International Journal of Shape Modeling
Vol. 15, Nos. 1 & 2 (2009) 93–109
© World Scientific Publishing Company
DOI: 10.1142/S0218654309001227

World Scientific
www.worldscientific.com

SELF-INTERSECTING POLYGONS RESULTING FROM CONTOUR EVOLUTION FOR SHAPE SIMILARITY

PADRAIG CORCORAN

Department of Computer Science
National University of Ireland Maynooth
Maynooth, Co. Kildare Ireland
padraigc@cs.nuim.ie
http://www.cs.nuim.ie/~padraigc

ADAM WINSTANLEY

Department of Computer Science
National University of Ireland Maynooth
Maynooth, Co. Kildare Ireland

PETER MOONEY

Department of Computer Science
National University of Ireland Maynooth
Maynooth, Co. Kildare Ireland

JAMES TILTON

Computational & Information Sciences and Technology Office (CISTO)
NASA Goddard Space Flight Center (GSFC)
Greenbelt, MD, USA

In this paper we prove a well known contour evolution technique can result in inconsistent non-simple or self-intersecting polygons. This technique is used as a pre-processing step to a number of shape matching and part-decomposition strategies which are only well-defined for simple polygons. We analyze one such class of shape matching strategies, which use a highly cited method based on turning-functions to determine similarity. We prove that due to the possibility of self-intersecting polygons these methods are not well-defined. A simple alteration to the original contour evolution technique, which ensures the evolution of a consistent simple polygon, is proposed. This technique only alters the result slightly relative to the original evolution technique and therefore maintains the property of suitable shape evolution.

Keywords: Shape Similarity; Turning-Function; Contour Evolution.

1991 Mathematics Subject Classification: 68U05

1. Introduction

Object recognition represents an extremely powerful capability of the Human Visual System (HVS). In fact, many believe it is this ability which defines human cognition.[1] It has been shown that object shape is the single most important feature used by the HVS to recognize objects.[2] The shape of an object is commonly represented using its corresponding boundary contour. Often this contour needs to be simplified or its resolution reduced in order to remove distortions while preserving the perceptual appearance at a scale suitable for recognition and part decomposition.[3] Many algorithms which attempt to achieve this goal have been proposed in the domains of Geographical Information Science (GIS)[4] and object recognition.[5]

When performing simplification it is important that the resulting contour is consistent with the original where the definition of consistent is context or problem dependent. This topic of consistent contour simplification has been the focus of some research in the domain of GIS where vector map data is simplified.[6,7] Berg et al.[7] proposes a map simplification technique which is consistent in the sense that it will not introduce self-intersecting polygons and all point features will remain within their corresponding original polygon. Silva et al.[8] describe a simplification algorithm that possesses these features but also maintains the consistency of line features. In another work these authors describe a method capable of maintaining coincidence and incidence topology consistency.[9] The authors are unaware of any similar work in the domain of object recognition. As will be proved in this paper, maintaining the consistency of contours for the purpose of object recognition is extremely important.

Latecki et al.[5] proposed a contour evolution technique for contour simplification and part decomposition in order to facilitate object recognition. This technique is highly cited in the literature and has many applications. For example Latecki et al.[3,10] describe a partial shape similarity algorithm which measures the similarity between such object parts . The core of this algorithm is the turning-function shape similarity metric of Arkin et al.[11] which is used to measure the similarity between parts.

In this paper we prove the above contour evolution technique of Latecki et al. can result in self-intersecting polygons. We then prove that the above turning-function shape similarity metric is not well-defined for self-intersecting polygons. This implies the above partial shape similarity algorithm is also not well-defined and the contour evolution method is inconsistent given its context. We proposed an alteration to the contour evolution method which always produces simple consistent polygons and in turn makes the partial shape similarity algorithm well-defined.

The layout of this paper is as follows. In the following section we review the contour evolution technique of Latecki et al. and prove by construction it can result in a self-intersecting polygon. In section 3 we directly prove the turning-function shape similarity metric of Arkin et al. to be only well-defined for simple polygons and as a consequence the partial shape similarity algorithm of Latecki et al. is not well-defined. Section 5 describes the dataset used in this study and some of the challenges it presented. This is followed by a presentation of results. In the final section we draw conclusions from this work.

2. Contour Evolution

The basic concept of the Latecki et al.[5] contour evolution technique is to replace two consecutive line segments with a single line segment formed by joining their endpoints in each evolution step to obtain a shape hierarchy. In each step convex parts of the contour are identified as significant object parts. In order to produce an intuitive shape evolution a suitable order of substitution must be used. Latecki et al. proposed to perform substitution in an order where line segments that contribute the least to the overall shape are substituted first and the process converges when a convex polygon is formed. Referring to Figure 1, $s_1 = AB$ and $s_2 = EF$ are two consecutive line segments where $B = E$ is their common endpoint. $\beta = \beta(s_1, s_2)$ represents the turn-angle, that is $\beta = angle(EF) - angle(AB)$ where $angle(s)$ represents the angular direction of the line s.

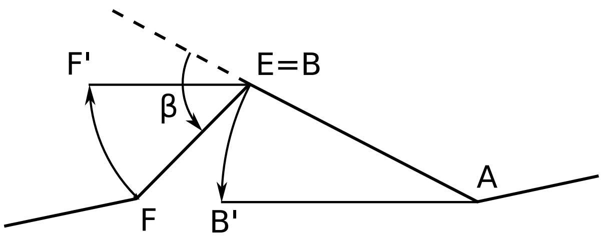


Fig. 1. Two consecutive line segments $s_1 = AB$ and $s_2 = EF$ are displayed.

Without loss of generality, it is assumed that $\beta > 0$. The circular arc-lengths by which endpoints B and F must be rotated around points A and E respectively such that they have the same direction is determined. These rotated points are referred to as B' and F' respectively. The circular arc-lengths, such that they are of equal length, can be determined by the function K in Eq. (1).

$$K(s_1, s_2) = \frac{\beta(s_1, s_2)\, l(s_1)\, l(s_2)}{l(s_1) + l(s_2)} \tag{1}$$

Where l is the length function normalized with respect to the total polygon perimeter. Latecki et al.[5] proved that the value $K(s_1, s_2)$ can be interpreted as the cost for linearization of arc $s_1 \cup s_2$. Using an example we will now prove by construction that in certain cases the above contour evolution technique results in a self-intersecting polygon. Referring to Figure 2, let A, B, C and D be four points on a contour such that the point D lies inside a triangle formed by A, B and C. Let $K(AB, BC)$ and $K(BC, CD)$ be the two smallest K values associated with this polygon. This could occur in cases where all other arc-lengths are significantly greater than those displayed in Figure 2. The turn-angles of ABC and BCD are represented by α and δ respectively with $\delta > \alpha$. Also relating to arc-lengths, $l(AB) < l(CD)$.

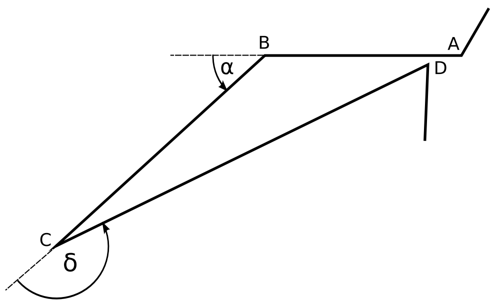


Fig. 2. Four points forming part of a larger polygon are displayed.

Using these properties of arc-lengths and turn-angles we now prove by construction using two lemma that prove $K(AB, BC) < K(BC, CD)$.

Lemma 2.1.

$$\frac{l(AB)\,l(BC)}{l(AB)+l(BC)} < \frac{l(BC)\,l(CD)}{l(BC)+l(CD)}$$

Proof.

$$\begin{aligned}
&\Rightarrow \frac{l(AB)}{l(AB)+l(BC)} < \frac{l(CD)}{l(BC)+l(CD)}\\
&\Rightarrow \frac{l(AB)+l(BC)}{l(AB)} < \frac{l(BC)+l(CD)}{l(CD)}\\
&\Rightarrow \frac{l(AB)}{l(AB)} + \frac{l(BC)}{l(AB)} < \frac{l(BC)}{l(CD)} + \frac{l(CD)}{l(CD)}\\
&\Rightarrow 1 + \frac{l(BC)}{l(AB)} < 1 + \frac{l(BC)}{l(CD)}\\
&\Rightarrow \frac{l(BC)}{l(AB)} < \frac{l(BC)}{l(CD)}\\
&\Rightarrow \frac{1}{l(AB)} < \frac{1}{l(CD)}\\
&\Rightarrow l(AB) < l(CD)
\end{aligned}$$

Which is true by definition of the polygon in question. □

Lemma 2.2. $K(AB, BC) < K(BC, CD)$

Proof. By substitution into Eq. (1)

$$\Rightarrow \frac{\beta(AB, BC)\, l(AB)\, l(BC)}{l(AB) + l(BC)} < \frac{\beta(BC, CD)\, l(BC)\, l(CD)}{l(BC) + l(CD)}$$

$$\Rightarrow \frac{\alpha l(AB)\, l(BC)}{l(AB) + l(BC)} < \frac{\delta l(BC)\, l(CD)}{l(BC) + l(CD)}$$

$$\text{let } x = \frac{l(AB)\, l(BC)}{l(AB) + l(BC)} \text{ and } y = \frac{l(BC)\, l(CD)}{l(BC) + l(CD)}$$

$$\Rightarrow \alpha x < \delta y$$

$x < y$ by Lemma 2.1 and $\alpha < \delta$ by definition of the polygon in question; therefore $\alpha x < \delta y$ proving Lemma 2.2. □

Since $K(AB, BC) < K(BC, CD)$ the point B will be deleted next and this will result in the self-intersecting polygon displayed in Figure 3. This proves that in certain cases the contour evolution technique of Latecki et al. will result in a self-intersecting polygon.

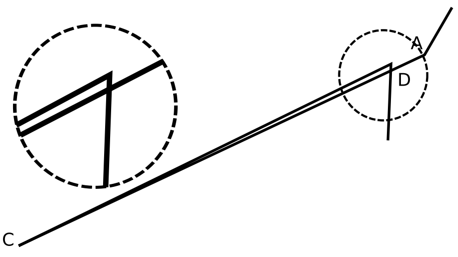


Fig. 3. The result of deleting point B from the polygon in Figure 2 is displayed. A self-intersecting polygon is formed with the intersection in question displayed at a larger scale on the left.

3. Turning-Function Similarity

Using the parts identified through evolution and part decomposition described in section 2, Latecki et al.[5] proposed to determine object similarity based on the shape similarity between these parts alone while ignoring the remainder of the contour.[3,10] The shape similarity between pairs of parts was determined using the turning-function shape similarity metric of Arkin et al.[11] In[12] the authors proposed to perform contour evolution without part decomposition and to measure similarity between complete contours using the metric of Arkin et al. In this section we prove in the case of self-intersecting polygons this metric is not well defined.

The boundary of a polygon A can be represented by a turning-function $\Theta_A(s)$. This function measures the angle of the counter-clockwise tangent as a function of arclength s, measured from a reference point O on the boundary of A. $\Theta_A(0)$ represents the angle v which the tangent at point O makes with some reference orientation such as the x-axis. $\Theta_A(s)$ accumulates the turning which takes place; increasing with left-hand turns and decreasing with right hand-turns. The polygon is rescaled such that the total perimeter is 1; $\Theta_A(s)$ is therefore a function from $[0,1]$ in R. A simple polygon and corresponding turning-function are displayed in Figure 4. For a simple polygon $\Theta_A(s+1) = \Theta_A(s) + 2\pi$. For a self-intersecting polygon $\Theta_A(s+1) = \Theta_A(s) + n2\pi$ where n is an integer representing the total number of anticlockwise turns minus the total number of clockwise turns. Arkin et al.[11] state that "*the function $\Theta_A(s)$ is well-defined even for arbitrary (not necessarily simple or closed polygonal) paths A in the plane*". Although this is case we now prove that the actual method they propose for determining the similarity between two turning-functions is not well-defined for self-intersecting polygons.

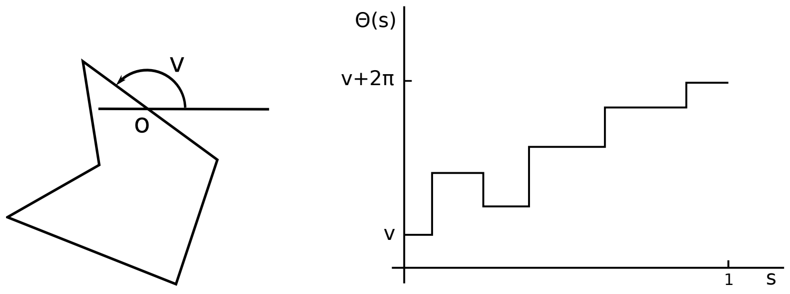


Fig. 4. The turn-function function $\Theta_A(s)$ for the simple polygon in (a) is shown in (b).

Consider two polygons A and B and their corresponding turning-functions $\Theta_A(s)$ and $\Theta_B(s)$. A measure of similarity between A and B can be determined by the distance between $\Theta_A(s)$ and $\Theta_B(s)$ according to a given metric function space.[11] Eq. (2) defines the L_p distance between A and B where $\|.\|_p$ denotes the L_p norm.

$$\begin{aligned}\delta_p(A,B) &= \|\Theta_A - \Theta_B\|_p \\ &= \left(\int_0^1 |\Theta_A(s) - \Theta_B(s)|^p ds\right)^{\frac{1}{p}} \qquad (2)\end{aligned}$$

δ_p is sensitive to both rotations of A (or B) and choice of reference point on the boundary of A (or B). Shifting the reference point O along A's boundary by distance t gives $\Theta_A(s+t)$. Rotating A by angle θ gives $\Theta_A(s) + \theta$. It is necessary to minimize the distance over all such values to obtain the best alignment; that is solve Eq. (3).

$$\begin{aligned}
&d_p\left(A,B\right)\\
&=\left(\min_{\theta\in R,t\in[0,1]}\int_0^1\left|\Theta_A\left(s+t\right)-\Theta_B\left(s\right)+\theta\right|^p ds\right)^{\frac{1}{p}}\\
&=\left(\min_{\theta\in R,t\in[0,1]}D_p^{A,B}\left(t,\theta\right)\right)^{\frac{1}{p}}\\
&\textit{where}\\
&D_p^{A,B}\left(t,\theta\right)=\int_0^1\left|\Theta_A\left(s+t\right)-\Theta_B\left(s\right)+\theta\right|^p ds \qquad (3)
\end{aligned}$$

Arkin et al. proved using three lemma's that $d_2\left(A,B\right)$ can be computed by initially finding the optimal θ for any fixed value t. Unfortunately this proof was based on the assumption that the polygon is question is simple and is therefore not well-defined for self-intersecting polygons. By analysing the original three lemma and altering the third we derive an equation to determine to optimal θ for any fixed value t which is well-defined for both simple and self-intersecting polygons.

Lemma 3.1. *$d_p\left(A,B\right)$ is a metric for all $p>0$.*

Proof. The proof of this lemma provided by Arkin et al. is independent of the total turn of the turning functions in question. It is valid in the case of self-intersecting polygons. □

Lemma 3.2. *For any fixed value of t, and for any $p\geq 1$, $D_2^{A,B}$ is a convex function of θ.*

Proof. The proof of this lemma provided by Arkin et al. is independent of the total turn of the turning functions in question and is therefore valid in the case of self-intersecting polygons. □

In order to simplify the following lemma we use the same notation as Arkin et al. That is $f\left(s\right)=\Theta_A\left(s\right)$, $g\left(s\right)=\Theta_B\left(s\right)$, $h\left(t,\theta\right)=D_2^{A,B}\left(t,\theta\right)$. The value n represents the number of anti-clockwise turns.

Lemma 3.3. *Letting $h\left(t,\theta\right)=\int_0^1\left(f\left(s+t\right)-g\left(s\right)+\theta\right)$. In order to minimize $h\left(t,\theta\right)$, the optimal θ is given by:*

$$\begin{aligned}
\theta^*\left(t\right)&=\int_0^1\left(g\left(s\right)-f\left(s+t\right)\right)ds\\
&=\alpha-2\pi nt \qquad (4)
\end{aligned}$$

where

$$\alpha=\int_0^1 g\left(s\right)ds-\int_0^1 f\left(s\right)ds$$

Proof.

$$\begin{aligned}\frac{\partial h\left(h,\theta\right)}{\partial \theta} &= \int_0^1 \left(2\theta + 2f\left(s+t\right) - 2g\left(s\right)\right) ds \\ &= 2\theta + 2\int_0^1 \left(f\left(s+t\right) - g\left(s\right)\right) ds\end{aligned}$$

Lemma 3.2 informs us that the minimum occurs when this equals zero and we solve for θ:

$$\theta^* = \int_0^1 \left(g\left(s\right) - f\left(s+t\right)\right) ds.$$

Next,

$$\begin{aligned}&\int_0^1 f\left(s+t\right) ds \\ &= \int_t^{t+1} f\left(s\right) ds \\ &= \int_t^1 f\left(s\right) ds + \int_1^{t+1} \left[f\left(s-1\right) + 2\pi n\right] ds \\ &= \int_t^1 f\left(s\right) ds + \int_0^t f\left(s\right) ds + 2\pi nt \\ &= 2\pi nt + \int_0^1 f\left(s\right) ds\end{aligned}$$

Thus,

$$\begin{aligned}\theta^* &= \int_0^1 g\left(s\right) ds - 2\pi nt - \int_0^1 f\left(s\right) ds \\ &= \alpha - 2\pi nt\end{aligned}$$

□

Substituting this expression for θ^* into $d_2\left(A,B\right)$ gives a one variable minimization problem. This method is well-defined for both simple and self-intersecting polygons. If $n = 1$, that is the polygon is simple, then θ^* equals $\alpha - 2\pi t$. This is the original equation derived by Arkin et al. and therefore this proves that it is only well-defined in the case of simple polygons.

Eq. (4) is well-defined in the sense that it satisfies the properties required to satisfy Eq. (2). To prove that it gives a measure of shape similarity similar to human perception requires further analysis. Arkin et al. proved this to be the case for simple polygons. In the case of self-intersecting polygons such analysis is not straight forward.[13] This point is illustrated by Figure 5 where a self-intersecting polygon may be interpreted as representing more than one object. A strategy to avoid this problem involves ensuring that a measure of similarity between self-intersecting polygons is never required. This is the focus of the following section.

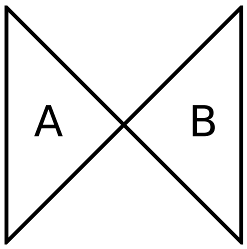


Fig. 5. This self-intersecting polygon can be interpreted as forming two regions A and B.

4. Contour Evolution Alteration

In this section we propose an alteration to the original contour evolution method of Latecki et al.[5] which will always result in a simple polygon. The metric of Arkin et al. is well-defined and has been shown to give an intuitive measure of similarity for such polygons. Therefore applying this metric to polygons returned from the proposed contour evolution method is well-defined.

The proposed contour evolution method differs in one major way to the original method of Latecki et al. Instead of removing the point with the lowest corresponding K value irrespective of result it removes the point with the lowest corresponding K value such that a simple polygon results. This strategy of taking an existing contour simplification method and altering its outcome to give a context dependent consistent result has used in a number of publications.[6,7,14]

Using the same notation as Latecki et al., let $D_m = s_0, \ldots, s_{m-1}$ be a decomposition of a contour C into consecutive line segments. The proposed algorithm, displayed in Algorithm 1, iteratively computes the evolution D_k until k is reduced to r or the polygon becomes convex.

Algorithm 1 Contour Evolution of D_m

```
1: $k = m$
2: **while** $(D_k$ not convex$)$ & $(k > r)$ **do**
3:     In $D_k$ determine minimum $K(s_i, s_{i+1})$ such the removal of the corresponding
       line segments results in a simple polygon.
4:     $D_{k-1} = D_k$ where the line segments $s_i, s_{i+1}$ are replaced by a line segment
       joining the endpoints of the arc $s_i \cup s_{i+1}$.
5:     $k = k - 1$
6: **end while**
```

Two key methods are required in the implementation of this algorithm. Firstly we must be able to determine if a given polygon is simple. Berg et al.[7] propose to determine if a simplified polygon is simple by examining if any points fall within the region defined by the difference between the two polygons in question. The complexity of determining if a point lies within a region is $O(n)$ where n represents the number of points used to represent the polygon.[15] Given that this must be performed for n points the total complexity of this algorithm is $O\left(n^2\right)$. A less computationally complex solution is to check line intersections and this can be computed in $O(nlogn)$.[15,16] This was the strategy adopted in this work. Secondly we must be able to determine if a given polygon is convex. This is determined by examining if all exterior angles have the same sign and has computational complexity of $O(n)$.[16] In this work we used the implementations of the above methods which are contained in the open-source CGAL C++ library.[15]

The proposed algorithm adds another layer of complexity on top to the original algorithm of Latecki et al. In the original algorithm, given the corresponding K values for each point, the point to remove next is determined by finding the minimum of these values. This can be determined in complexity $O(n)$. On the other hand, this same step in the proposed algorithm has a worst case complexity of $O(nlogn + n * nlogn) = O\left(n^2 logn\right)$ which occurs if the point with the largest K value is always the only one which can be removed without resulting in a self-intersecting polygon. The first $O(nlogn)$ term is required to sort the K values while the second $O(nlogn)$ term is the complexity required to determine if the corresponding polygon is simple. As will be shown in section 6, removal of the point with the corresponding lowest K value only results in a self-intersecting polygon a small percentage of times. This gives the above step in the proposed algorithm an average complexity of $O(n + nlogn) = O(nlogn)$ where $O(n)$ is required to determine the minimum K.

In order to reduce the number of evolution iterations it is common to initially reduce the points representing a contour to a smaller but representative number using a less computationally demanding method.[5] In this work we uniformly marked points along the contour and then remove all others in a sequential manner if their removal resulted in a simple polygon.

5. Shape Dataset

The MPEG-7 dataset of binary shape images was used in this work.[17] This dataset contains 70 varied object categories (e.g. apple, bird) with 20 objects per category giving a total of 1400 images. An example image in the dataset is displayed in Figure 6 (a). To extract the contour coordinates from these images in a clockwise direction, a contour following algorithm which traces along the interior of the boundary was used[18] (p. 796). If the binary shape contains a thin section of width one pixel wide for length greater than a single pixel, the contour following algorithm will retrace its own steps. This results in two boundary line segments lying along its path. Consider the image in Figure 6 (b) which represents the upper right leaf of the apple in Figure 6 (a). Because the tip of the end of this leaf is only a single pixel wide and of length greater than one pixel, the contour following algorithm re-traces

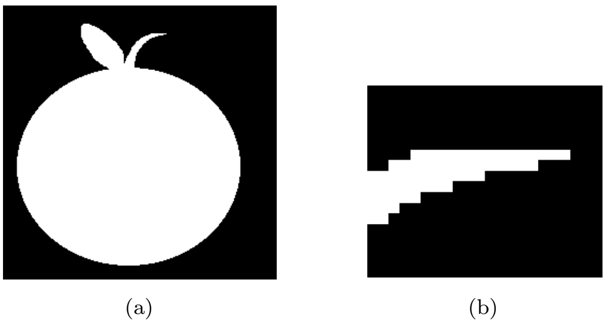


Fig. 6. An image (apple-7) taken from the MPEG-7 dataset is displayed in (a). An enhanced view of the tip of the leaf in (a) is displayed in (b).

its path as it follows the inside of the contour. The resulting contour is displayed in Figure 7 (a). Due to the fact that two boundary segments lie along the same path, the polygon in question is technically self-intersecting. Because two lines which overlap each other intersect at every point along the common line. The function *isSimple* from the CGAL library classifies such polygons as self-intersecting. To overcome this issue we post-processed the resulting contours using the following algorithm. For each contour point we look ahead a distance of 1 to $n - 1$ points where n represents the number of points representing the contour in question. If it is found that two points which have the same coordinates occur we remove the section of the contour between these points. Figure 7 (b) displays the result of applying this algorithm to Figure 7 (a). We can see that the contour section, corresponding to where two parts originally lay along the same path, has been removed.

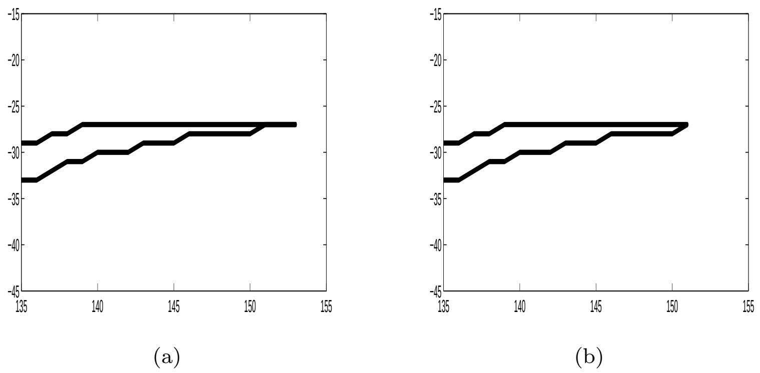


Fig. 7. The results of the contour following algorithm applied to Figure 6 (b) is displayed.

6. Results

The results section is organized as follows. Firstly we statistically analyse the tendency of the original contour evolution algorithm of Latecki et al. to produce self-intersecting polygons. Next we analyse how our proposed alteration to the contour evolution technique effects the resulting polygons. 600 images from the MPEG-7 dataset were randomly selected and the corresponding contours extracted. We then reduced the number of points representing each of these contours to 150 using the uniform selection approach described in section 4. Figure 8 (a) displays an original object contour while Figure 8 (b) shows the corresponding result following uniform selection. From this figure we can see that while the number of representation points has been reduced significantly the original shape is still accurately represented.

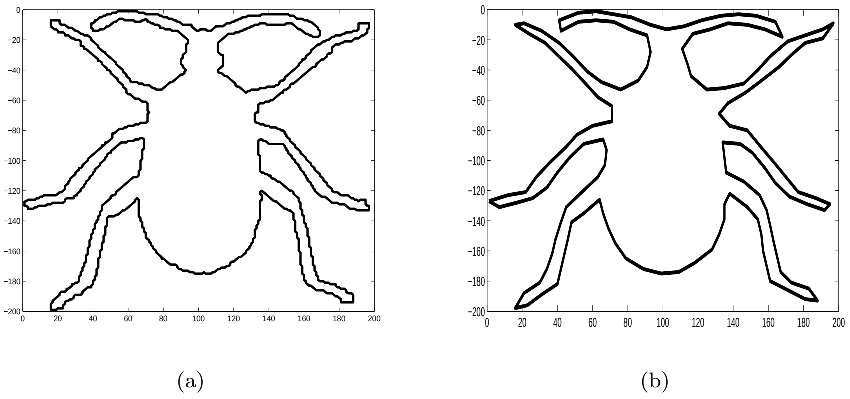


Fig. 8. The contour of a binary image (Beetle-13) extracted using a contour following algorithm is displayed (a). This is represented by 1540 points. The result of reducing this number of points to 150 is displayed in (b).

Next we evolved each of these polygons to 36 representation points using the original evolution strategy of Latecki et al. and our proposed evolution strategy. This number of points was chosen because it is the same number used in[12] and we found it provided suitable representation of object shape and parts. For 21 of the 600 polygons in the dataset, our proposed evolution procedure at least once choose not to remove a point with the corresponding lowest K value because doing so would result in a self-intersecting polygon. This represents 3.5% of the polygons in the dataset. This is a small but significant percentage which shows the evolution strategy of Latecki et al. to be not well-defined when applied to real data. For each of these polygons we counted the number of evolution steps a point with the lowest K was not removed for this reason. These values are plotted in increasing order in Figure 9 and range from 1 to 44. Given that the evolution only consisted of 114 steps for each polygon, these counts represent a significant percentage of the total steps.

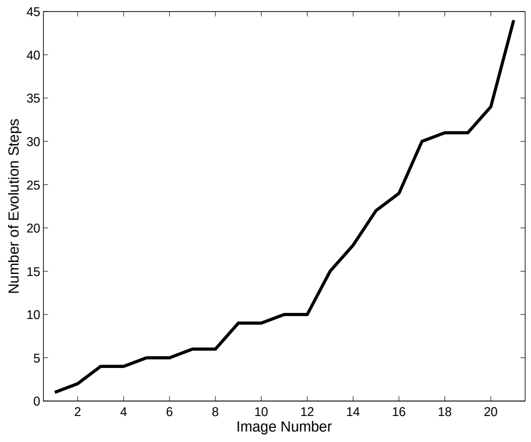
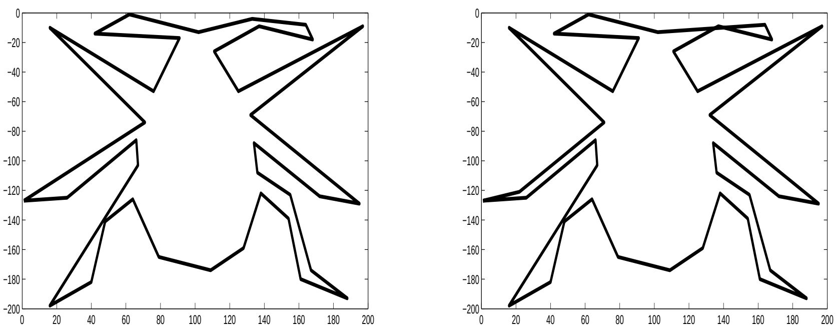


Fig. 9. This figure displays for each of the effected images a sorted count of the number of evolution steps a point with the lowest K value was not removed by the proposed algorithm.

We now provide a qualitative evaluation of our proposed evolution algorithm in comparison to the original algorithm of Latecki et al. We only compare results on images which the algorithm of Latecki et al. produced self-intersection polygons; for all other polygons both algorithms produced the same results. Figure 10 (a) displays the simple polygon returned by our proposed evolution algorithm when applied to the polygon in Figure 8 (b). On the other hand, Figure 10 (b) shows the self-intersecting polygon returned by the Latecki et al. evolution algorithm when applied to the same polygon. This polygon contains a single self-intersection which is shown in Figure 11. Apart from the self-intersection the shapes represented by the polygons in Figures 10 are similar. They do however contain some minor differences; for example the middle left leg is represented differently. The similarity in shape signifies that despite producing a different polygon, the propose algorithm still maintains the accurate shape representation properties of the original Latecki et al. algorithm.[5] This similarity between simple and self-intersection polygons is difficult to quantify due to the fact that most existing metrics assume that the polygons in question are both simple. For example, when determining similarity using the Haussdorff distance the interior of the polygons must be considered.[20] By the *Jordan curve theorem*, the interior of a self-intersecting polygon is undefined.

As a second example, both evolution algorithms were applied to the polygon in Figure 12. Figure 13 (a) displays the simple polygon returned by our proposed evolution algorithm. On the other hand, Figure 13 (b) shows the self-intersecting polygon returned by the Latecki et al. evolution algorithm. The self-intersection in question is shown in Figure 14. As in the previous example shapes returned by both



(a) (b)

Fig. 10. This figure shows the polygons returned by the proposed algorithm and the algorithm of Latecki et al. in (a) and (b) respectively when applied to the polygon in Figure 8 (b). Both polygons contain 36 points.

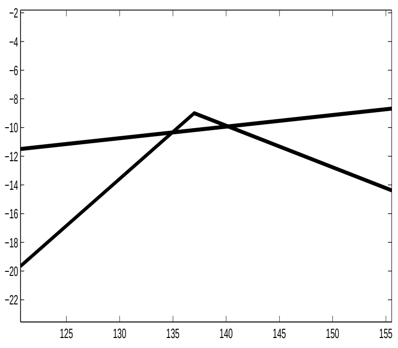


Fig. 11. An enhanced view of the upper right of the polygon in Figure 10 (b), showing a self-intersection, is displayed.

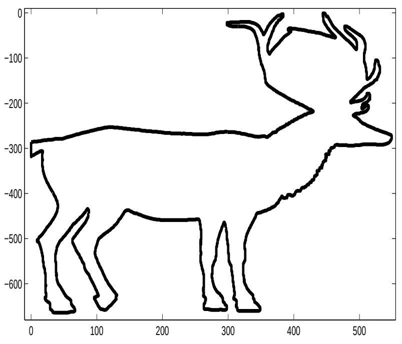


Fig. 12. This figure displays the contour of an object (deer-12) which is represented by 150 points.

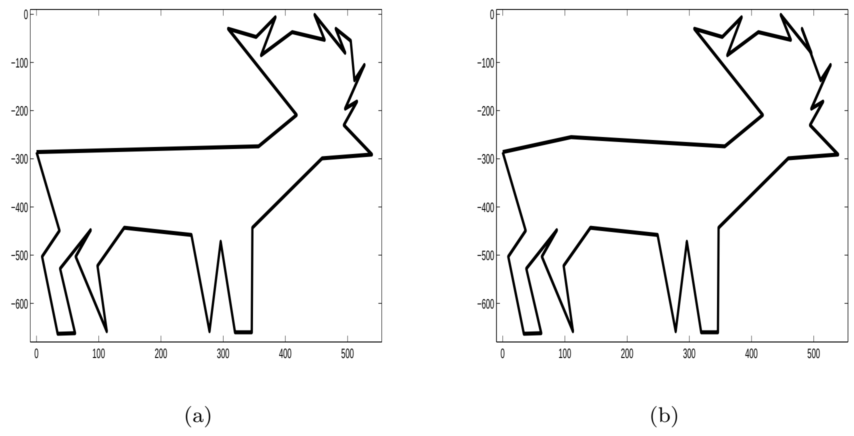


Fig. 13. This figure shows the polygons returned by the proposed algorithm and the algorithm of Latecki et al. in (a) and (b) respectively when applied to the polygon in Figure 12. Both polygons contain 36 points.

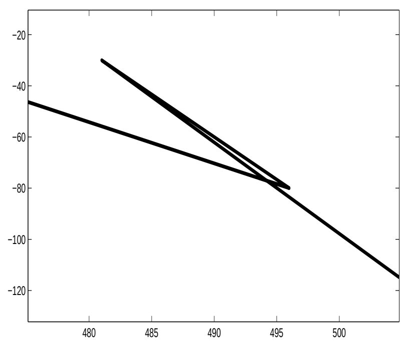


Fig. 14. An enhanced view of the upper right of the polygon in Figure 13 (b), showing a self-intersection, is displayed.

algorithms are similar but do contain a small number of minor differences. These are located along the animals back and the front of the antlers.

7. Conclusion

In this paper we have proven that the well known contour evolution technique of Latecki et al. can evolve a simple polygon into a self-intersecting polygon. Experimental results on a shape dataset demonstrate that this regularly occurs in practice. In this paper we focused on a class of existing shape matching algorithms which use this technique as a pre-processing step. These algorithms subsequently derive a measure of shape similarity via a turning function metric. We have proven, that in its current form, this metric is only well-defined for simple polygons. In-turn, this implies the above class of shape matching algorithms are not well-defined. A modification to the original evolution technique was proposed and implemented. This method always generates simple polygons and therefore is suitable for shape simplification.

Although we have focused exclusively on a single contour evolution technique the concept of consistent contour simplification is under-researched outside the domain of GIS. There exist many other polygon simplification algorithms which are commonly used as a pre-processing step to shape matching[19](page 341). This work could therefore be extended to analyze whether such algorithms always generate simple polygons and if not are they used within frameworks which assume they do.

As was discussed previously our proposed contour evolution technique adds an extra layer of computational complexity to the original evolution algorithm. Future work will investigate whether a means to reduce this overhead exists.

Acknowledgements

The authors would like to thank Joseph S. B. Mitchell[11] for kindly answering a number of questions relating to his work on the turning-function shape similarity metric. This work was partly funded by a Strategic Research Cluster grant (07/SRC/I1168) from Science Foundation Ireland under the National Development Plan.

References

1. S. Harnad, To Cognize is to Categorize: Cognition is Categorization, in *Handbook of Categorization*, eds. H. Cohen and C. Lefebvre (Elsevier, 2005) 20–45.
2. I. Biederman and G. Ju, Surface versus edge-based determinants of visual recognition, *Cognitive Psychology*, **20** (1988) 38–64.
3. L. Latecki and R. Lakämper, Shape similarity measure based on correspondence of visual parts, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22** (2000) 1185–1190.
4. D. Douglas and T. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, *The Canadian Cartographer*, **10** (1973) 112–122, 1973.
5. L. J. Latecki and R. Lakämper, Convexity rule for shape decomposition based on discrete contour evolution, *Computer Vision and Image Understanding*, **73** (1999) 441–454.
6. A. Saalfeld, Topologically consistent line simplification with the Douglas Peucker algorithm, *Cartography and Geographic Information Science*, **26** (1999) 7–18.
7. M. D. Berg, M. V. Kreveld, and S. Schirra, A new approach to subdivision simplification, *ACSM ASPRS Annual Convention*, **4** (1995) 79–88.
8. A.C. da Silva and S.-T. Wu, A robust strategy for handling linear features in topologically consistent polyline simplification, in *GeoInfo*, Sao Paulo, Brazil (Nov. 2006), pp. 19–34.

9. A. C. Silva and S.-T. Wu, Preserving coincidence and incidence topologies in saalfelds polyline simplification algorithm, in *GeoInfo*, Sao Paulo, Brazil (Nov. 2005), pp. 107–121.
10. L. Latecki, R. Lakämper, and D. Wolter, Optimal partial shape similarity, *Image and Vision Computing*, **23** (2005) 227–236.
11. E. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell, An efficiently computable metric for comparing polygonal shapes, *IEEE Trans. Pattern Anal. Mach. Intell.*, **13** (1991) 209–216.
12. S. Antani, D. Lee, L. Long, and G. Thoma, Evaluation of shape similarity measurement methods for spine x-ray images, *Journal of Visual Communication and Image Representation*, **15** (2004) 285–302.
13. M. Johnston, C. Scott, and R. Gibb, Problems Arising From A Simple GIS Generalisation Algorithm, in *Proc. Eleventh Annual Colloquium of the Spatial Information Research Centre*, eds. P. Whigham (Dunedin, New Zealand 1999), pp. 191–200.
14. A. C. Silva and S.-T. Wu, Consistent handling of linear features in polyline simplification, *Advances in Geoinformatics* (2007) 1–17.
15. G. Giezeman and W. Wesselink, 2D polygons, in *CGAL-3.2 User and Reference Manual*, eds. C. Board, (2006).
16. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing* (Cambridge University Press, New York, 2007).
17. L. Latecki, R. Lakämper, and T. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, South Carolina (June 2000), pp. 424–429.
18. R. Gonzalez and R. Woods, *Digital Image Processing (3rd Edition)*, (Prentice-Hall, Upper Saddle River, NJ, USA, 2007).
19. M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, (Thomson-Engineering, 2007).
20. P. Rotter, A.M.J. Skulimowski, C. Kotropoulos and I. Pitas, Fast shape matching using the Hausdorff distance, in *Mirage: Computer Vision / Computer Graphics Collaboration Techniques and Applications, INRIA Rocquencourt, France (March 2005).*