

A Symmetric Convexity Measure

Paul L. Rosin and Christine L. Mumford

Cardiff School of Computer Science,

Cardiff University,

Cardiff,

UK

{Paul.Rosin@cs.cf.ac.uk, C.L.Mumford@cs.cardiff.ac.uk}

Abstract

A new area-based convexity measure for polygons is described. It has the desirable properties that it is not sensitive to small boundary defects, and it is more symmetric with respect to intrusions and protrusions than other published convexity measures. The measure requires a maximally overlapping convex polygon, and this is efficiently estimated using a genetic algorithm (GA1). A second genetic algorithm (GA2) is then used to fine tune the result. In addition, the convex polygon is used to generate other values, measuring the amount of protrusions and intrusions that a polygon contains. Furthermore, the scheme can be modified to find the convex skull, which yields another new convexity measure. Examples of the measures' application to medical image analysis are shown.

1 Introduction

Convexity is a useful attribute of shape, and has applications in classification, image segmentation, figure/ground separation, etc. Whereas its mathematical definition provides a binary property, in image analysis we prefer a continuous measure, so that a shape can be assigned a degree of convexity. This provides more information, enabling classification for example to be more discriminating, and also allows the concept of convexity to be applied more usefully and robustly to real life irregular and noisy shape data.

Many convexity measures for polygons in the literature are either area [2, 19] or perimeter based [24]. The most common computes the ratio of the area of the polygon to the area of its convex hull [18], $C_A = \frac{area(P)}{area(CH(P))}$, where P denotes the input polygon, and $CH(P)$ its convex hull. This measure is nominally area-based, and so narrowing the intrusion (i.e. decreasing its area) in the polygon in the lower row in Figure 1 increases the measured convexity. In the limit, as the area of the intrusion tends to zero (an infinitely thin cut), the convexity tends to one (perfect convexity). However, there is an obvious asymmetry in the measure since a similar defect, but in the opposite sense, namely the protrusion in the upper row in Figure 1 produces a lower measured convexity. Even worse, in this example as the protrusion narrows the measured convexity decreases rather than increases, disagreeing with our perception. Thus, the measure inherits the convex hull's sensitivity to protrusions. Such sensitivity to thin protrusions is more appropriate to perimeter based convexity measures such as $C_L = \frac{perimeter(CH(P))}{perimeter(P)}$, or Žunic and Rosin's [24] method (C_f) which is also sensitive to thin intrusions. While in some applications it desirable to be sensitive to thin intrusions/protrusions, in others an area-based criterion is more appropriate.

2 The New Convexity Measure

In this paper we propose an alternative area-based convexity measure related to C_A which has the advantages that it is not sensitive to small (in area) boundary defects, and it is symmetric with respect to intrusions and protrusions. The basic idea is to replace the convex hull of the polygon P by its "robustified" version which we define as the convex polygon Q that best fits P in the sense of maximising the overlap of P and Q , or equivalently minimising the area of $P \text{ XOR } Q$.¹ For the two shapes in Figure 1, Q in both cases is the underlying rectangle. The right column in Figure 1 highlights $P \text{ XOR } Q$, which has the same area for the two shapes. To make the measure scale invariant it needs to be normalised with respect to size. This can be done using either the area of the input polygon or by the area of the fitted convex polygon. These alternative normalisations give the following

¹Thus the convexity measure is similar in spirit to many other shape measures which fit an ideal model of the shape (e.g. rectangle, circle, ellipse) to the data, and measure the deviations from the fitted model [16].

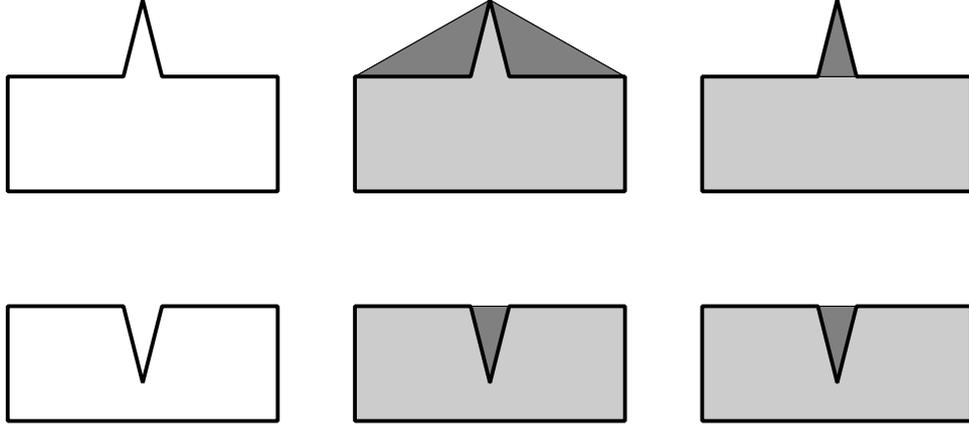


Figure 1: Measuring convexity of the two shapes in the left column; middle: the standard convex hull based approach C_A has a larger discrepancy in area (shown in dark gray) when the input polygon has a protrusion rather than an intrusion; right: the proposed approach fits a polygon (in this case a rectangle) producing similar discrepancies in area for both protrusion and intrusion.

measures: $E_P = \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(P)}$ and $E_Q = \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(Q)}$. The latter ensures greater symmetry (see Figure 1) but is potentially less robust than the former if Q is not estimated accurately.

To make the measure maximal for convex shapes E_P is modified to form the convexity measure

$$C_P = \frac{1}{1 + \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(P)}}, \quad (1)$$

lying in the range $[0, 1]$ which replaces our previously published measure [17] $C_P = 1 - \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(P)}$. For our implementation of the convexity measure only an approximation to Q is found in order to improve computational efficiency. Consequently there is no guarantee that $\text{area}(P \text{ XOR } Q) < \text{area}(P)$. Thus we found that a small proportion of shapes (less than 1% of the thousands we have tested) produced negative values for the latter C_P formulation due to a small value of $\text{area}(P)$ relative to the overlap. An example is given in Figure 2a in which our estimated value for Q is especially poor, incorrectly producing a large value of $\text{area}(P \text{ XOR } Q)$ relative to $\text{area}(P)$. For better solutions to Q (e.g. Figure 2b) this is avoided.



Figure 2: The fitted convex polygon Q in (a) has little overlap with the input polygon, which in combination with a low value of $\text{area}(P)$ leads to a negative value of the convexity measure $1 - \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(P)}$. This is avoided using the normalisation $C_P = \frac{1}{1 + \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(P)}}$. Neither normalisation yields negative values when a better solution for Q is found; for comparison the optimal Q is shown in (b).

Just as E_P can lie outside $[0, 1]$, so can E_Q , which is not bounded from above. This can be demonstrated by the star shaped polygon in Figure 3. For a star with very short arms Q is a regular polygon centred on the star. However, as the arms grow outwards the space between them grows much faster than the increased area of the arms. This leads to Q flipping to fit one of the arms, and so for an n -sided star (n is odd) $E_Q \approx n - 1$. We therefore normalise E_Q to form the measure

$$C_Q = \frac{1}{1 + \frac{\text{area}(P \text{ XOR } Q)}{\text{area}(Q)}} \quad (2)$$

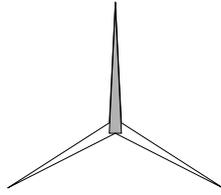


Figure 3: The convex polygon Q is fitted to one arm of the star.

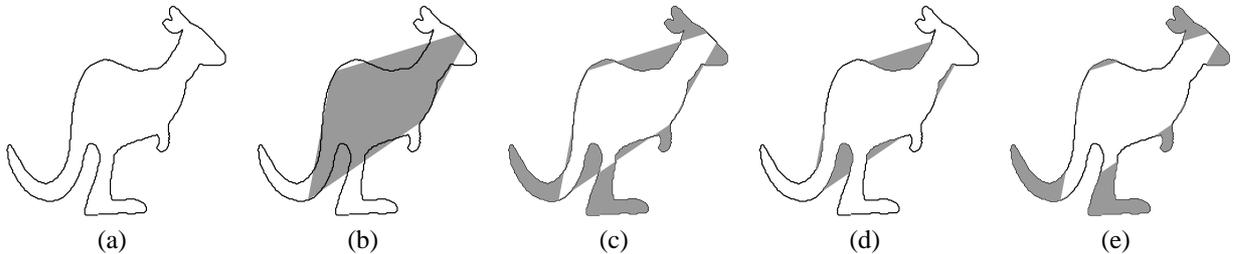


Figure 4: Set operations applied to polygons. (a) input polygon P , (b) Q overlaid on P , (c) $P \text{ XOR } Q$, (d) $Q \setminus P$, (e) $P \setminus Q$.

to keep it in the range $[0, 1]$.

Given the polygon Q it is now also possible to use it to define two new shape measures. The amount of protrusions and intrusions can be quantified by $\frac{\text{area}(P \setminus Q)}{\text{area}(Q)}$ and $\frac{\text{area}(Q \setminus P)}{\text{area}(Q)}$. Only the former needs to be normalised, and so the measures of protrusiveness and intrusiveness are

$$C_{\text{prot}} = \frac{\text{area}(P \setminus Q)}{\text{area}(P \setminus Q) + \text{area}(Q)} \quad (3)$$

$$C_{\text{int}} = \frac{\text{area}(Q \setminus P)}{\text{area}(Q)}. \quad (4)$$

Figure 4 demonstrates the various quantities obtained by performing the set operations between a polygon and its fitted polygon Q .

The protrusiveness and intrusiveness measures make intuitive sense, as demonstrated on the comb shapes in Figure 5. A small gap between the teeth (Figure 5a) results in the top edge of P being treated as a solid with indentations, i.e. zero protrusiveness and a non-zero intrusiveness. Making the gaps larger than the teeth (Figure 5b) reverses the interpretation, i.e. non-zero protrusiveness and zero intrusiveness.

The protrusiveness and intrusiveness measures are not exactly symmetric due to the normalisation required to keep their values in the range $[0, 1]$. However, the extra term $\text{area}(P \setminus Q)$ in (3) is small w.r.t. $\text{area}(Q)$ – otherwise the region $P \setminus Q$ would be treated as part of Q rather than external to Q . Therefore, the protrusiveness and intrusiveness values will be similar in symmetric situations. For instance, if the intrusions in Figure 5a are reflected out to form protrusions as in Figure 5c, such that the area of Q remains constant, then the intrusiveness (Figure 5a) and protrusiveness (Figure 5c) values only differ by less than 10%. These two shapes also demonstrate that the protrusion and intrusion measures provide significantly different information compared to the convexity measure. While Figures 5a and 5c have identical convexity values, they are very distinct according to both their protrusion and intrusion measure values.

3 The Genetic Algorithms

We have developed two genetic algorithms (GAs) to compute our new convexity measure. For a given input polygon, P , both of the GAs attempt to minimize $\text{area}(P \text{ XOR } Q)$ as their objective function, this being the key component in our new convexity measure. The first algorithm, GA1, is of primary importance and is applied before the second, GA2. GA2 can be regarded as a “fine tuner” that may be applied, if desired, to the output produced by GA1, and it will attempt to improve on it. GA1 is confined to a discrete search space consisting entirely of non-empty subsets of vertices selected from P , but GA2 is not restricted in this way. Indeed, starting from the vertex subset produced by GA1, GA2 will perturb the coordinates of these vertices, and cause them to “drift,” in an attempt to find a convex polygon that more closely matches the outline of P . Thus, the search space for GA2

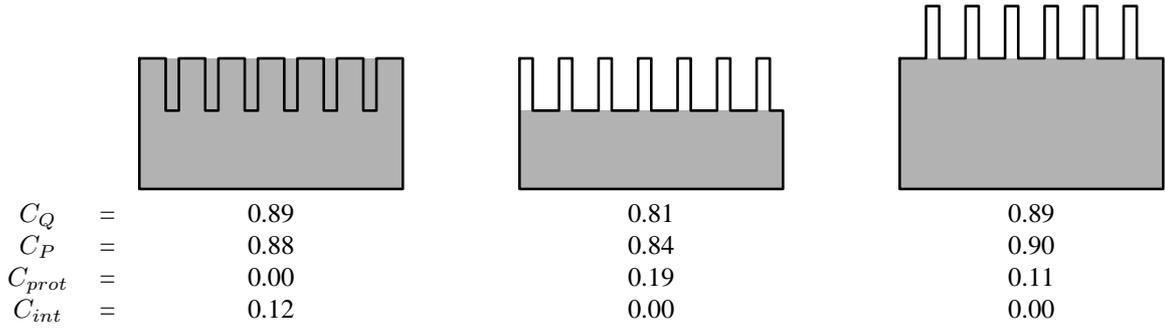


Figure 5: Protrusion and intrusion values for comb-like shapes.

is continuous, and although (slower) exact alternatives exist for GA1 (see Section 5.1), we cannot conceive of an exact solution method that would replace GA2.

The general framework for GA1 and GA2 is the same and is outlined in Figure 6. The two GAs differ in their representations, however: GA1 uses a bit-string chromosome to encode the presence/absence of vertices from P (see Section 3.2), while GA2 encodes the coordinates of the vertices from the estimate of Q output from GA1 (see Section 3.3). We will now describe the GA framework and the two representations in more detail.

Procedure GA

begin

Generate and repair a population of N individuals

Evaluate the objective function for each individual and store it

Store *best-so-far*

Repeat

For each member of the population

This individual becomes the first parent

Select a second parent at random

Apply crossover to produce offspring

Apply a single mutation to the offspring

Repair the offspring

Evaluate the objective function produced by offspring

If offspring duplicates existing values of the objective function within the population, delete it

else

If offspring better than weaker parent then it replaces it in the population

If offspring better than *best-so-far* then it replaces *best-so-far*

Endfor

Until stopping condition satisfied

Return *best-so-far*

End

Figure 6: Algorithm 1: A simple steady state genetic algorithm

3.1 The GA Framework

Figure 6 outlines the simple steady state [21] GA used in our study. Like all GAs, our GA operates on a population of candidate solutions and relies on a repeated cycle of selection, breeding and modification in order to improve the solution quality within the population. In what follows the term *parent* refers to any candidate solution that is selected for breeding and modification, and the term *offspring* describes a candidate solution resulting from such operations. Our GA, first presented in [14], is particularly easy to implement: it relies on simple comparisons between parents and their offspring to determine which individuals will live and which will die [3], and does not need to calculate the global selection probabilities typical of other GAs. These simple ideas have also been successfully applied to multi-objective optimization [13, 12, 22].

A population of N individuals is first generated, each representing a subset of vertices which provide an estimate of Q . The “repair” operation mentioned in the pseudocode is needed to correct estimates of Q that are invalid, because they are not convex. The outer loop of the GA corresponds to the concept of a “generation,” and the inner loop performs the reproduction, selecting parents in pairs, then applying *crossover* and *mutation*. Crossover is a sharing of genetic material from the two parents to produce an offspring, and mutation makes small

random changes in the resulting offspring chromosome. The pairs of parents are selected in the following way: the first parent is selected deterministically in sequence, but the second parent is selected uniformly, at random. Following the production and repair (see Section 3.2) of an offspring, the objective function is calculated for the offspring and a decision is then made as to whether it lives, or whether it dies. If the value of the objective function for the offspring is better than that of its weaker parent, it survives at the expense of that parent and replaces it in the population (i.e., the subset of vertices representing the offspring’s estimate of “Q” replaces the subset of vertices stored as the weaker parent). In our experiments we run the GA for a fixed number of generations, and this provides our stopping condition. The eventual output of the GA is the best value for $area(P \text{ XOR } Q)$, and the corresponding vertex set for our estimate of Q .

Worthy of note is the deletion of duplicates policy incorporated in the GA, whereby new offspring automatically “die” if they duplicate existing values for the objective function, $area(P \text{ XOR } Q)$, within the population. We have found this policy helpful in encouraging diversity and preventing *premature convergence*: i.e. a situation where the population is swamped, at an early stage, by similar or identical individuals.

3.2 Representation and Genetic Operators used for GA1

Vertex subsets are represented by bit strings, $\{b_1, b_2, b_3, \dots, b_i, \dots, b_n\}$, in GA1. A bit, b_i , is set to “1” if and only if vertex i , from the original polygon, is included in the subset. Otherwise, bit i is “0”.

The genetic operations used for GA1 are *one-point crossover* and *point mutation*. One-point crossover involves randomly choosing a single cut point, i . The offspring is then be constructed by combining $\{b_1^1, b_2^1, b_3^1, \dots, b_i^1\}$ from parent 1 with $\{b_{i+1}^2, b_{i+2}^2, b_{i+3}^2, \dots, b_n^2\}$ from parent 2, giving an offspring $\{b_1^1, b_2^1, b_3^1, \dots, b_i^1, b_{i+1}^2, b_{i+2}^2, b_{i+3}^2, \dots, b_n^2\}$. A single point mutation is applied to the new offspring which involves selecting a bit at random and flipping it, either from zero to one or from one to zero.

A potential problem with random bit strings, is that there is no guarantee that the vertex subsets they represent will produce a convex polygon when plotted. Data collected from 1137 polygons sampled to contain 22 points produces only 3.47% solutions that are valid before repair. To overcome this problem and ensure that all estimates of Q are valid (i.e., convex), a problem-specific repair routine has been developed for the GA1.

The repair routine is applied to each random bit string in the initial population, and also to every offspring, following crossover and mutation. The routine accepts, as arguments, an arbitrary subset of vertices from P , as encoded in the bit string, and removes vertices, as necessary, to return the convex hull. The convex hull represents a valid estimate of Q , and it is from this that the objective function, $area(P \text{ XOR } Q)$, is calculated. Whether to write back the original bit string representing the original arbitrary subset of vertices, or the repaired bit string representing the convex hull of that subset, is a matter of choice.

Our tests on small data sets in section 5.1 show that better optimisation was achieved when the repaired bit strings were not written back into the population (see Figure 7). Also, in our lesion rating experiments in section 5.3 we obtained better results (both in terms of the original optimisation criterion and also with the correlation of the measure against expert ratings) if the repaired bit strings were not written back into the population (see table 1). The probable explanation is that the repair operation reduces diversity in the population. There is a many to one mapping between vertex subsets and their convex hulls. Although it is the convex hull of a vertex subset that is used to compute the convexity measure, maintaining the redundancy (i.e. extra vertices) in the GA would appear to provide more scope for exploiting the genetic operators, crossover and mutation, making it easier to escape from local optima.

3.3 Representation and Operators used for GA2

Vertex coordinates from GA1 for the best estimate of Q form the basis of the representation used in GA2. Given that there are m vertices in our best estimate of Q , each chromosome in GA2 will consist of m (x_i, y_i) coordinate pairs. The coordinate values for the GA2 chromosomes are initialized by randomly perturbing the coordinates of the vertices from the GA1 estimate of Q :

$$x'_i = (int)(x_i + r_i \times X) \quad (5)$$

$$y'_i = (int)(y_i + r_i \times Y) \quad (6)$$

where r_i is a random number in the range $f \times [-0.5, 0.5]$, X and Y are the distances between the minimum and the maximum x and y values, respectively, for the vertices in the GA1 estimate of Q , and f is set to 0.05 following some initial experimentation.

One-point crossover is used for GA2, as in GA1, with offspring receiving the first part of its chromosome from the coordinate pairs of parent one, and the final part from the coordinate pairs of parent two. Mutation in GA2 involves the perturbation of a coordinate pair, selected at random, applying the perturbation equations 5 and 6.

The derived polygons are repaired, when required, by the same process as GA1: deleting vertices as necessary to produce a convex shape. Repaired shapes with deleted vertices are not replaced into the population in GA2 (the same policy as GA1).

3.4 Computational Complexity of the GAs

Genetic algorithms belong to a class known as *approximate methods*: i.e., procedures that do not guarantee to find the best solution, but attempt to produce a “good” solution in a reasonable amount of time. Thus, assuming that we “fix” the population size, N , (the inner loop), and the number of generations (the outer loop), it follows that the overall run time complexity will be dominated by the computational effort required within the nested loops. This activity consists of eight components: selection of parents, crossover, mutation, repair, evaluation, checking for duplicates and finally, a “survival test,” to check whether new individual is better than its weaker parent. The selection of parents and the “survival test” both run in constant time, with respect to the number of vertices, n , while the crossover and mutation operators have linear complexity for GA1 and GA2. To repair the offspring requires computing the convex hull. For simple (non-intersecting) polygons the computation complexity is $O(n)$ [11]; however, the process of generating individuals does not guarantee that the ordered sequence of vertices is simple. The more general case of computing the convex hull of a set of n points is $O(n \log n)$. To avoid this extra overhead the candidate polygon Q could be tested for simplicity. In theory this can be performed in linear time as a consequence of Chazelle’s linear-time triangulation algorithm [5]. However, in practise, we have found experimentally that non-simple polygons are generated fairly rarely, and it is therefore more efficient to use the linear complexity convex hull algorithm, and check the final solution produced by the GA. If it self-intersects (this occurred in less than 1% of the cases during our testing) then the GA is run again with a more general $O(n \log n)$ convex hull algorithm.

Computing the XORed area for evaluating individuals is the most complex operation in the processing pipeline. Performing polygon intersection operations takes $O(n \log n + k)$ time, where the two polygons have n_1 and n_2 vertices, $n = n_1 + n_2$, and k is the complexity of the output [5]. The number of intersections between a concave and convex polygon is $\max(2n_1, 2n_2)$ [20], and so the overall complexity of the intersection operation is $O(n \log n)$.

Finally, checking for duplicates, which involves simply examining the objective function value for each member of the population, takes constant time for a fixed size population. The complexity of the GA as a whole is thus $O(n \log n)$.

4 Another Convexity Measure

The previous sections have described how an approximation to the desired polygon is computed using a genetic algorithm. An advantage of this approach is that it is straightforward to use the same optimisation framework with alternative fitness functions. Although the main goal of this paper is to define a *symmetric* convexity measure, we also describe in passing how the GA can be easily applied to compute another new convexity measure.

The convex skull of a polygon is its largest convex subpolygon. In a similar way to roundness measures, which use the smallest circumscribed circle and largest inscribed circle [23], the convex hull (CH) and convex skull (CS) could be used together to define another convexity measure, e.g.

$$C_S = \frac{\text{area}(\text{CS}(P))}{\text{area}(\text{CH}(P))}. \quad (7)$$

In section 1 we noted that using the convex hull to measure convexity had the weakness that C_A is unequally sensitive to protrusions while being insensitive to intrusions. The advantage of combining both the convex hull and convex skull is that, like the perimeter based methods, C_S is sensitive to both narrow protrusions *and* intrusions, but is potentially more robust since it uses area based measurements.

Although the concept of the convex skull might be considered similar to the convex hull, its computation is much more difficult and expensive – the fastest known exact algorithm [4] runs in $O(n^9)$ time! A recent paper used a pixel/voxel approach which iteratively removes concavities, to produce a reasonably fast but approximate solution [1]. Instead we will use the framework described in this paper to find the convex skull. The GA should search for polygons with the following properties:

1. the polygon is convex,
2. the polygon is contained in the original shape P , and
3. the polygon has maximum area.

The final property is a goal restricted by the first two properties that are ideally hard constraints. Either of these hard constraints can be easily satisfied, for instance the first by computing its convex hull and the second by taking the intersection with the input polygon P . However, independently applying these processes to repair the candidate solutions is not possible since each invalidates the repair made by the other. That is, computing the intersection after forming the convex hull can make the result concave again, while computing the convex hull after generating the intersection can make the result lie outside P .

Therefore, a different heuristic procedure was applied to convexify a test polygon T . The vertex in T that along with its two adjacent neighbouring vertices made the largest area triangle external to T was first identified. The two neighbouring vertices were then deleted, and the process iterated until T became convex. While there are situations in which T grows, we have found that in most cases T shrinks, i.e. the result is a polygon lying inside T . First intersecting T with P and then applying the convexification process produces reasonably good candidate solutions except that the 2nd hard constraint is occasionally violated. This has been allowed, relaxing the constraint to make it soft, and adding a term to penalise it.

Thus, to summarise, a candidate solution S is repaired, yielding $R = \text{convexify}(S \cap P)$ from which the following fitness function is constructed and maximised: $\text{area}(\text{CS}(R)) - \text{area}^2(R \setminus P)$.

5 Experiments

5.1 GA1 and Optimality

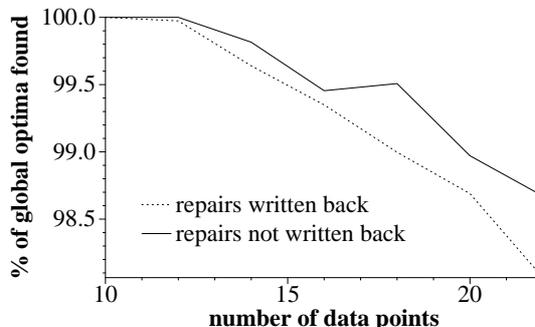


Figure 7: Proportion of optimal solutions found by the genetic algorithms

For very small polygons, a simple exhaustive search can be used to compute Q exactly by exploring all possible subsets of vertices of P . Unfortunately this naïve method is not practical for larger instances due to its run time of $O(2^n)$. Nevertheless, the technique proved useful during the development stages of GA1, providing us with optimum solutions for comparison.

We applied exhaustive enumeration to 1137 test polygons, resampling each several times to contain 10, 12, 14, \dots , 22 points, from which the optimal Q was determined. We then compared these results with results obtained by GA1. The GA was run ten times on each data set, initialising with different random seeds. The proportion of solutions that matched the optimum is shown in Figure 7. Separate tests were carried out to decide whether or not it is better to write back the repaired offspring into the population. We were not able to test GA2 for optimality because the number of possibilities for perturbed coordinates is unlimited.

Clearly, GA1 is highly successful at locating global optima for polygons with small numbers of data points, with the version that does not replace repaired offspring into the population doing slightly better than the version that does replace them. Although it is not possible to extrapolate the exact percentage accuracy for figures with larger numbers of vertices, our requirements are satisfied provided our solutions are of sufficient quality to classify images correctly. Thus, our GAs are biased to produce underestimates of the true convexity measure, but generate results in a realistic time frame. Recently, a dynamic programming (DP) solution has been developed by Kolesnikov and Fränti [9] to determine the optimal polygon, Q . With a time complexity varying between $O(n^3)$ and $O(n^4)$, this approach is clearly more efficient than our naïve exhaustive search, and provides a useful alternative to our GA1 for instances that are not too large.

There is no requirement that there is a unique optimal solution. Tests on 290 polygons, each consisting of 20 vertices, found that about 5% had multiple optima (where solutions with XOR values within 0.2% of the minimum XOR value were considered as essentially optimal). Some examples are shown in Figure 8. Since the XOR values of the multiple solutions are almost identical this means that the C_P scores are also almost identical. C_Q is not

$area(P \text{ XOR } Q)$	=	5462	5452	4323	4329	2479	2480	6843	6833
C_Q	=	0.796	0.783	0.256	0.269	0.836	0.839	0.653	0.650
C_P	=	0.707	0.707	0.254	0.253	0.796	0.795	0.553	0.554
C_{prot}	=	0.063	0.103	0.743	0.729	0.070	0.058	0.263	0.270
C_{int}	=	0.190	0.163	0.009	0.037	0.120	0.129	0.173	0.168

Figure 8: Multiple solutions with near optimal $area(P \text{ XOR } Q)$ values.

so stable, as the alternative Q polygons may have different areas. Nevertheless, C_Q and also the C_{prot} and C_{int} measures tend to have similar values in most cases. The difference in the measure values between corresponding multiple optima averaged over the 14 instances was only $\Delta C_P = .0002$, $\Delta C_Q = .0154$, $\Delta C_{int} = .0382$, and $\Delta C_{prot} = .0394$.

5.2 Qualitative Evaluation

Revisiting Figure 1, the computed convexity values according to the various measures for the pair of shapes are $C_Q = \{0.95, 0.95\}$, $C_P = \{0.96, 0.95\}$, $C_S = \{0.79, 0.47\}$, $C_A = \{0.81, 0.95\}$, $C_L = \{0.91, 0.86\}$, $C_J = \{0.89, 0.84\}$. As expected, the new convexity measure C_Q behaves consistently for the symmetric pair, while C_P is very nearly consistent. All the other measures give differing values for the upper and lower shape, especially C_S which is particularly sensitive to intrusions. The protrusion and intrusion values are $C_{prot} = \{0.04, 0.00\}$ and $C_{int} = \{0.00, 0.05\}$, confirming the results in Figure 5.

The convexity values for the star shape in Figure 3 are $C_Q = 0.40$, $C_P = 0.62$, $C_S = 0.04$, $C_A = 0.13$, $C_L = 0.89$, $C_J = 0.88$. The large area of the convex hull has resulted in very low values for C_S and C_A . The perimeter based measures appear to have overestimated the convexity value. Since there is relatively little overlap between P and Q there is not such a well defined core of the shape, with respect to which the protrusions and intrusions are computed. Thus, these values ($C_{prot} = 0.60$ and $C_{int} = 0.02$) are not so intuitively convincing as in the examples in Figures 1 and 5.

For Figure 5 we get $C_S = \{0.61, 0.61, 0.74\}$. Since the first two shapes have identical convex skulls and convex hulls their convexity values are identical.

The new convexity measure was applied to 1137 polygons from a variety of sources, covering both large and small shapes (between 200 and 5000 pixels), real and synthetic, and computation typically took a few seconds. The population size was 1000, and GA1 was run for 100 generations – this setup also applies to all the results presented in section 5. A sample of the results is shown in Figure 9, ranked in order of increasing convexity. The first point is that the results appear intuitively perceptually correct. It is also of interest to note that for polygons consisting of a mainly convex part with intrusions then the fitted polygon Q (shown shaded) is roughly the convex hull of P . Likewise, for polygons consisting of a mainly convex part with protrusions, Q is roughly the convex skull of P .

Polygons which are stick-like shapes with large concavities (e.g. the first few in the top two rows) are naturally assigned low convexity values. The different effects of the two normalisations can be seen: C_Q ranks such polygons with small area polygons Q lower than C_P , which gives the lowest ratings to those with large area Q .

Examples of C_S are shown over the full range of values. In addition, the bottom row is provided to enable direct comparison with C_Q . It can be seen that the measures differ most noticeably for shapes with large area concavities – these generate relatively small convex skulls (consequently yielding low C_S values) while Q (and therefore C_Q) is less affected.

Of course, the results are not optimal – this is most noticeable in the “F” in the third row in Figure 9. Rerunning the GA ten times with different random initialisations produces similar results about half the time. The other half produces results which have about half the XORed area, such as that shown in Figure 10, and are obviously much closer to the optimum. This leads to the convexity C_Q increasing from 0.175 to 0.515, and C_{prot} decreasing from 0.825 to 0.458.

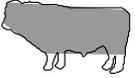
$C_Q =$											
	0.102	0.201	0.300	0.404	0.510	0.602	0.700	0.800	0.900	1.000	
$C_P =$											
	0.156	0.222	0.317	0.403	0.512	0.605	0.700	0.800	0.901	1.000	
$C_{\text{prot}} =$											
	0.000	0.101	0.200	0.301	0.403	0.504	0.603	0.730	0.825	0.900	
$C_{\text{int}} =$											
	0.000	0.100	0.200	0.306	0.406	0.512	0.631	0.800			
$C_S =$											
	0.008	0.105	0.201	0.302	0.401	0.500	0.601	0.702	0.800	0.904	1.000
$C_S =$											
	0.033	0.023	0.037	0.139	0.115	0.193	0.299	0.553	0.718	1.000	

Figure 9: Some polygons P with their convexity values and protrusiveness and intrusiveness values are shown underneath. P is overlaid on a filled version of Q except for the C_S results in which the gray polygon is the convex skull.

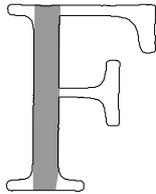


Figure 10: A better fitting Q found by another run of the GA.

5.3 Lesions

We demonstrate the convexity measure on a medical image analysis task. Lee *et. al* [10] presented results for classifying lesions as either benign or malignant melanomas based on the irregularity of their boundaries. They had 14 dermatologists rate a set of 40 lesions from a four point scale according to their probability of being a melanoma. These ratings were averaged over the 14 experts and then compared against various shape measures by computing the Spearman rank correlation between them. Some examples of the mean expert score values are given in Figure 11. Their “overall irregularity index” achieved a correlation value of 0.88, outperforming the alternative measures they considered. However, we have found that the standard convexity measure C_A performs just as well, yielding a correlation value of 0.888. The perimeter based version (C_L) and Žunic and Rosin’s [24] convexity measure (C_J) performed rather poorly, with correlation values of 0.453 and 0.463 respectively. Moreover, applying our new measures we see (table 1) that they do substantially better, reaching a correlation value of 0.958. It is interesting to note that in this application intrusions appear to be substantially more salient than protrusions.

The table also shows the results of experimenting with different schemes for reducing the number of data points. Not only does this improve run time, but by reducing the size of the solution space it could potentially benefit the optimisation process if the subsampling were done in an intelligent manner. Alternatively, if “good” vertices are eliminated (e.g. those belonging to the optimal convex polygon) then the subsampling could result in a poorer quality solution being found. Two approaches were tested: uniform subsampling of the curve, and breakpoints found by polygonal approximation (Ramer’s algorithm [15] was used); many other schemes are possible, e.g. a multigrid approach. With the polygonal approximation algorithm the error threshold (maximum deviation between the curve and polygonal approximation) was successively set to the values $\{1, 2, 3\}$, which over the full data set

was effective to the average sampling rates of $\{0.235, 0.122, 0.081\}$ respectively. It can be seen that the polygonal breakpoints provide better C_Q and C_P results than uniform sampling for equivalent data reduction rates, but that in general, both schemes suffer a loss of performance compared to operating directly on the full data.

In addition to our experiments with different numbers of data points, table 1 also shows the results obtained by applying different variations of our genetic optimisation process. For the complete data, row 1 contains the correlation scores for a version of GA1 where the repaired offspring are written back into the populations, row 2 covers results for GA1 where the repaired solutions are evaluated but not replaced into the population, and row 3 gives the results obtained by applying GA2 (i.e. perturbed solutions) to the output of GA1 (repairs not written back). For all the sampled data sets, GA1 (with repairs not written back) occupies the first row of results, with the GA1 followed by GA2 sequence occupying the second row. Examination of the mean error column clearly indicates that the application of GA2 produces better results than are obtained from GA1 alone. However, it is clear from the correlation data that improved optimisation does not necessarily lead to improved classification.

Measuring convexity using the convex skull based measure (C_S) applied to the complete pixel data gave a correlation value of 0.948, which is almost as good as the result from C_Q and C_P . Again, to emphasise the importance of a good repair process, if the same repair mechanism is used as for C_Q and C_P , i.e. taking the convex hull of the candidate solution: $R = CH(S)$, then the resulting polygons are generally cruder approximations to the convex skull, and the correlation value of the measured convexity is substantially reduced to 0.700.

Method	C_Q	C_P	C_{int}	C_{prot}	mean error
complete data					
GA1 repairs written back	0.946	0.948	0.856	0.756	2303
GA1 repairs not written back	0.958	0.958	0.927	0.830	2262
GA1 and GA2	0.958	0.958	0.927	0.834	2259
data subsampling rate: 0.1					
GA1 (repairs not written aback)	0.933	0.937	0.869	0.777	2021 (2375)
GA1 and GA2	0.924	0.927	0.877	0.776	2011 (2456)
data subsampling rate: 0.05					
GA1	0.828	0.833	0.809	0.693	1873 (2537)
GA1 and GA2	0.820	0.828	0.792	0.692	1859 (2601)
polygonal approximation of data; $t = 1$					
GA1	0.926	0.929	0.870	0.718	2251 (2305)
GA1 and GA2	0.946	0.949	0.886	0.723	2248 (2399)
polygonal approximation of data; $t = 2$					
GA1	0.936	0.939	0.860	0.751	2293 (2390)
GA1 and GA2	0.945	0.948	0.879	0.748	2282 (2447)
polygonal approximation of data; $t = 3$					
GA1	0.944	0.941	0.781	0.661	2364 (2508)
GA1 and GA2	0.945	0.948	0.827	0.647	2334 (2502)

Table 1: Absolute Spearman rank correlation scores for lesion data. Mean XOR error computed on the input data is given; where subsampled data was used the XOR error with respect to the original full data is shown in brackets.

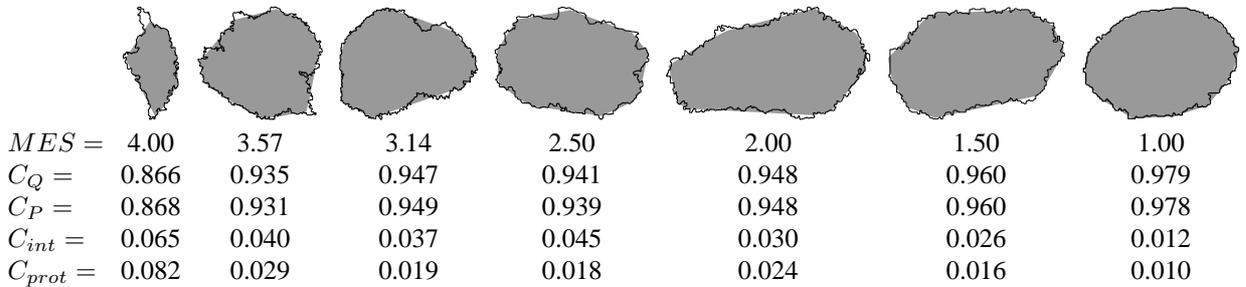


Figure 11: Examples of lesions covering the full range of mean expert score (MES) values, along with their computed shape measures.

5.4 Greebles

Among the 1137 polygons tested were 53 “greebles”; some examples are shown in Figure 12. Greebles are a popular source of test objects used in psychology, e.g. [8]. They are objects synthesised to a standard configuration: a vertically-oriented body with four protrusions: two “boges”, a “quiff” and a “dunth”. Since the appearance of members within this class is qualitatively similar, despite their individual differences, their measured shape attributes should also be similar. The measures are compared by determining the ranks of the greebles in the set of 1137 polygons ordered by each of the convexity measures. The standard deviation of the greeble ranks was 68.9 for C_Q , 69.0 for C_P , and 78.7 for C_S . In comparison, C_A and C_L produced standard deviations of 89.3 and 113.6, while C_J produced 96.5. The lower values of the new measures demonstrates their high level of stability and consistency.

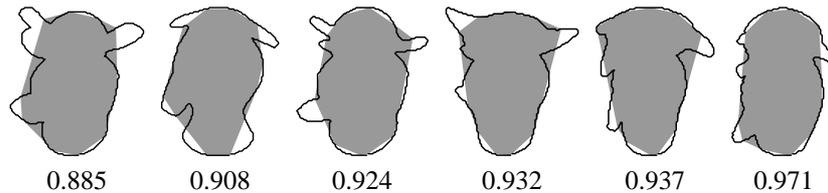


Figure 12: A selection of greebles covering their full range of C_Q convexity values.

6 Summary and Conclusions

A new area-based convexity measure for polygons has been defined, and a genetic algorithm based solution has been proposed as an efficient computation tool for this measure, given that exact enumeration is computationally expensive. Compared to the traditional convexity measure our new measure is more robust and symmetric, and has been shown to outperform it on a medical analysis task involving the estimation of the likelihood of melanoma from lesion boundaries. In addition, the determination of the best fit convex polygon for the convexity measure leads to the simple computation of a further two measures, quantifying the degree of protrusions and intrusions present in the input polygon.

A minor modification to the GA – namely altering the fitness function and the candidate solution repair mechanism – enabled an efficient approximate solution to the convex skull problem. This in turn provided an additional convexity measure to be computed. Unlike C_Q (and to a lesser degree C_P), C_S does not have the appealing symmetry property. Nevertheless, it performed well on the medical classification task – almost comparable with C_Q and C_P . Although its performance on the greebles was not as good as C_Q and C_P it was better than the remaining convexity measures.

Our genetic solution can be applied in two consecutive stages: GA1 followed by GA2. GA2 can be regarded as “fine tuner” and is optionally applied to the output produced by GA1. Both GAs search for a convex shape to match the outline of the original shape as closely as possible. Although GA1 is confined to the original vertices, GA2 is allowed to perturb the coordinates of vertices, so that the derived shape “drifts,” in an attempt to match the outline of the original shape even more closely. Results clearly indicate that the application of GA2 improves the results by producing a lower XORed error than is the case when GA1 is applied on its own. However, there is no evidence that this improved XORed error leads to better classification of data. Furthermore, experiments in preprocessing the polygons to reduce the number of vertices before applying the genetic algorithms resulted in some loss in the quality of the solution, demonstrating that it was better for the GAs to operate directly on the full data set.

We have seen that, because GAs are approximate methods, some solutions produced by GA1 are suboptimal. Although optimal solutions (in which Q ’s vertices are limited to a subset of P ’s vertices) could be obtained using dynamic programming, this is computationally expensive [9]. A compromise between optimality and computation time would be to improve the initial GA1 solution using dynamic programming over a limited domain. If each vertex in the initial GA1 solution is restricted to lie in a window of size L positions, then using dynamic programming only $O(L^2n)$ combinations need to be tested to find the optimal solution with those constraints [6]. However, it should be emphasised that the result is still not guaranteed to be a global optimum when L is less than $O(n)$.

Finally, the methods described could be applied to 3D data in a straightforward manner. Although by selecting random polyhedron vertices the GA loses the topological structure of edges and faces, this will be recovered when the 3D convex hull algorithm reconstructs a (convex) polyhedron from these unstructured points. In addition,

relatively efficient techniques for performing boolean operations on polyhedra exist [7]. Along with the calculation of polyhedra volumes, this covers all the essential components of the method.

7 Acknowledgements

We would like to thank Alan Murta for his “Generic Polygon Clipper” code, Ben Kimia for providing some of the polygon data, and Alexander Kolesnikov for advice.

References

- [1] G. Borgefors and R. Strand. An approximation of the maximal inscribed convex set of a digital object. In *Int. Conf. Image Analysis and Processing*, pages 438–445, 2005.
- [2] L. Boxer. Computing deviations from convexity in polygons. *Pattern Recognition Letters*, 14:163–167, 1993.
- [3] D.J. Cavicchio. *Adaptive Search Using Simulated Evolution*. PhD thesis, University of Michigan,, 1970.
- [4] J.S. Chang and C.K. Yap. A polynomial solution for the potato-peeling problem. *Discrete Comput. Geom.*, 1:155–182, 1986.
- [5] B. Chazelle. Triangulating a simple polygon in linear time. *Computational Geometry*, 6:485–524, 1991.
- [6] J.G. Choi, S.W. Lee, and H.S. Kang. Dynamic programming approach to optimal vertex selection for polygon-based shape approximation. *IEE Proc.-Vis Image Signal Process.*, 150(4):287–291, 2003.
- [7] D. Eppstein. Asymptotic speed-ups in constructive solid geometry. *Algorithmica*, 13(5):462–471, 1995.
- [8] I. Gauthier, P. Williams, M.J. Tarr, and J. Tanaka. Training greeble experts: a framework for studying expert object recognition processes. *Vision Research*, 38:2401–2428, 1998.
- [9] A. Kolesnikov and P. Fränti. Optimal algorithm for convexity measure calculation. In *Int. Conf. Image Processing*, pages 353–356, 2005.
- [10] T.K. Lee, D.McLean, and M.S. Atkins. Irregularity index: A new border irregularity measure for cutaneous lesions. *Medical Image Analysis*, 7(1):47–64, 2003.
- [11] D. McCallum and D. Avis. A linear algorithm for finding the convex hull of a simple polygon. *Inform. Process. Lett.*, 9:201–206, 1979.
- [12] C.L. Mumford. A simple approach to evolutionary multi-objective optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications*. Springer Verlag, 2004.
- [13] C.L. Mumford. Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In *GECCO*, volume LNCS 3102, pages 1389–1399. Springer, 2004.
- [14] C.L. Mumford-Valenzuela, J. Vick, and Y. Pearl. Heuristics for large strip packing problems with guillotine patterns: An empirical study. In D.Z. Du and P.M. Pardalos, editors, *Metaheuristics: Computer Decision-Making*. Kluwer Academic Press, 2003.
- [15] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.
- [16] P.L. Rosin. Measuring shape: Ellipticity, rectangularity, and triangularity. *Machine Vision and Applications*, 14:172–184, 2003.
- [17] P.L. Rosin and C.L. Mumford. A symmetric convexity measure. In *Int. Conf. Pattern Recognition*, pages IV: 11–14, 2004.
- [18] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Chapman and Hall, 1993.
- [19] H.I. Stern. Polygonal entropy: a convexity measure. *Pattern Recognition Letters*, 10:229–235, 1989.

- [20] S. Suri. Polygons. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, 1997.
- [21] G Syswerda. Uniform crossover in genetic algorithms. In *Proc. Third Int. Conf. on Genetic Algorithms*, pages 2–9. Lawrence Erlbaum Associates, 1989.
- [22] C.L. Valenzuela-Mumford. A simple evolutionary algorithm for multi-objective optimization (SEAMO). In *Congress on Evolutionary Computation (CEC)*, pages 717–722, 2002.
- [23] L. Van-Ban and D.T. Lee. Out-of-roundness problem revisited. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):217–223, 1991.
- [24] J. Žunić and P.L. Rosin. A new convexity measurement for polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):923–934, 2004.