A Convexity Measure for Open and Closed Contours

Padraig Corcoran[1]
padraig.corcoran@ucd.ie
Peter Mooney[2]
Peter.Mooney@nuim.ie
Adam Winstanley[2]
adam.winstanley@nuim.ie

[1] School of Computer Science and Informatics,
University College Dublin,
Ireland.

[2] Department of Computer Science,
National University of Ireland Maynooth,
Ireland.

Abstract

Convexity represents a fundamental descriptor of object shape. This paper presents a new convexity measure for both open and closed simple contours. Given such a contour this measure extracts two corresponding open convex hulls. The shape similarity between these two hulls and the original contour is then computed and normalized to give a measure of convexity. The time complexity of the proposed technique is $O(n)$. The authors believe this technique represents the first measure of convexity which uses shape similarity and which can be applied to both open and closed contours. The proposed technique is shown to provide similar or greater performance relative to two other state of the art techniques.

1 Introduction

Object recognition represents an extremely powerful capability of the Human Visual System (HVS). It has been shown that shape is the single most important feature used by the HVS to recognize objects [3]. Many general shape descriptors exist, such as Fourier descriptors and moments, which provide a feature vector of high dimensionality capable of accurately describing shape. Alternatively many shape descriptors exist which measure a single characteristic of shape [16]. Such characteristics include circularity [12], rectangularity, triangularity [13], rectilinearity [21] and convexity [14, 22, 23]. In machine vision research object contours are commonly extracted from images using an edge detection and linking strategy [19]. Due to occlusion, scene complexity and image noise, contour extraction techniques do not necessarily always return a closed object contour. Instead, in many cases, a set of open contours corresponding to object parts are returned. This does not represent an obstacle to recognition in the HVS as psychophysical studies have shown that recognition can be successfully achieved by partial contours alone [3]. As such, many methods exist to accurately describe the shape of open contours so that inference regarding object class can be determine [4, 17, 20]. From the above discussion it should be clear that in order to perform contour based object recognition one must be able to accurately describe the shape of both open and closed contours.

In this paper we focus exclusively on the shape characteristic of convexity. Convexity represents an important descriptor of shape for many reasons. It is generally accepted that the parts of an object's contour which exhibit high convexity generally correspond to object parts [7]. Liu et al. [10] demonstrated, through the use of several psychophysical experiments, that convexity plays a strong role in perceptual organization. They showed that in some cases convexity dominates the effects of Gestalt properties, such as good continuation. Convexity is also a nonaccidential property which can distinguish structures from noise in real images [6]. Although many convexity measures exist for closed contours the work of Zunic et al. [23] represents the only measure suitable for open contours. However we will illustrate in the results section of this paper that this measure suffers from a degenerate case and is computationally expensive to compute.

Most convexity measures can generally be classified as area or perimeter based methods [15]. Since open contours do not enclose a region, area based methods cannot be applied. The most commonly used perimeter based method compares the perimeter length to that of its convex hull. In this paper we propose a new perimeter based convexity measure called *SC* which determines convexity by determining the shape similarity between the open contour in question and its corresponding open convex hull. The authors believe this is the first time shape similarity has been used to determine convexity. It is motivated by the following example. Consider the closed contour and its corresponding convex hull displayed in Fig. 1(a). The shape similarity between this contour and its corresponding convex hull is a function of its convexity. If the contour is convex its corresponding convex hull will have an identical shape. On the other hand if the contour is not convex the degree of shape similarity will be a function of convexity. The computation of *SC* requires three steps. Firstly the convex hull of the open contour is computed and from this two open hulls are extracted. Next the shape similarity between both open hulls and the original contour is determined. Finally the minimum of these values is normalized to give a measure of convexity. *SC* has the following desirable properties [22]:

1. *SC* is a number from $(0,1]$;
2. *SC* equals 1 if and only if the contour is convex;
3. There are shapes for which *SC* approaches 0;
4. *SC* is invariant under similarity transformations (translations, rotations and scaling).

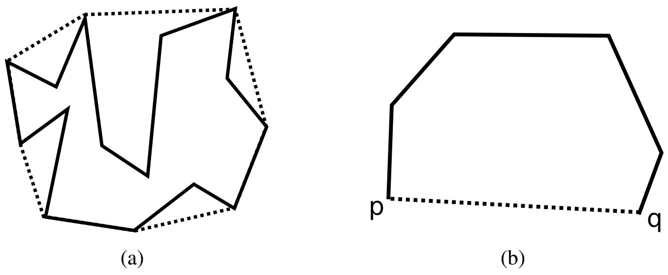


Figure 1: Closed and open contours (solid lines) and their corresponding convex hulls (dotted lines) are displayed in (a) and (b) respectively.

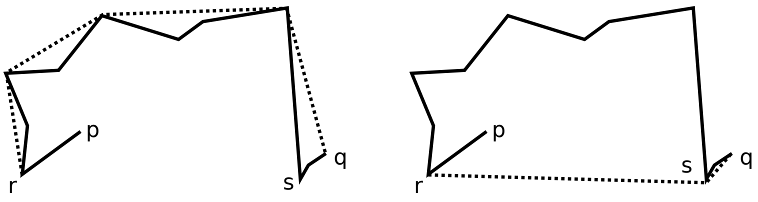
The layout of this paper is as follows. Section 2 describes the proposed technique for extracting the open convex hull corresponding to the open contour in question. Section 3 describes how shape similarity is computed and normalized to give the *SC* measure. Section

4 evaluates the accuracy of *SC* relative to other convexity measures. Finally in Section 5 we draw conclusions.

2 Convex Hull Extraction

Consider the open contour and its corresponding convex hull in Fig. 1(b). Although this open contour is convex it does not have identical shape to its corresponding closed convex hull. This is due to the fact that we are comparing an open contour to a closed contour. To overcome this difficulty the subset of the closed convex hull corresponding to the open contour must be extracted before shape similarity is determined. In Fig. 1(b) this can be achieved by extracting the subset of the closed convex hull which contains the vertices from p to q listed in clockwise order. Extraction of such a subset presents two challenges. Firstly the desired subset may not contain all vertices of the corresponding closed convex hull. To illustrate this consider the open contour in Fig. 2(a) and its corresponding desired open convex hull. This open convex hull does not contain the vertex s which would appear on the corresponding closed convex hull. Secondly for every open contour two corresponding open convex hulls exist. For example the second open convex hull corresponding to the above example is shown in Fig. 2(b). In order to compute a convexity measure it must be determined which open convex hull this calculation should be a function of. In the above example, clearly any convexity measure should be a function of the open convex hull in Fig. 2(a).

To overcome both of these issues we propose a novel method for extracting the correct subset of a closed convex hull. For a given open contour this method extracts two subsets of the convex hull which correspond to a clockwise (CW) and counter-clockwise (CCW) ordering of vertices. These are referred to as the CW and CCW hulls respectively. The similarity between the original contour and both hulls is then determined and the most similar is then used to derive a measure of convexity. The CW and CCW hulls are extracted using a process of two steps which we now describe.



(a) (b)

Figure 2: A open contour (solid lines) and its corresponding CW and CCW hulls (dotted lines) are displayed in (a) and (b) respectively.

Firstly a modified version of the on-line convex hull algorithm of Melkman [11] is applied. The original algorithm of Melkman is shown in Algorithm 1; this pseudocode in based on that of [18]. The function isLeft(p_1, p_2, p_3) returns a value of *True* if p_3 is left of the line through p_1 and p_2; returns a value of *False* if p_1, p_2 and p_3 are collinear and returns a value *False* if p_3 is right of the line through p_1 and p_2. As input the algorithm takes a set of vertices $W = \{v_1, v_2, \ldots, v_n\}$ which define a contour. A deque data structure $D = \left\{d_{bot}, \ldots, d_{top}\right\}$ is used to store those elements of W which form the convex hull in a

CCW order. The algorithm initializes D so that the first three vertices of W form a CCW triangle (lines 2-9). Next it sequentially processes each of the remaining vertices v_i in order (lines 10-22). Each of these vertices will satisfy one of two conditions. Firstly v_i may lie inside the current hull in which it case processing proceeds to the next vertex (lines 11-13). Secondly v_i may lie outside the current hull and becomes a new hull vertex extending the previous hull. In this case vertices on the previous hull may become interior to the new hull and must be discarded (lines 14-16 and 18-20). Once such vertices have been discarded v_i is added to form the new hull (lines 17 and 21).

Algorithm 1 Melkman Convex Hull Algorithm

```
1:  Input: $W = \{v_1, v_2, \ldots, v_n\}$
2:  D.push_back($v_3$)
3:  **if** isLeft ($v_1, v_2, v_3$) **then**
4:      D.push_back($v_1$)
5:      D.push_back($v_2$)
6:  **else**
7:      D.push_back($v_2$)
8:      D.push_back($v_1$)
9:  **end if**
10: **for** $i = 4$ to $n$ **do**
11:     **if** ( (isLeft($d_{bot}, d_{bot+1}, v_i$) $> 0$) & (isLeft($d_{top-1}, d_{top}, v_i$) $> 0$) ) **then**
12:         continue
13:     **end if**
14:     **while** isLeft($d_{bot}, d_{bot+1}, v_i$) $\leq 0$ **do**
15:         D.pop_front()
16:     **end while**
17:     D.push_front($v_i$)
18:     **while** isLeft($d_{top-1}, d_{top}, v_i$) $\leq 0$ **do**
19:         D.pop_back()
20:     **end while**
21:     D.push_back($v_i$)
22: **end for**
```

Algorithm 2 Operations on deque D

```
1: **function** push_back_D($v_k$)
2:     D.push_back($v_k$)
3:     VL.push_back($k$)
4:     $vp_k = vp_k + 1$

5: **function** pop_back_D()
6:     D.pop_back()
7:     VL.pop_back()
8:     $vp_{vl_{top}} = vp_{vl_{top}} - 1$
```

Melkman's original algorithm returns the vertices forming the convex hull listed in CCW order. We altered this algorithm to return the list of vertices listed in the order in which they appear in the original contour. Consider the simple (non-self-intersecting) contour $\{v_1, v_2, \ldots, v_8\}$ in Fig. 3. Melkman's algorithm returns the convex hull vertices in the CCW order $\{v_3, v_2, v_4, v_6, v_8, v_7, v_3\}$. Our first alteration to Melkman's algorithm returns the vertices of the convex hull in the same ordering as the original contour; that is $\{v_2, v_3, v_4, v_6, v_7, v_8\}$. To achieve this we introduced two additional data structures VP and VL. $VP = \{vp_1, vp_2 \ldots, vp_n\}$ is list which has the invariant property $vp_i > 0$ if and only if v_i is a point on the current hull. The second data structure $VL = \{vl_{bot}, \ldots, vl_{top}\}$ is a deque which is used to maintain the invariance of VP. There is a one-to-one correspondence between elements of VL and D. This property is used to maintain the correspondence between elements of D and W; that is, $d_p = v_q$ if and only if $vl_p = q$. VP and VL are integrated into the algorithm of Melkman using new versions of the functions push_back(v_k), push_front(v_k), pop_front() and pop_back(). These new functions are entitled push_back_D(), push_front_D(v_k), pop_front_D() and pop_back_D() respectively. Algorithm 2 describes two of these functions while the remaining two have a similar form. When Algorithm 1 begins both D and VL are empty and $vp_i = 0$ for $i = 1 \ldots n$. Following calculation of the convex hull the vertices listed in the same order in which they appear in the original chain can be obtained by removing all elements v_i from W for which $vp_i == 0$.

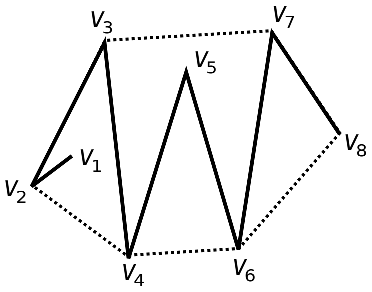


Figure 3: A simple open contour and its corresponding convex hull are displayed.

Given a list of convex hull vertices having an order equal to the original contour we now extract the CW and CCW hulls. This step is inspired by the algorithm of Graham [5] for calculating the convex hull of a set of planar points. To extract the CW hull we process the convex hull vertices in the order they appear in the original contour and remove those which do not make a right turn. To extract the CCW hull we process the convex hull vertices in the same order and remove those which do not make a left turn. For example the CW hull extracted from Fig. 3 will contain the vertices $\{v_2, v_3, v_7, v_8\}$, and the CCW hull will contain the vertices $\{v_2, v_4, v_6, v_8\}$.

Lemma 1. *Given a simple open contour containing n vertices, the proposed algorithm computes both CW and CCW hulls in $O(n)$ time complexity.*

Proof. The original algorithm of Melkman [11] computes a CCW ordering of convex hull vertices in $O(n)$. Our algorithm for extracting the list of convex hull vertices having an order equal to the original contour adds two unit time operations to each operation on D (Algorithm 2). The overall complexity of this algorithm remains $O(n)$. Extracting the CW and CCW hulls both require a single pass over the list of convex hull vertices. The number of vertices on the hull will be less than or equal to n and the time complexity of this step is $O(n)$. Consequently the overall time complexity is $O(n)$. □

Opposed to altering the original algorithm of Melkman to obtain vertices listed in the desired order, an alternative approach would be to run the algorithm in its original form and post-process the result. The authors initially considered this approach but were unable to develop a robust implementation with time complexity not greater than $O(n)$.

3 Convexity Measure

In order to determine shape similarity between a contour and its corresponding hulls, the turning-function representation of Arkin et al. [1] was used. This function is denoted $\Theta(s)$ and measures the angle of the counter-clockwise tangent as a function of arclength s. $\Theta(s)$ accumulates the turning which takes place; increasing with left-hand turns and decreasing with right hand-turns. The contour is rescaled such that the total perimeter is 1; $\Theta(s)$ is therefore a function from $[0,1]$ in $\mathbb{R}$. The similarity of two contours A and B is determined by Equation 1 [8]. The variable θ represents a rotation of B and the value which minimizes this integral can be calculated by Lemma 3 in [1].

$$D(A,B) = \min_{\theta \in R} \int_0^1 \left(\Theta_A(s) - \Theta_B(s) + \theta\right)^2 ds \tag{1}$$

Let DU and DL be the distance between a contour W and its CW and CCW hulls by Equation 1 respectively. By assigning DH to be the minimum of these values the convexity of W is determined by the function SC in Equation 2. If a open contour is convex it will have an identical shape to its CW or CCW hull and consequently DH will equal 0. In this case SC will evaluate to 1 indicating a convex contour. The less convex a open contour the more dissimilar its shape will be relative to its CW and CCW hulls. At its limit the value SC will approach 0 as DH approaches infinity. Due to these facts SC satisfies the first three properties necessary for a measure of convexity presented in Section 1. The shape similarity measure D is similarity transformation invariant and as a result property four is also satisfied. SC can be applied to a closed contour by representing the contour as an open contour where the first and last vertices are equal.

$$SC(W) = \frac{1}{1+DH} \tag{2}$$

Theorem 1. *The convexity measure SC for a simple contour can be computed in $O(n)$ time complexity.*

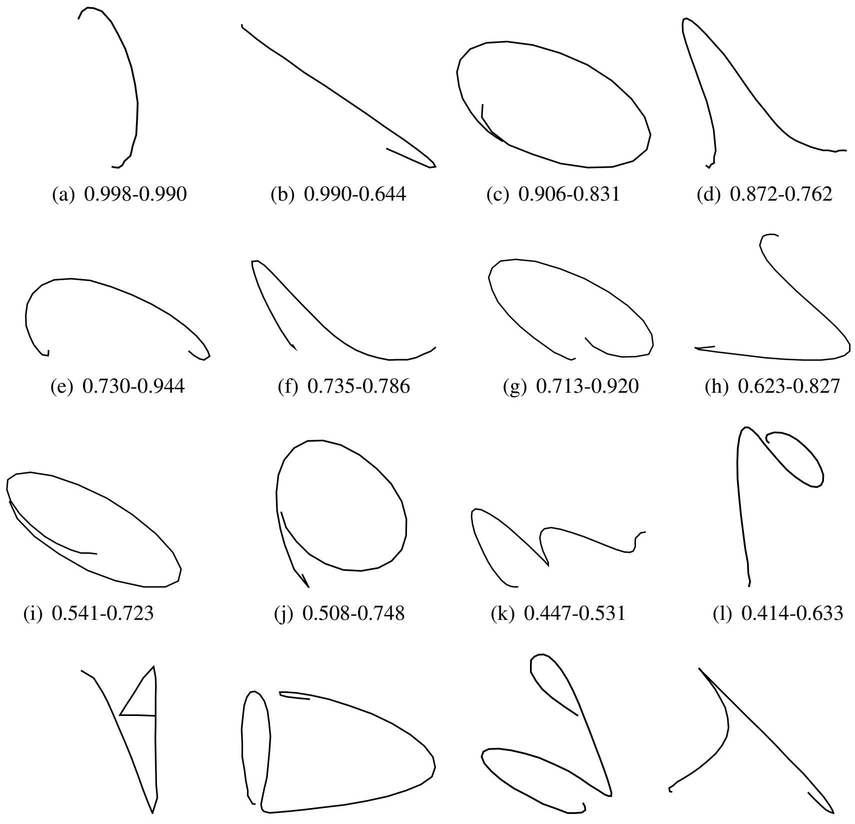
Proof. From lemma 1 the CW and CCW hulls of a contour can be calculated in $O(n)$. If the contour contains n vertices and its CW hull contains m vertices the integral D in Equation 1 for a given θ can be calculated in $O(n+m)$ [1]. Since $n \geq m$ is always true this reduces to $O(n)$. The value θ which minimizes D can be calculated in constant time by lemma 3 of [1]. The overall time complexity of SC is therefore $O(n)$. □

4 Results

To evaluate the performance of the proposed convexity measure for open contours the convexity measure M of Zunic and Rosin [23] was used as a benchmark. If A and B are two randomly chosen points on the open contour W the convexity $M(W)$ is defined as the probability that the open line segment AB does not intersect W or completely belongs to W. The measure M returns a convexity measure in the range $(0,1]$ with values closer to 1 indicating greater convexity. To the best of our knowledge there exists no other convexity measure which can be directly applied to open contours. To evaluate the performance of the proposed convexity measure for closed contours the convexity measure CP of Zunic and Rosin [22] was used as a benchmark. CP equates the convexity of a polygon as the ratio of the polygons l_2 perimeter to its l_1 perimeter minimized over all rotations of the polygon. It returns a convexity measure in the range $(0,1]$ with values closer to 1 indicating greater convexity.

Fig. 4 displays a set of open contours taken from the UJI Pen Characters Data Set [2]. Under each sub-figure the corresponding convexity measures are listed in the order SC followed by M. The sub-figures are ordered by SC value in descending order. The contour in Fig. 4(a) is very close to if not convex; consequently each convexity measure returns a corresponding value close to 1. The contour in Fig. 4(b) also exhibits high convexity and is assigned a high convexity value by SC. However examining the corresponding M value suggests a contour of low convexity. This value is less than the M values corresponding to many other significantly less convex contours; for example Fig. 4(h). The reason for this apparent inconsistency is due to the fact that the metric M is not robust in the presence of almost collinear line segments. Consider the series of line segments in Fig. 5. These segments represent a series of almost collinear segments but the turn angles have been exaggerated for illustration purposes. If a contour consists of a series of collinear or almost collinear segments it should be determined as have high convexity. Despite this fact, the segment joining any two random points on such a contour, for example A and B in Fig. 5, will have a high probability of intersecting the contour. This in turn results in such a contour receiving a low M convexity value. Another undesirable property of the M measure is that it suffers from high computational complexity due to the fact that it estimates convexity using Monte Carlo methods. Our own C++ implementation of M running on a Intel 2.8 GHz dual core processor requires between 5 and 10 seconds to compute convexity for a single contour in Fig. 4(a) to Fig. 4(p) using 100,000 samples. This number of samples was taken directly from the original implementation [23]. Fig. 6 displays a set of closed contours taken from the MPEG-7 data set [9]. Under each sub-figure the corresponding convexity measures are listed in the order SC followed by CP.

To quantify the performance of the proposed measure SC, as a shape descriptor relative to the benchmark methods, Precision-Recall curves for a k-nearest neighbour retrieval algorithm were calculated. Thirty object classes were selected from the MPEG-7 dataset and and for each class 10 object instances were selected to gave a dataset of 300 closed contours. For each contour the most salient part was selected by visual inspection to give a dataset of 300 open contours. The complete MPEG-7 dataset was not used because of the significant manual work required to select contour parts. For each closed contour its k nearest neighbours for $k = 1,\ldots,9$, in terms of both the SC and CP convexity measures, were calculated. For example the 7 nearest neighbours, in terms of SC, to the contour in Fig. 8(a) are displayed in Fig. 8(b)-8(h). Fig. 8(a), 8(b) and 8(h) all belong to the class chicken. For each value of k corresponding precision and recall values were calculated. The resulting Precision-Recall curves for the measures SC and CP are displayed in Fig. 9(a). These curves demonstrate that both measures offer similar performance. The same methodology was applied to the set of open contours. For example the 7 nearest neighbours, in terms of SC, to the contour in Fig. 7(a) are displayed in Fig. 7(b)-7(h). Fig. 7(a), 7(c) and 7(e) all belong to the class bat wing. The resulting Precision-Recall curves for the measures SC and M are displayed in Fig. 9(b). It is evident from these curves that SC offers greater performance compared to M.



(a) 0.998-0.990 (b) 0.990-0.644 (c) 0.906-0.831 (d) 0.872-0.762

(e) 0.730-0.944 (f) 0.735-0.786 (g) 0.713-0.920 (h) 0.623-0.827

(i) 0.541-0.723 (j) 0.508-0.748 (k) 0.447-0.531 (l) 0.414-0.633

(m) 0.307-0.613 (n) 0.304-0.683 (o) 0.242-0.597 (p) 0.212-0.742

Figure 4: The values are listed as SC-M under each contour.

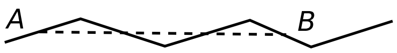
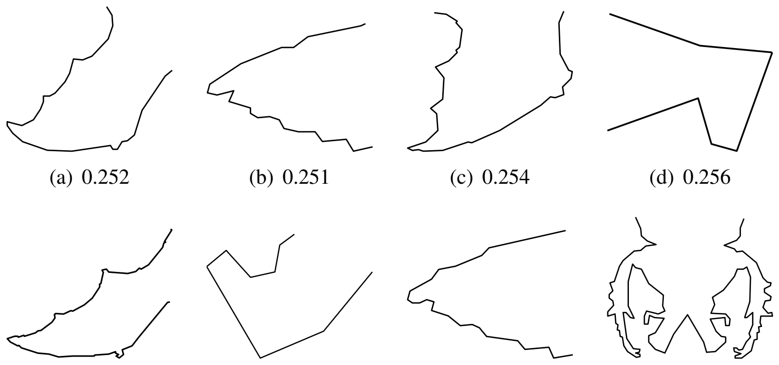


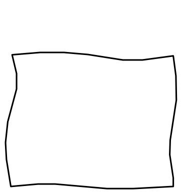
Figure 5: A series of almost collinear segments.

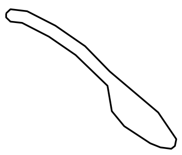


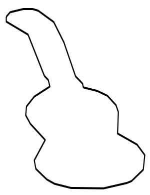
(a) 0.252 (b) 0.251 (c) 0.254 (d) 0.256

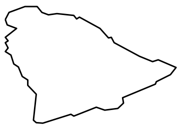
(e) 0.256 (f) 0.246 (g) 0.258 (h) 0.260

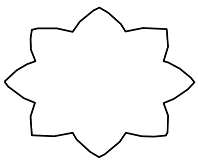
Figure 7: The corresponding *SC* value is displayed under each contour.

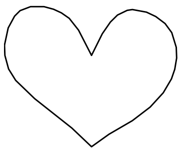

(a) 0.98-0.94

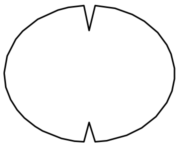

(b) 0.94-0.92

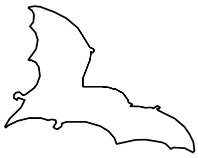

(c) 0.91-0.90

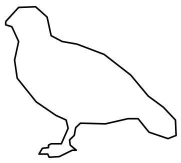

(d) 0.84-0.89

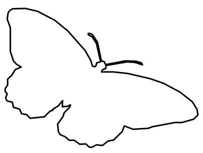

(e) 0.79-0.82

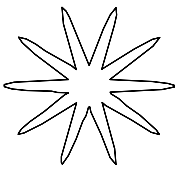

(f) 0.77-0.86

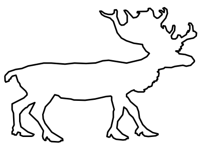

(g) 0.70-0.84

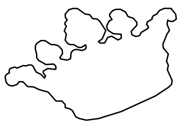

(h) 0.64-0.75

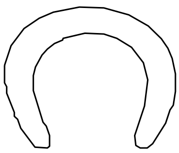

(i) 0.57-0.79

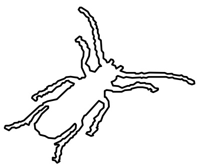

(j) 0.46-0.64


(k) 0.42-0.39


(l) 0.34-0.42


(m) 0.33-0.56


(n) 0.32-0.67


(o) 0.28-0.39

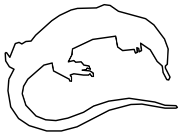
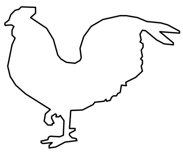
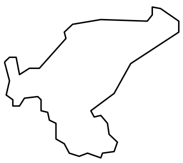
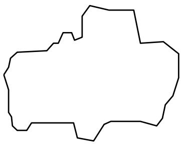
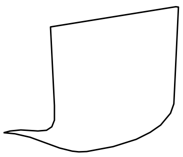

(p) 0.24-0.47

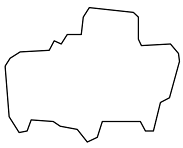
Figure 6: The values are listed as SC-CP under each contour.

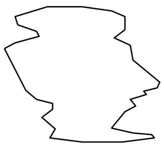

(a) 0.811

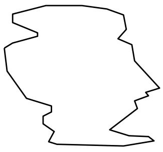

(b) 0.815


(c) 0.806


(d) 0.806


(e) 0.817


(f) 0.804


(g) 0.819

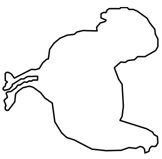
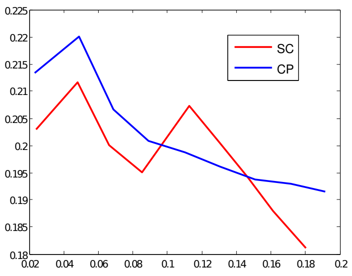
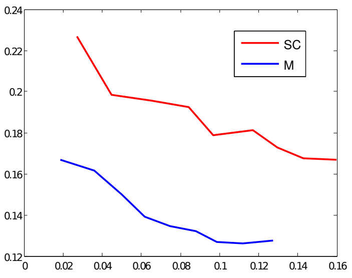

(h) 0.822

Figure 8: The corresponding *SC* value is displayed under each contour.



(a)



(b)

Figure 9: PR curves.

5 Conclusions

This paper proposes a new measure entitled *SC* for determining the convexity of both open and closed contours. The authors believe *SC* to be the first convexity measure to use shape similarity to determine convexity. Quantitative results demonstrate that *SC* provides similar performance to the current state of the art measure for closed contours while it offers improved performance relative to the state of the art measure for open contours. The $O(n)$ time complexity of *SC* makes it suitable for machine vision applications.

Acknowledgements

Research presented in this paper was part-funded by a Strategic Research Cluster grant (07/SRC/I1168) from Science Foundation Ireland under the National Development Plan.

References

[1] E. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.

[2] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] I. Biederman and G. Ju. Surface vs. edge-based determinants of visual recognition. *Cognitive Psychology*, 20(1):38–64, 1988.

[4] M.R. Daliri and V. Torre. Classification of silhouettes using contour fragments. *Computer Vision and Image Understanding*, 113(9):1017–1025, 2009.

[5] R. L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133, 1972.

[6] D.W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):23 –37, jan 1996.

[7] L. J. Latecki and R. Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73(3):441–454, 1999.

[8] L.J. Latecki. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, 2000.

[9] L.J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 424–429, 2000.

[10] Z. Liu, D. W. Jacobs, and R. Basri. The role of convexity in perceptual completion: beyond good continuation. *Vision Research*, 39(25):4244 – 4257, 1999.

[11] A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, 1987.

[12] D. Proffitt. The measurement of circularity and ellipticity on a digital grid. *Pattern Recognition*, 15(5):383 – 387, 1982.

[13] P.L. Rosin. Measuring shape: ellipticity, rectangularity, and triangularity. *Machine Vision and Applications*, 14(3):172–184, 2003.

[14] P.L. Rosin. Classification of pathological shapes using convexity measures. *Pattern Recognition Letters*, 30(5):570–578, 2009.

[15] P.L. Rosin and C.L. Mumford. A symmetric convexity measure. *Computer Vision and Image Understanding*, 103(2):101–111, 2006.

[16] P.L. Rosin and J. Zunic. Measuring Squareness and Orientation of Shapes. *Journal of Mathematical Imaging and Vision*, 39(1):13–27, 2011.

[17] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270 –1281, 2008.

[18] D. Sunday. Convex Hull of a 2D Simple Polyline, 2010. URL `http://softsurfer.com/Archive/algorithm_0203/algorithm_0203.htm`.

[19] Q. Tang, N. Sang, and T. Zhang. Extraction of salient contours from cluttered scenes. *Pattern Recognition*, 40(11):3100–3109, 2007.

[20] Z. Ying and D. Castanon. Partially Occluded Object Recognition Using Statistical Models. *International Journal of Computer Vision*, 49(1):57–78, 2002.

[21] J. Zunic and P.L. Rosin. Rectilinearity measurements for polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1193–1200, 2003.

[22] J. Zunic and P.L. Rosin. A new convexity measure for polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):923–934, 2004.

[23] J. Zunic and P.L. Rosin. Convexity measure for shapes with partially extracted boundaries. *Electronics Letters*, 43(7):380–382, 2007.