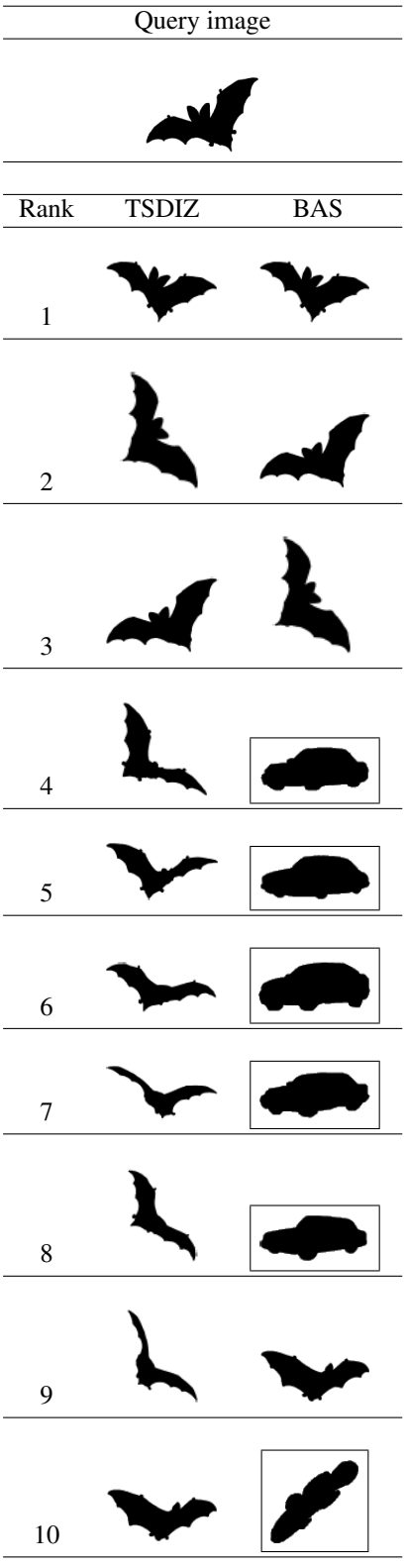


Table 2. Visual CBIR example.

References

[1] F. A. Andaló. Descritores de forma baseados em tensor scale. IC – Unicamp, 2007. http://www.liv.ic.unicamp.br/~feandalo/pubs/msc.pdf.

[2] F. A. Andaló, P. A. V. Miranda, R. da S. Torres, and A. X. Falcão. Detecting contour saliences using tensor scale. In *Proceedings of the 14th IEEE International Conference on Image Processing*, volume 6, pages 349–352, September 2007.

[3] F. A. Andaló, P. A. V. Miranda, R. da S. Torres, and A. X. Falcão. A new shape descriptor based on tensor scale. In *Proceedings of the 8th International Symposium on Mathematical Morphology*, pages 141–152, October 2007.

[4] N. Arica and F. T. Y. Vural. BAS: a perceptual shape descriptor based on the beam angle statistics. *Pattern Recognition Letters*, 24(9–10):1627–1639, 2003.

[5] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954.

[6] L. da F. Costa and J. R. M. Cesar. *Shape analysis and classification: theory and practice*. CRC Press, Florida, USA, 2001.

[7] R. da S. Torres and A. X. Falcão. Content-based image retrieval: theory and applications. *Revista de Informática Teórica e Aplicada*, 13(2):161–185, 2006.

[8] R. da S. Torres and A. X. Falcão. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3–13, 2007.

[9] R. da S. Torres, A. X. Falcão, and L. da F. Costa. A graph-based approach for multiscale shape analysis. *Pattern Recognition*, 37(6):1163–1174, 2004.

[10] A. X. Falcão, J. Stolfi, and R. A. Lotufo. The image foresting transform: theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[11] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Addison-Wesley, 2nd edition, 2001.

[12] M. K. Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2):1163–1174, 1962.

[13] K. V. Mardia and P. E. Jupp. *Directional statistics*. John Wiley and Sons, 1999.

[14] P. A. V. Miranda, R. da S. Torres, and A. X. Falcão. TSD: A shape descriptor based on a distribution of tensor scale local orientation. In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, 2005.

[15] P. K. Saha. Tensor Scale: A local morphometric parameter with applications to computer vision and image processing. *Computer Vision and Image Understanding*, 99:384–413, 2005.

[16] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, 2004.