Turning Shape Decision Problems into Measures

Ralph R. Martin, Paul L. Rosin
Dept of Computer Science
Cardiff University

Abstract

This paper considers the problem of constructing shape measures; we start by giving a short overview of areas of practical application of such measures.

Shapes can be characterised in terms of a set of properties, some of which are Boolean in nature. E.g. is this shape convex? We show how it is possible in many cases to turn such Boolean properties into continuous measures of that property e.g. convexity, in the range [0–1]. We give two general principles for constructing measures in this way, and show how they can be applied to construct various shape measures, including ones for convexity, circularity, ellipticity, triangularity, rectilinearity, rectangularity and symmetry in two dimensions, and 2.5D-ness, stability, and imperforateness in three dimensions. Some of these measures are new; others are well known and we show how they fit into this general framework.

We also show how such measures for a single shape can be generalised to multiple shapes, and briefly consider as particular examples measures for containment, resemblance, congruence, and similarity.

Keywords: shape measures, shape properties, convexity, circularity, ellipticity, triangularity, rectilinearity, rectangularity, symmetry, containment, resemblance, congruence, similarity, 2.5D, stability, genus, hole.

1 Introduction

1.1 Shape Measures

This paper is concerned with shape measures—numbers which characterise various properties of a geometric shape. Some, such as invariant moments [13], are already well known and well defined, whereas others such as "circularity" are intuitively understood even if they do not have a single precisely agreed meaning—in many cases, ad-hoc definitions have been proposed. The importance of finding shape measures that are simple to compute, with intuitive meanings, has already been noted by Peura [31].

This paper gives a *principled, general approach* to defining a wide class of shape measures, based on Boolean shape properties. Thus, we show how to take methods for deciding whether a shape is, for example, convex, rectilinear, or circular, and use these to define continuous measures of convexity, rectilinearity, and circularity. However, our approach is *not* suitable for other measures such as compactness or elongation (there is no commonly accepted Boolean test for compact or elongated), or invariant moments (including area), Fourier descriptors, and eigenshapes (there is no Boolean property associated with these). We also note that defining measures is not the only way to characterise shapes, and other approaches such as fuzzy set theory and scale space methods can also be used.

For simplicity, we mainly consider various two dimensional measures; some of these measures also work directly in three dimensions without modification. We also present several three dimensional measures.

It is possible to make a distinction between *boundary* features and *region* features, depending on whether the measure is computed using only the boundary of the shape, or using its interior too. Our method is applicable to both. Note that this is a different distinction to *local* and *global* features, which either consider properties within a small neighbourhood, or within the whole shape. Again, our method is applicable to both.

While the examples in this paper are presented in terms of shapes whose boundaries are mathematical curves (polygons, closed smooth curves, etc.) or surfaces, our approach is equally applicable to shapes defined in terms of sets of pixels, voxels, or chain codes, for example.

Some of the measures we derive in this paper are well known, while others appear to be new. We consider the main contribution of this paper to be the set of meta-methods we propose for generating measures, not the individual measures given as examples—although some of these may be of interest in their own right. A wide range of examples is given to justify the wide applicability of our meta-methods.

1.2 Uses of Shape Measures

Numerical shape measures have various uses [31]. Exact computations with real numbers are not reliable in the presence of noise in measured data, or floating point arithmetic in numerical algorithms, under which circumstances a numerical measure of a property is more robust than a Boolean decision. For example, an object is defined to be *rectilinear* if all of its internal angles are a multiple of 90° [43]. However, if data is collected from a digital image, and lines are fitted to a detected polygon, it is unlikely that its angles will exactly meet this requirement. Instead, a *continuous measure* of rectilinearity used in conjunction with a threshold may give us a more satisfactory means of identifying rectilinear polygons in images.

A second use of shape measures which generalise Boolean tests is for defining manufacturing tolerances, or assessing the goodness of manufacture of objects. Tolerances may be defined in terms of form, size, and position [33], where *form* refers to object shape. A suitable continuous measure of shape, such as circularity (more commonly termed *roundness* in metrology [25]), will let us specify how far from circular a manufactured object may be, while still acceptable.

Shape measures have many other uses [22, 38]. In geometric search algorithms, shape measures may be useful for prioritising cases for consideration in a best-first approach. A common approach to shape recognition [32] and retrieval of similar shapes from a database [7] computes a descriptive vector of numbers (shape measures) which represent a shape, and then seeks shapes with similar vectors. While invariant moments provide a set of numbers useful for representing shape, their interpretation is not entirely intuitive. Instead, if the elements of the vector are quantities such as circularity, convexity and other measures of the type we propose, a "naive" user can have an intuitive understanding of what the vector means, and can refine a database search by increasing or decreasing a chosen component of the vector, or weighting the relative importance of the different components.

Shape measures can also be used in shape classification, i.e. deciding which class a given shape belongs to, or even that it belongs to no existing class. This can also be done by using

a vector descriptor of shape measures, together with some metric for measuring distances between a shape and suitable examples or distributions of shapes [4].

1.3 Measures of Multiple Shapes

Our approach can readily be extended to conjoint properties of multiple shapes. A simple example of a Boolean property of two shapes A and B is *inside*: is A inside B? This can be extended to a continuous property, the "insideness" (i.e. degree of containment) of A in B.

Such properties also have practical applications. For example, algorithms to *register* or align [3] two shapes can be based on optimising a transform which positions and orients A such that it has maximum insideness with respect to B (and perhaps that simultaneously, B has maximal insideness with respect to A).

An even simpler property of two shapes is *resemblance*, the continuous property generated from the question "Are A and B the same?". (We avoid using the natural word "similarity" for resemblance—the concept of two shapes being *similar* has a precise geometric meaning, which allows congruence compounded with scaling. Here we use resemblance as the continuous property derived from *congruence*.) Resemblance measures may be used in shape simplification [38], for example, where various candidate simplified shapes can be generated from a given input shape, and the best one (the one with highest resemblance) selected for further simplification. Veltkamp [38] has an extensive discussion of desirable properties for a measure of resemblance, and points out that finding a single measure satisfying all of these requirements is not possible.

1.4 Non-uniqueness of Shape Measures

An important starting point is that several different algorithms may exist for deciding a single Boolean property of a shape. For example, two (of many possible) definitions of convexity of a polygon are (a) every straight line joining two points on the boundary lies entirely within the polygon, (b) each of the polygon's internal angles is less than 180°. Using our approach to produce measures from these two definitions leads to two different measures for convexity—for example, a given non-convex polygon might have a convexity measure of 0.8 according to one of these measures, and 0.9 according to the other.

While at first sight, this may appear to be a weakness of our approach, we claim that the converse is true. If a shape is non-convex, for example, different aspects of its non-convexity may be important for different applications—is it the *number* of "indentations" in the boundary of the polygon which is important? Or is it their overall *size* which matters? A measure based on definition (b) above will be more suitable if the first of these two requirements is more important. Having multiple measures for the same property lets users choose the measure most suited to their requirements.

This choice can be made in several ways. One is to draw up a set of axioms the measure should satisfy, and reason which definition, if any, produces a measure with the required properties. A less rigorous approach is to experimentally compute each measure for a set of test shapes, and to choose the measure which produces results closest to those desired for a given application. Also relevant are the computational effort required, and the robustness of those computations, for the different measures.

If we are using a vector of measures for shape matching (for example), having multiple measures for the same property can be desirable—using a larger vector of measures will gen-

erally provide greater discriminatory power between shapes, albeit at the expense of greater computing time.

1.5 Organisation of Paper

The rest of the paper is organised as follows. Section 2 presents our general method for turning shape decisions into shape measures. Various Sections then illustrate the use of this method to produce measures for a variety of properties of both single shapes and multiple shapes. In some cases we define several measures for a single property, as discussed above. We also note where the measures we derive are already known in the literature. We end the paper with some conclusions.

2 Generating Shape Measures

We now explain our method for generating a continuous shape measure from a Boolean shape property.

2.1 The Decision Property

We require that the Boolean shape property (e.g. "Is the shape convex?") can be decided with an algorithm which makes a number of tests, and makes a decision in each case. We call these tests the *decision tests*. If the object fails *any* test, this determines that the shape does not have the given property (see Algorithm 1). (We must of course be careful that the tests are not only necessary tests, but that the overall test is sufficient for the property we wish to determine. For example, testing if all diameters of a shape are the same is not sufficient to show that a shape is circular—other shapes also exist with constant diameters, such as the Reuleaux Triangle [34].)

Algorithm 1 Decision property meta-algorithm

```
property ← true
for all decision test cases do
  if result-of-this-decision-test = false then
    property ← false
    exit for
  end if
end for
return property
```

Our method works directly when there is a finite number of tests, but it can also be extended to work in cases where the algorithm is only an "in-principle" one which must test an infinite number of cases, as will be shown later.

We now give a simple concrete example. As noted in Section 1.4, we can test if an object is convex by testing if *all* straight lines, both of whose ends lie on the boundary of the object, lie entirely within the object . (This is an in-principle algorithm which requires an infinite number of tests.)

Note that the locality of the decision tests may vary. For example, they may be based on point samples, or on information within a small ball of given size centred on a sample point.

Alternatively, they may be based on extended structures, such as lines which cut across the shape, or a computation involving several sample points from different parts of the shape (not lying within any small ball).

2.2 Simple Measure Algorithms

The simplest form of our approach converts the decision algorithm into an algorithm for computing a shape measure (see Algorithm 2) by replacing the "reject-property-if-test-failed" step in the decision algorithm by keeping count of the cases which pass the test, and dividing by the total number of cases.

Algorithm 2 Shape measure meta-algorithm

```
count ← 0
count-of-true ← 0
for all decision test cases do
  count ← count+1
  if result-of-this-decision-test = true then
    count-of-true ← count-of-true+1
  end if
end for
return count-of-true/count
```

Clearly, Algorithm 2 must be modified when the number of tests is infinite. Two possible approaches which allow us to compute a measure in such cases are as follows. The first is to analytically compute an equivalent result, by integrating a suitable *characteristic function*, which returns zero or one, over appropriate limits. The second is to approximate the measure by using a Monte Carlo method, i.e. using a finite number of tests to sample the infinite number of possible tests, and computing the ratio of tests returning true to the total number of sample tests. The samples should cover the shape in a representative manner; they can be chosen at random, or using a low-discrepancy sequence for greater efficiency (or accuracy) [5]. Even if we can formulate the measure as an integral, we may still have to resort to computing it numerically as in the second approach.

Not all Algorithms of Type 2 are useful, as they may return a shape measure of zero for most objects if the Boolean test used is likely *not* to be satisfied in most of the individual tests when the shape is not exactly of the desired kind. A good example is provided by a test for circularity, which asserts that all points of the shape are the same average distance from its centre. For many non-circular shapes, only a small finite set of points will have this average distance out of the infinite number of points on the curve. This will be discussed further in Section 4.1.

2.3 Weighted Measure Algorithms

Algorithm 2 can be extended by weighting cases according to importance in some way. One generally useful approach replaces the individual Boolean tests by new tests which return a continuous result in the range 0–1 which indicate by how much the shape fails an individual test, rather than a Boolean answer. Then we sum these continuous results, rather than counting the number of true results, as shown in Algorithm 3.

Algorithm 3 Weighted shape measure meta-algorithm

count $\leftarrow 0$
amount-true $\leftarrow 0$
for all decision test cases **do**
  count $\leftarrow$ *count*+1
  amount-true $\leftarrow$ *amount-true*+*result-of-this-decision-test*
end for
return *amount-true*/*count*

To give a concrete example, using the same convexity definition as before, we replace the Boolean test "Given two points on the boundary of the object, is the line between those points wholly inside the object?" with the numerical test "Given two points on the boundary of the object, what fraction of the line between those points lies inside the object?"

We will write r as shorthand for *result-of-this-decision-test*. We note that *averaging* the r values is an arbitrary computation (for example, it might be more useful in some contexts to take the *minimum*), but there again, so generally is the *result-of-this-decision-test* function itself. Note that as r lies in the range $0 \leq r \leq 1$, then r^2 is also a valid *result-of-this-decision-test* function, as it too lies in the same range: $0 \leq r^2 \leq 1$. Indeed, an infinite number of other transformations of r will also yield a new function of the same type. Clearly, choosing different functions will lead to different measures, and can even be used to control how a measure responds to different types of shape.

When designing a Boolean test for some shape property, it is clear that the Boolean tests involved must be robustly computable for the test to work. This will not be the case if, for example, the tests involve equality of real numbers. (E.g. an object is rectilinear if all its internal angles are 90°; this can not be robustly decided if the angles are real numbers arising from an image processing process.) This is also a problem for measures of Type 2, as Boolean decisions are still needed. However, a method which is not robust in these cases *can* be made into a robust algorithm of Type 3, as the Boolean decisions are replaced by numerical values (for rectilinearity, we now consider discrepancies from 90°). Thus, mathematical definitions which are *not* algorithmically suitable for testing a Boolean property may still be suitable for computing a *measure* of a property.

2.4 Measures for Multiple Shapes and Other Generalisations

As noted in the Introduction, our approach can readily be extended to conjoint properties of multiple shapes. For example the Boolean property of two shapes, B *is inside* A, can also be computed by a set of decision tests in an Algorithm of Type 1—"Does every point of B lie inside A?" This can be extended to a continuous property, the degree of containment of B in A, by counting the fraction of points of B inside A in an Algorithm of Type 2. (This is also an example of an infinite number of tests being reduced to an appropriate integral. This measure is equal to the area of B lying inside A divided by the total area of B.)

Our methods may be extended in other ways, or used as the cores of more complicated algorithms, as will be shown later when we consider the *resemblance* of two shapes, and the geometric *similarity* of two shapes.

2.5 Discussion

As previously noted, one advantage of Algorithms of Type 3 over Algorithms of Type 2 is that they are more robust. Because they are compounded from individual continuous measures rather than Boolean tests, they degrade gracefully in the presence of noisy input data and computational inaccuracies.

Algorithms of Types 2 and 3 result in a measure with a value of *one* for a shape which exactly satisfies the decision algorithm, e.g. a convex object has convexity measure 1. The measure tends to zero as fewer and fewer of the cases satisfy the individual decision tests in a binary or numerical sense respectively.

As noted earlier, different decision algorithms for the same Boolean property lead to different measures. Various considerations are important when choosing between alternative measures

- How robustly and reliably can the individual decision tests be performed? E.g. an algorithm based on estimating curvatures from discrete data on the boundary of an object is likely to be less robust than one sampling points in the interior of an object.
- How complex is each decision test?
- How many decision tests need to be performed? Time complexity analysis can answer this question in the case of a finite number of tests. If integrals are approximated by sampling methods, the expected accuracy obtained will decrease with the number of variables of integration, for a given amount of computational cost. For example, a method based on sampling triples of points from an object boundary will, all else being equally likely, give a less robust measure than a method based on pairs of points.
- How stable is the measure? In other words, if a small perturbation is made to the shape, does it cause a small or large change in the measure? Most reasonable definitions of measures are stable e.g. the definitions for circularity given later can readily be seen to be stable. However, a measure based on, say, counting the number of inflection points in a curve would be unstable if a part of the curve is almost linear. Further questions of this type can be asked concerning the continuity of the measure as a map, as the input shape changes; they need to be analysed on a case by case basis.
- Does the measure characterise local or global deviations from a shape property? A circle with a thin notch cut out of it has a local distortion, whereas a circle stretched slightly in one direction to become an ellipse has a global distortion. Some measures may be better at capturing the former, others the latter.

An alternative to *choosing* between different measures for a given property is to *combine* the results of two or more different types of test in some way within an Algorithm of Type 3 to produce a further measure with desired properties.

Shape measures are generally most useful if they are invariant with respect to position and orientation, and for certain applications, scale too. Ones based on our approach will always have these properties for Algorithms of Type 2, and will do so for Algorithms of Type 3 provided that the individual tests do, which will normally be the case.

The methodology of accumulating evidence based on many individual samples or measurements is not new. In particular, the Hough transform [17] is related to our approach.

The difference between the two is that the Hough transform is commonly used to estimate geometric parameters by accumulating evidence for hypotheses and then searching for evidence with maximal support. In contrast, the measures in this paper are not concerned with recovering geometric models from the data, but focus on the level of support as an end rather than a means.

Some rather general classes of algorithms can be seen, at least in a trivial sense, as examples of our method. For example, suppose we wish to measure how well a set of points lie on some class of curve (say a circle). We can find the best curve of the desired class. The Boolean property "Do these points lie on a circle?" can be turned into a Type 3 algorithm by using as the individual tests "How far is each point from the best-fit curve, normalised by (say) the maximum distance between any two points in the set?". Clearly, this is not a new idea: for example, see [36], but does suggest the generality of our framework.

A further class of algorithms is suggested by the principle that a shape S belongs to a class of shapes X (triangles, ellipses, etc.) if and only if, when we compute the smallest bounding X around S, all points of X lie inside S. This can be converted to a Type 2 algorithm by finding the ratio of the area of S to the area of X [36]. A particular example of this principle is discussed under convexity below.

In the following Sections, we now consider various explicit Boolean properties of shapes, and shape measures based upon them.

3 Convexity

This section gives a (by no means exhaustive) set of definitions of convexity of 2D shapes which are then used to define convexity measures. Some of them are boundary measures, whilst others are interior measures. Some of them are specific to 2D shapes, whilst others work equally well in higher dimensions. Some of them are specific to polygons, whilst others work for general closed curves.

Various convexity measures have been proposed previously, for example the ratio of the perimeter of the convex hull to the perimeter of the object [31]. Some proposed convexity measures are quite complex [29, 37, 44], whereas others have been devised with efficiency of computation in mind [2, 10].

3.1 Internal Angles are Less Than 180°

A simple polygon with n vertices and no holes is convex if all of its internal angles a_i are less than 180°. Based on this, an algorithm of Type 2 simply computes the convexity C by counting the number of internal angles less than 180°, and dividing by n.

This can be converted to an algorithm of Type 3 by taking into account by how much each internal angle greater than 180° exceeds 180°:

$$C = \sum_{i=1}^{n} \frac{\min(1, (360 - a_i)/180)}{n}$$

Internal angles of less than 180° are given a value of 1 in the sum, while internal angles greater than 180° are given a value which linearly falls from 1 to 0 as the angle increases from 180° to 360°.

Clearly, the second of these two measures always gives a higher value than the first, as the first measure uses a value of 0 in the sum in place of a value between 0 and 1 for all angles exceeding 180°.

The Type 2 measure can be generalised in various obvious further ways, e.g. by taking into account lengths of edges adjacent to each vertex, and the total perimeter length of the polygon.

These algorithms are efficient, as they take time $O(n)$.

3.2 Lines between Vertices are Contained

A simple polygon with n vertices and no holes is convex if all lines connecting pairs of vertices lie entirely inside the polygon. The fraction of the total number of such lines entirely within the polygon gives a Type 2 measure, while a Type 3 measure can be obtained by summing the fraction of each line's length which lies within the polygon.

These convexity measure algorithms are less efficient than the ones in Section 3.1, as the number of lines connecting all pairs of vertices is $O(n^2)$, illustrating the importance of carefully choosing the initial definition upon which to base a shape measure algorithm.

A related but more efficient algorithm can be obtained by using the definition that each line connecting the two vertices on either side of each vertex must lie entirely inside the polygon. An algorithm of Type 2 based on this definition would give exactly the same results as an algorithm of Type 2 based on internal angles. However, this would not be true of an algorithm of Type 3.

Rather than restricting the lines to connections between pairs of vertices it may be desirable to place sample points on the boundary at equally-spaced intervals. Vertex sampling can be seen as the limiting case of *weighted* sampling of the boundary of the shape with more samples in regions of high curvature.

3.3 All Points Inside Convex Hull are Inside

The test "Are all points inside the convex hull of the shape also inside the shape?" is a trivial test for convexity. However, it is not quite so trivial when used as a basis for a convexity measure. It is also an example of a shape measure involving an infinite number of tests—there are an infinite number of points inside a shape.

A Type 2 measure is easily derived as the ratio of the number of points inside the object to the number of points inside its convex hull. Although both are infinite sets, the ratio is well defined, and is simply the ratio of the object's area to the area of its convex hull, a measure of convexity which has been widely used before.

Using this property to produce a Type 3 measure is less obvious. One way in which it could be done is by using a "result-of-this-decision-test" function with value 1 for points inside the object, and a value ranging from 1 down to 0 for points lying between the object and its hull, linearly interpolating according to distance between the object boundary and the hull. (This value would not be a good choice for highly convoluted objects like a spiral, however.)

3.4 Sign of Curvature of Boundary

At all points of the boundary of a convex curve (containing no straight line segments), the curvature has the same sign. This is a natural extension of the simpler test described earlier for polygons—"Is each of the polygon's internal angles less than 180°?", as the turning angle

of a closed curve is just the integral of its curvature with respect to arc-length; edges of a polygon have zero curvature while its vertices represent delta functions in the curvature.

Again, this uses an infinite number of points, but this time on the boundary of the shape rather than in its interior. To express this analytically, we use integrals involving the arc-length of the shape's boundary. If we assume that the boundary is a parametric curve $\mathbf{r}(u) = (x(u), y(u))$ with at least C^2 continuity, and no straight line segments, then the curvature can be computed at each point by

$$\kappa(u) = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}},$$

The sign of curvature changes at inflection points, for which $\kappa(u) = 0$. The corresponding values of u can be found by solving this equation. Let us put these values, and the start and end values of u, into an ordered list $u_0, u_1, u_2, u_3, \ldots, u_n$. (Care must be taken to insert multiple roots multiple times correctly.) Let us suppose the curve is convex (turns to the left when going round it anticlockwise) between u_0, u_1, between u_2, u_3 and so on, and has curvature of the opposite sign between u_1, u_2 etc. (The alternative case where the curve starts off bending the opposite way is similar.) The arc-length of any section of the curve is

$$s = \int_a^b |\mathbf{r}'(u)| du,$$

so we may define a measure of convexity to be

$$C = \frac{\sum_{i \text{ even}} \int_{u_i}^{u_{i+1}} |\mathbf{r}'(u)| du}{\int_{u_0}^{u_n} |\mathbf{r}'(u)| du},$$

or in other words, the ratio of the length of the curve which has convex curvature to the total length of the curve.

This definition of convexity leads to an integral which can be computed algorithmically (with some difficulty) for common classes of curves like B-splines. However, it is not well-defined for polygons or other curves having straight-line segments, and has the further disadvantage that in the limit it computes a lowest convexity of 1/2, rather than 0, for highly concave and convoluted shapes (consider where the inflection points might lie in shapes like Figure 1).

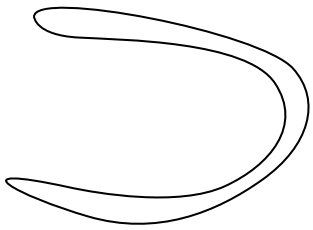

Figure 1: A highly concave object.

3.5 All Straight Lines are Contained

This test is a generalisation of the convexity test given in Section 3.2—but now, all lines joining any two points *inside* the shape, rather than just on its boundary, must lie entirely within the shape.

There is no obvious simple analytical way to compute the fraction of contained lines which lie entirely within the shape to produce a Type 2 measure, but it is certainly possible to use a Monte-Carlo method to estimate this measure, using the approach in Algorithm 4.

Algorithm 4 Type-2 algorithm for convexity measure based on line containment

```
count ← 0
count-of-true ← 0
for number-of-times-desired do
  count ← count+1
  p1 ← random-point-inside(shape)
  p2 ← random-point-inside(shape)
  l ← line(p1,p2)
  if not intersects(l,boundary(shape)) then
    count-of-true ← count-of-true+1
  end if
end for
return count-of-true/count
```

Random points inside the shape can be generated by generating points inside its bounding box and discarding those not inside the shape, for example.

A Type 3 measure based on this definition of convexity can easily be computed with a similar algorithm: divide the sum of the fractional lengths of each random line which lies entirely within the shape by the total number of random lines.

3.6 No Line Intersects More Than Twice

For another measure for convexity, we use the definition: an object is convex if and only if every infinite line intersects the object's boundary in no more than two places.

For a finite sized object, the set of infinite lines which intersect the object *at all* is clearly of measure zero. To make a practical Type 2 Monte-Carlo algorithm based on this definition, we need to restrict the set of test lines to some set which have a chance of intersecting the object. We may choose to enclose the object with the smallest bounding sphere whose centroid is at the centroid of the object, and then just generate random lines which intersect the sphere. A method for generating such lines uniformly is given in [20]; a further refinement also given there is to generate a low-discrepancy distribution of lines to provide better sampling.

Our Type 2 algorithm now counts the number of such lines with at most two intersections, compared to the total number of lines sampled.

A Type 3 algorithm instead could count 1 for each line with two or less intersections, and $2/n$ for lines with n intersections, compared to the total number of lines sampled.

3.7 Polygon Equals Kernel

To produce a final example of a convexity measure, we start from a less obvious idea. A *star-shaped* polygon is one for which there exists at least one interior point from which the whole boundary is visible. For most star-shaped polygons, there is an interior region satisfying this visibility condition; it is called the *kernel* of the polygon. For convex polygons, the kernel of the polygon is the whole polygon itself.

Thus, a Type 2 measure for convexity computes the number of points in the kernel divided by the number of points in the polygon, i.e. the relative areas. This measure has a value of 1 for convex polygons, a measure of between 0 and 1 for star-shaped polygons, and a measure of zero for all polygons which are not star-shaped.

This measure can be computed efficiently, as linear time algorithms are known for computing the kernel of a polygon [18], and for the areas of polygons.

3.8 Summary

These different measures for convexity illustrate various points made earlier in Section 2. Some apply to a wider class of input shapes than others: three out of the seven alternatives are applicable only to polygons. Of those, two of them, the measures in Sections 3.1 and 3.7, have a computational cost of $O(n)$, where n is the number of polygon vertices, while the other in Section 3.2 has a cost of $O(n^2)$. Some can be determined exactly (e.g. the measure in Section 3.7), while others are estimated by sampling (e.g. the measure in Section 3.5). The measures given in Sections 3.3 and 3.5 require inside-outside testing, which can easily be performed if the shape is represented as an implicit curve (or as a polygon), but would be harder to do with parametric curves. The algorithms proposed in Section 3.6 cannot be implemented in any straightforward way if lines and shapes are represented as discrete geometry by sets of pixels.

The measure in Section 3.1 is not stable with respect to changes in representation, and noise. Given a polygon with m edges, it could also be described as a degenerate n-sided polygon with $n > m$, thus with a different fraction of internal angles more than 180°. Furthermore, in the presence of noise, a single straight edge may be replaced by a series of many edges which waver in and out, half of whose internal angles are more than 180°, seriously affecting the measure.

4 Circularity

Here we take circularity to measure how much an object resembles a perfect circle. Other, more technical definitions of circularity are also used. For example in inspection, circularity is defined in terms of the difference in radii of two concentric circles bounding the object. This requires computation of a suitable center (the minimum radial separation center), such that the difference in radii between the two concentric bounding circles is minimum. Our algorithms do not address this specialised requirement.

Circularity of a shape cannot be determined from *local* properties of its boundary. This is clear from Figure 2, composed of two joined circular arcs. Each point of the boundary, apart from the intersections of the arcs, has a circular neighbourhood, but the whole shape is not a circle. Furthermore, intuitively we would not expect this shape to have a high circularity measure. Clearly, *global* tests are needed to determine circularity. Construction of

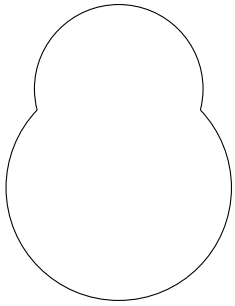

Figure 2: Not a circle.

the centroid provides the global nature of the first measure proposed below. Sample points are not restricted to a neighbourhood in the second measure.

4.1 Equidistance from Centre

The simplest definition of a circle is that all points are equidistant from the centre; this can be readily formulated as a Boolean test for a circle.

We can produce a Type 2 algorithm by slightly modifying this definition to be that all points are the same distance as the *average* distance from the centre. However, such a Type 2 algorithm will almost certainly not be useful. Most non-circular curves will cross the average distance a finite number of times as they vary from sometimes being more and sometimes less than the average distance. When compared to the infinite number of points on the curve, this will give a circularity measure of zero.

Nevertheless, a Type 3 algorithm, which takes into account how far each point is from the average radius, can still be useful. For the measure to return a value in the range 0–1, we cannot simply use $r(R) = 1 - |R - R_0|/R_0$ as the *result-of-this-decision-test*, where R is the radius of the curve at this point and R_0 is the average radius, as this quantity becomes negative for $R > 2R_0$. Instead, we can use $r(R) = 1 - |R - R_m|/R_m$ where R_m is the maximum radius of the curve. This too, has its problems, however, in that a curve which is almost a circle with a long thin spike in one place will have a low circularity measure—most points are close to the average radius, not the maximum radius. An alternative is to use $1 - |R - R_0|/R_m$.

Note, however, that it may be difficult to compute R_m robustly for certain types of object. An alternative definition for *result-of-this-decision-test* which avoids R_m is

$$r(R) = \begin{cases} 1 - (R_0 - R)/R_0 & \text{if } R \leq R_0 \\ e^{(R_0 - R)/R_0} & \text{if } R > R_0 \end{cases}$$

Our measure of circularity in either case becomes

$$\frac{\int_C r(R)\, R\, d\theta}{\int_C R\, d\theta}$$

taken around the closed curve C. If the curve has very simple form in polar coordinates with respect to its centroid, it may be possible to evaluate this measure analytically.

Haralick [9] gives a related measure for circularity of R_0/σ where σ is the variance in radius; his measure is not in the range 0–1.

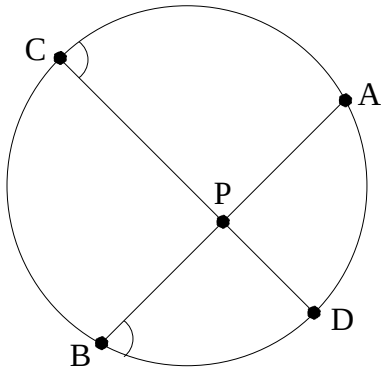

Figure 3: Chord theorem.

4.2 Chord Theorem

The chord theorem provides a necessary and sufficient test that a shape is a circle. If an arbitrary point P is placed inside a circle then all pairs of chords passing through P satisfy $PA/PD = PC/PB$; see Figure 3.

If the shape is not convex, a chord through P may intersect the boundary of the shape in more than two places. We choose here to take the two outermost intersections as being equivalent to those for a circle, but clearly other choices are possible too.

A Type 2 algorithm based on this test will again give a measure of zero for most shapes. A Type 3 algorithm can be constructed by measuring the discrepancy between the two ratios $r_1 = PA/PD$ and $r_2 = PC/PB$. We use as *result-of-this-decision-test* $\min(r_1, r_2)/\max(r_1, r_2)$ which must always be in the range 0–1, and has the value 1 as desired when $r_1 = r_2$.

This quantity could be averaged over random positions for P inside the shape, and random orientations of the two chords. However, the resulting circularity measure would depend on sampling a four-dimensional parameter space, which would require a large number of samples to ensure that a repeatable value is calculated accurately. We can reduce the size of the parameter space by only considering chords through a P fixed at the centroid of the shape. This also ensures that the lengths in the tests are not too small (except for a *few* points in shapes where the boundary passes close to its centroid), which means that individual ratios are calculated more accurately.

Likewise, we can further reduce the size of the parameter space to be sampled by constraining the chords used to be perpendicular, thus limiting the parameter space to a one-dimensional set of orientations. (In the limit that parallel chords are used, i.e. identical chords, $B = D$ and $A = C$, so we obtain $r_1 = r_2$ whatever the shape. This justifies the use of perpendicular chords as those which give most information.)

The above measure can again be expressed as an integral in polar coordinates with respect to the centroid.

The idea of parallel chords above suggests an alternative, and simpler, measure for circularity. Suppose we take single chords APB through the centroid of the shape. For a circle, $AP = PB$; for other shapes we could average the quantity $\min(PA, PB)/\max(PA, PB)$ over all orientations in a Type 3 algorithm. Unfortunately, the requirement that $AP = PB$ for all chords, while necessary, is not sufficient to define a circle, and is also satisfied by such shapes as a square.

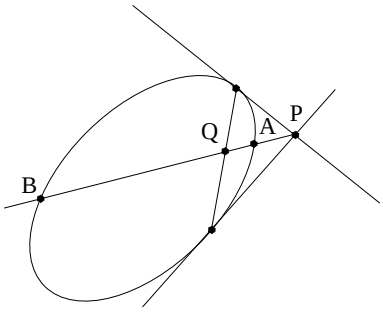

Figure 4: Test for ellipticity.

5 Ellipticity

5.1 Harmonic Range

A property of conics is that certain constructions of four points form a harmonic range (also called a harmonic ratio or harmonic section), as shown in Figure 4. An arbitrary point P is chosen such that two lines can be drawn through it to form tangents to the conic. The line through the resulting tangent points is called the polar. A third ray from P is drawn which intersects the conic at two points A and B, and the polar at Q. Two ratios of the lengths formed are equal:

$$\frac{QA}{AP} = \frac{BQ}{BP}.$$

As with the chord theorem for circles, a Type 3 algorithm can be constructed by measuring the discrepancy between the two ratios $r_1 = QA/AP$ and $r_2 = BQ/BP$ such that *result-of-this-decision-test* $= \min(r_1, r_2)/\max(r_1, r_2)$.

Algorithmically, rather than first selecting P and then searching for two tangent rays, it is more convenient and efficient to reverse the process. Thus, we select two random points on the conic, and find the point of intersection of their tangents. The third ray can then be chosen randomly, although again as we did for the chord theorem algorithm, it is more efficient and effective to choose a specific ray, in this case the one passing through the midpoint of the polar. The number of intersections of the third ray is tested; it should form exactly two intersections and if it does not, this is evidence against the shape being ellitptical.

In this Section we only consider closed curves, which therefore excludes hyperbolae and parabolae which share the harmonic range property with ellipses. However, shapes large parts of whose boundaries are hyperbolic or parabolic, need to be considered in case they misleadingly have a high measure using this approach. Informally, we can see why such shapes receive low scores from many of the individual harmonic section tests. The hyperbolic or parabolic part of the boundary may be anywhere from almost all of the shape's perimeter down to a vanishingly small part in the limiting cases. Let us consider two cases towards the extremes, when a single hyperbolic or parabolic part is either relatively small or large (see Figure 5). In the first case many of the intersection points B in Figure 4 will lie on the non-conic (straight, in the Figure) segment, and the lengths clearly will not form a harmonic ratio. In the second case most of the tangents will intersect somewhere in the region of the apex of the conic, which will cause a disproportionately large number of the intersection points B to still lie on the non-conic segment.

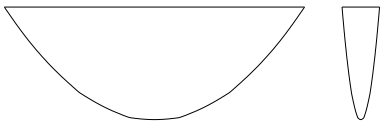

Figure 5: Testing for harmonic sections with a parabolic or hyperbolic segment.

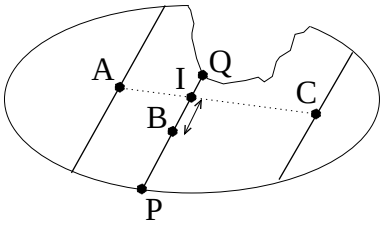

Figure 6: Test for ellipticity.

A harmonic section can arise on the boundaries of other shapes, but if all points of the shape satisfy the test, it is a conic section. This leads to the possibility that other shapes could receive a high ellipticity score. However, given the specificity of the test, this can only arise for a very restricted class of non-elliptic shapes.

The final method is outlined in Algorithm 5.

5.2 Midpoints of Parallel Chords are Linear

As a second test for ellipticity we can use the property of conics that the centres of parallel chords lie on a straight line. Again, this property is not specific to ellipses unless we restrict attention to closed shapes.

The property shows that if three parallel chords are determined for a shape, then the distance between the midpoint of one chord from the straight line through the remaining two midpoints provides a measure of error suitable for a Type 3 algorithm. (We must separately handle chords with more than two intersections for non-convex shapes.) As shown in Figure 6, the discrepancy of the midpoint B relative to the midpoints AC can be normalised to give a result in the range $[0, 1]$:

$$\textit{result-of-this-decision-test} = \begin{cases} 1 - |BI/PQ| & \text{if segment } AC \text{ intersects segment } PQ, \\ 0 & \text{otherwise.} \end{cases}$$

A disadvantage with this scheme is its complexity. Triples of lines need to be sampled and furthermore this process should be repeated over all directions, leading to a Monte-Carlo method in a high-dimensional sampling space, leading to slow performance, or low accuracy for a given amount of computation.

5.3 Measuring Ellipticity via Circularity

Rather than developing higher-order tests for measuring ellipticity, it is possible to reuse the simpler, more efficient circularity measures if the shape is normalised first. The shape should be scaled to become more circular. This could be done in various ways. For instance, if a minimum bounding rectangle is placed around the shape, then the shape can be scaled to

Algorithm 5 Type-3 algorithm for ellipticity measure based on harmonic sections

count $\leftarrow 0$
amount-true $\leftarrow 0$
for *number-of-times-desired* **do**
  $p_1 \leftarrow$ *random-boundary-point(shape)*
  $p_2 \leftarrow$ *random-boundary-point(shape)*
  $t_1 \leftarrow$ *tangent*(p_1)
  $t_2 \leftarrow$ *tangent*(p_2)
  if parallel(t_1, t_2) **then**
    skip
  else
    count $\leftarrow$ *count*+1
    if *set-of-intersections*(t_1,*boundary(shape)*) $\neq \{p_1\}$ **then**
      skip
    else if *set-of-intersections*(t_2,*boundary(shape)*) $\neq \{p_2\}$ **then**
      skip
    else
      $P \leftarrow$ *intersection*(t_1, t_2)
      $Q \leftarrow (p_1 + p_2)/2$
      $l \leftarrow$ *line*(P, Q)
      if *number-intersections(l,boundary(shape))* $\neq 2$ **then**
        skip
      else
        $\{A, B\} \leftarrow$ *intersections(l,boundary(shape))*
        $r_1 = QA/AP$
        $r_2 = BQ/BP$
        amount-true $\leftarrow$ *amount-true*$+ \min(r_1, r_2)/\max(r_1, r_2)$
      end if
    end if
  end if
end for
return *amount-true*/*count*

transform the minimum bounding rectangle to a square. Alternatively, moments can be used to find the orientation and lengths of the principal axes of the shape, and the shape scaled so that both axes are of the same length. The circularity measure can then be applied directly to provide an ellipticity measure with the caveat that by distorting the shape through scaling the ellipticity measure will also be distorted to some degree.

Such schemes, however, go beyond direct application of the principles described in Section 2.

6 Triangularity

A decision method for triangularity of a polygon is "Is every point inside the minimum bounding triangle of the polygon also inside the polygon?" The minimum bounding triangle for a polygon can be found in linear time [27]. We can thus obtain a Type 2 algorithm by simply counting the number of points inside the shape, and dividing by the number of points inside the minimum bounding triangle. Clearly, this is just the ratio of the area of the polygon to the area of the triangle.

While this is an obvious measure for triangularity, if we take this in conjunction with one of the earlier measures given for convexity, a principle emerges. Our approach leads to a whole class of shape measures of the form: ratio of the area of the shape to the area of some suitable bounding shape.

It is difficult to find measures of triangularity not based on fitting triangles (e.g. least squares, inscribed, circumscribing) to the shape. We find the same problem with quadrilaterals, or in fact any arbitrary polygon. Beyond finding instances of the shape itself such general shapes do not seem to possess sufficiently powerful properties from which to construct a measure unless further constraints are added, e.g. convexity, regularity, or rectilinearity.

7 Rectilinearity

7.1 Internal Angles Equal 90°

A rectilinear (orthogonal) polygon is a polygon with edges parallel to an orthogonal coordinate system. Thus, a rectilinear polygon has internal angles each equal to a multiple of 90°. To make a reliable Type 3 algorithm, nearly collinear segments also need to be taken into account. Thus, the absolute deviation of each internal angle a_i from the nearest multiple of 90° is measured (note that this deviation must be less than 45°), and normalised so that deviations of 0° give a value of 1, up to deviations of 45° which give 0:

$$\textit{result-of-this-decision-test} = \frac{|45 - (a_i \bmod 90)|}{45}.$$

A further problem remains; if a long sequence of internal angles is consistently slightly over or under 90° then the first and last edges from the sequence could have a large discrepancy in orientation even though the sequence's average angular error is low. One solution is to measure the difference in orientation between each polygon edge and some reference orientation (e.g. the mode edge orientation), $\theta_i - \theta_R$:

$$\textit{result-of-this-decision-test} = \frac{|45 - (|\theta_R - \theta_i| \bmod 90)|}{45}$$

7.2 Angles Between Pairs of Edges Equal 0° modulo 90°

Another characteristic of a rectilinear polygon is that the angle between *any* pair of edges (not just adjacent ones) is a multiple of 90°. Measuring the difference in angles with a Type 3 algorithm as above uses (with the same notation as above)

$$\textit{result-of-this-decision-test} = \frac{|45 - (|\theta_j - \theta_i| \bmod 90)|}{45}$$

The advantages of this approach are that it does not need a reference direction, but at the same time, it does not only make local comparisons between edges, but between distant pairs of edges too. However this comes at a higher computational complexity of $O(n^2)$.

8 Rectangularity

Rectangularity of a polygon is often computed using its minimal bounding rectangle (which again can be found in linear time [24]), although alternative methods exist [35].

A new approach would be to use either of the measures we define for rectilinearity, as rectilinearity is necessary for a shape to be a rectangle. To make a complete and sufficient test we must also check that the shape has four sides, *or* that it is convex. The former is a less useful idea, as it is quite possible for a non-four-sided shape to be intuitively close to a rectangle (in a global sense). A simple way of combining the tests for convexity and rectilinearity is to compute these measures separately, and simply take their product.

More in the spirit of this paper is to define *result-of-this-decision-test* for use in a Type 3 algorithm as *result-of-this-decision-test* $= \min(r_c, r_r)$ or *result-of-this-decision-test* $= r_c \times r_r$, where r_c is the *result-of-this-decision-test* value for convexity, and where r_r is the *result-of-this-decision-test* value for rectilinearity, where the same geometric data is used to compute each measure, e.g. internal angle. Using the average of the products, rather than the product of the averages (measures), will lead to a stricter test for rectangularity.

Clearly, this approach of combining the computations used for separate measures to give some new measure is a general principle. It was seen earlier in the guise of **if**...**then**...**else** in Algorithm 5.

9 Symmetry

Symmetry in 2D is simpler than symmetry in 3D, and here we restrict the discussion to 2D symmetry. The only symmetries possible in 2D are C_n, n-fold rotational symmetries for $n \geq 2$, and D_n for $n \geq 1$, where n-fold rotational symmetry is combined with n mirror axes of symmetry through the centre of rotation. (D_1 symmetry is a single mirror axis, C_∞ represents circular symmetry.) Various symmetry detection algorithms, and approximate symmetry detection algorithms, are discussed in [23]. One approach [21, 41] starts by constructing an ordered set of elements representing the object. For example, for a polygon, this set could consist of pairs (α_i, l_i) where α_i is the angle between the i^{th} and $i + 1^{st}$ sides of the object considered cyclically, and l_i is the length of the i^{th} side. These elements are arranged as a string L. To find a rotational symmetry, a string search is carried out for (non-trivial) occurrence of the string L in the string LL made by concatenating L with itself. This search can be done in linear time using the Knuth-Morris-Pratt algorithm [15], leading to a test for

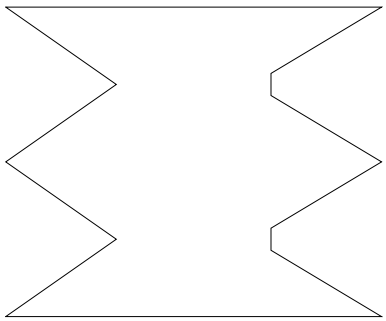

Figure 7: A nearly symmetric object.

any rotational symmetry in linear time. A simple modification of this method also allows for the detection of mirror symmetries by using strings in both forwards and reverse order [11]

We will now use these ideas to devise a measure for a specific symmetry. Suppose we wish to compute a measure of C_n symmetry for a particular n, for a polygon; the case for D_n follows in a similar way. We assume that the total number of vertices of the polygon is a multiple of n, say mn. (There are various ways of proceeding if this is not the case.) Then, taken cyclically, the elements e_i of the set described above have the property that

$$\begin{aligned} e_1 &= e_{1+m} = e_{1+2m} \ldots e_{1+(n-1)m} \\ e_2 &= e_{2+m} = e_{2+2m} \ldots e_{2+(n-1)m} \\ &\ldots \\ e_m &= e_{m+m} = e_{m+2m} \ldots e_{m+(n-1)m} \end{aligned}$$

An algorithm of Type 2 is not quite straightforward in this case, as the decisions to be made are not quite Boolean decisions, but are tests that all members of a subset of elements are equal. One way to turn these into proper Boolean decisions is to take the median of each subset, i.e. the most common value occurring in each subset, and test each member of the subset for equality with the median value. (Unfortunately, this is ill-defined when two or more element values are equally common.)

An algorithm of Type 3 can be obtained by measuring the difference between the elements of a subset from the mean value of the subset (which does not suffer from the degeneracy problem of the median). However, this is not straightforward, as each element is composed of both a length and angle. One possibility is to use as the measure

$$\max(0, \min(1 - |\alpha_i - \alpha_m|/\alpha_m, 1 - |l_i - l_m|/l_m))$$

where α_m and l_m are the mean values for the subset.

More sophisticated methods are needed when assessing symmetry in cases where multiple points ought to be matched to a single point under isometry [26], as in the polygon shown in Figure 7.

10 Containment

Containment tests whether one shape is inside another. We can naturally extend this to consider what fraction of one shape is inside another (measured in terms of areas), which can

be considered as a Type 2 algorithm which counts the number of points in the "inside" shape which are also in the "outside" shape.

Alternatively, if the boundaries of the shapes are more important than their interiors, then a measure of fractional perimeter length contained may be more appropriate than fractional area.

Clearly, measuring containment is only appropriate if the two shapes have been suitably located and oriented with respect to each other first.

11 Resemblance

Here we consider how much two shapes resemble each other. Various prior approaches to this problem include Bottleneck Matching [6], use of Hausdorff distances [1], and Size Functions [39]. For now, we assume the two shapes have been set in suitable positions and orientations for comparison. A more general concept, which allows the two shapes to be rotated and translated (and even scaled) into some best correspondence before they are compared is considered in the next Section.

The word 'congruent' already has a precise geometric meaning, which allows for rotation and translation, while 'similar' also allows scaling, so we avoid using the natural English word 'similar' for measuring the extent to which two shapes are the same. Instead we use the word *resemblance* for this property.

A simple decision test for equality of two shapes A and B is: "Is every point of A inside B *and* every point of B inside A?". If we extend the general principles given in Section 2, we may define a continuous resemblance measure by computing separately containment measures for A-inside-B and B-inside-A, and then combining them in some suitable way. Some obvious possibilities include $\min(A\text{-inside-}B, B\text{-inside-}A)$, and average($A$-inside-$B$, B-inside-A), assuming we use fractional area as the measure of containment. The former is probably more intuitive. One minus the normalised symmetric difference is another possibility, which has the advantage of leading to a metric.

12 Geometric Congruence and Similarity

Two shapes A and B are geometrically *similar* if a suitable transformation can be found which aligns them perfectly, i.e. after transformation they are the same. A simple approach to finding the transformation is: move B so that its centre of gravity is at the same position as that of A, then rotate B so that its principal axes are aligned with those of A, and then scale B so that the sum of its eigenvalues associated with the principal axes are the same as those of A (omitting the last step if we are interested in congruence rather than similarity). Having aligned the shapes, a measure for resemblance can now be used as a measure for similarity (or congruence respectively). This approach is not robust in cases where the two principal eigenvalues are almost the same, for example almost circular shapes, or other shapes with symmetry such as regular polygons.

A more sophisticated (and computationally expensive) approach would be to perform an optimisation over the space of transformations to find the transformation giving the maximum resemblance between the two shapes.

Veltkamp [38] lists various desirable properties that a resemblance measure should have: it should be a metric (measuring a *distance* between two shapes: see e.g. [2]), it should be

continuous in various ways, and it should be invariant under some class of transformations so that $d(T(A), T(B)) = d(A, B)$. For example, measures based on the definitions for resemblance above satisfy invariance under affine transformation, since the ratio of areas is invariant under affine transformation.

Another approach to congruence of polygons is based on *turning function distance*. The latter is defined for a polygon in a way which is inherently independent of position and orientation, thus avoiding the need to register the shapes before comparison. Using the description for polygons given in Section 9, based on side lengths and angles between adjacent sides, two polygons A and B are equal if the strings L_A and L_B describing them are equal (assuming each string starts at the same corresponding side of the polygon). We can convert this to a (simplistic) measure of shape resemblance using a Type 2 Algorithm, by counting in how many places the two strings match, and dividing by the length of the (longest) string. This measure could be maximised over all choices of starting side for the string if the strings are not given as starting in the same positions. Of course, such a simple strategy will perform badly if, for example one of the polygons has extra or fewer sides compared to the other. In practice, we could make this measure more meaningful by computing the *edit distance* [19] between the two strings, i.e. the minimum set of editing changes needed to convert one string into the other, and dividing by the total number of elements of the longest string, but such an approach is no longer directly in the spirit of this paper.

13 2.5D-ness

Here we define a 2.5D object to be one which, when placed in a given coordinate system with z pointing up, has only horizontal faces and vertical faces (i.e. ones with normals in a horizontal plane), and furthermore, has no overhangs (i.e. no downward pointing faces other than any base faces which it sits upon when placed upon a horizontal table). Other definitions of 2.5D objects exist: for example, ignoring vertical faces and the base face, for each x, y position in the object, there is a single z value denoting the upper surface of the object.

2.5D objects are often simpler to manufacture than more general 3D objects: they can be machined with a simple 3-axis milling machine using a single setup, they can be manufactured by layer-based rapid prototyping machines without using supports, and they can easily be removed from a mould. Manufacturing thus offers several potential applications of a measure of *2.5D-ness*: for example how much of the object could be manufactured in a single simple setup, and how much would need processing in other setups, perhaps with a more sophisticated machine. Of course, such a measure would be crude, and would not take into account such important issues as set-up time. However, combined with other measures and further analysis, it has the potential to be informative.

Starting with the definition that an object is 2.5D if it has no non-vertical faces, no non-horizontal faces, and no overhanging faces, a simple Type 2 algorithm for computing 2.5D-ness just considers whether each point of the surface of the object belongs to a face which is acceptable, resulting in the ratio: areas-of-acceptable-surfaces to the total surface area of the object. Other variants are possible: we could project each face of the object onto the base plane, and take their 2D convex hull. We then fire a line upwards from each point inside the hull, counting 1 for each line which only intersects a single surface of an appropriate type, and 0 for other cases—this places more emphasis on projected areas than actual areas.

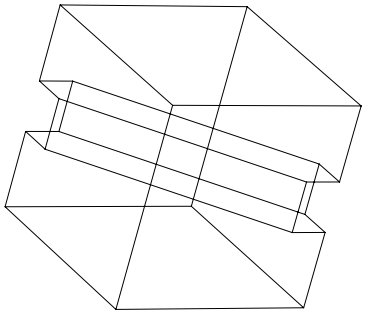

Figure 8: A through hole not ending in hole loops.

Note that the above measures includes an assumed vertical direction. For certain applications, more sophisticated measures which consider all possible vertical directions might be more useful.

14 Imperforateness

A three-dimensional object is imperforate if it has no through holes, i.e. has genus zero. By considering objects with through holes of various relative sizes, we can imagine a measure of how "big" the hole is. For example, a sphere with a short pin hole through it is very close to being imperforate, while a short section of tube with paper-thin walls is nearly all hole, and very far from being imperforate.

For boundary representation solid models, the Euler-Poincaré formula [12, 16] can be used to compute the genus:

$$V - E + F - L = 2(M - G),$$

where V is the number of vertices, E edges, F faces, L hole loops, M bodies, and G the genus. Hole loops may bound the base of a boss on a face, or the rim of a hole in a face, or they may be at the end of a through hole terminating in a face. Furthermore, the "end" of a through hole may not just lie in one face, but may run across several faces (see Figure 8). The topic of feature recognition in B-rep models has a large literature [8, 28, 30], and an item of frequent interest is detecting through holes, although many papers only consider special cases. We use an approach below due to Kim [14], based on an earlier idea of Woo's [42], to devise an algorithm for computing an *imperforateness* measure for polyhedra.

An object has genus zero if its through holes have zero "volume" (volume to be defined precisely later). If an object has through holes, holes with a small volume should have a smaller effect on the imperforateness measure than larger holes. (It is possible to imagine a thin hole which worms around many times inside the object and eats away much of its interior before emerging elsewhere: this would be a large volume hole. Here, the volume is considered to be more important than the size of the opening at the ends of the hole, but clearly other measures of the relative importance of different holes may have different priorities.)

Take a cube with several independent through holes and pockets, and subtract it from its convex hull. Each pocket and through hole results in a separate body. Let us colour those faces of each remaining body which were shared with the convex hull red, and other faces blue. *Pockets* have just one contiguous cluster of red faces, whereas *through holes* have two

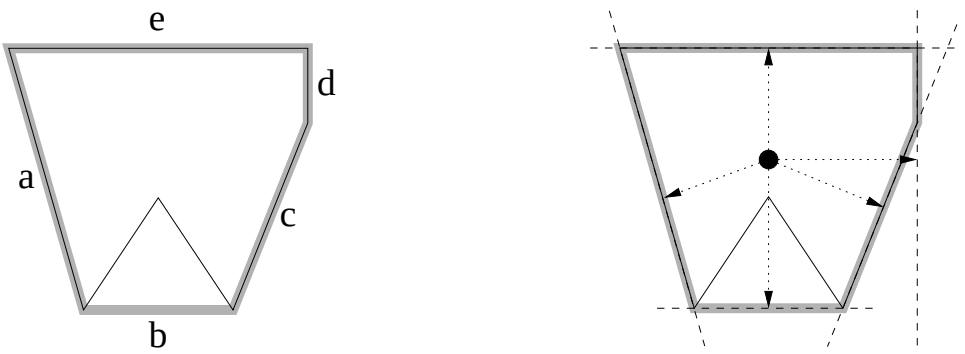

Figure 9: Side-on view of 3D object with uniform cross section. Its convex hull is shown in gray.

or more separate clusters of red faces. We define the "volume" of each through hole to be the volume of the appropriate body remaining after subtraction.

In a more general case, an alternating hierarchy of positive and negative volumes subtracted from their respective convex hulls is needed; partitioning of various volumes produced may also be needed to ensure that this hierarchy terminates [14]. From this hierarchy it is possible to extract *closure faces* which cover the openings of through holes [14], and so the volumes associated with the through holes can be computed as before.

This idea leads to a Type 2 measure for imperforateness, which iterates over all points inside the convex hull of the original object, and counts the fraction which are shared with through holes, resulting in the ratio of the "volumes" of the through holes to the volume of the convex hull.

Clearly, this measure has limitations—for example, two pockets entering the object from opposite sides could nearly meet, yet would not affect the measure. A second problem is that no account is taken of any internal voids which nearly come to the surface. Overall, then, it might be objected that the intuitive meaning of this measure is questionable. Nevertheless, such a measure might have uses in shape classification, for example.

15 Stability

In design engineering and manufacturing, a useful aspect of an object's shape is its stability, and is of particular importance during the assembly process. We can define a convex polyhedral object to be *unconditionally stable* if, in each case, it does not topple over when it is put down with one of its faces in contact with a horizontal ground plane. For instance, the outer shape in Figure 9(a) is not unconditionally stable since placing it on face d will result in it falling over onto face c. The test for each face can be computed by forming the perpendicular projection from the polyhedron's centre of mass to the plane containing the face (see Figure 9(b)). The object is stable when placed on any face only if the projected centre of mass lies within the face. Iterating over all faces makes up a Boolean test for stability.

It is simple to extend this stability test to non-convex polyhedra by performing tests on the faces of the *convex hull* of the polyhedron. Again considering Figure 9(a), it is easy to see that the inner polyhedron is stable in the orientation shown, as the centre of mass lies above

face b of the convex hull—the object is not resting on a face, but it *is* in a stable pose.

To produce a continuous measure of stability, we can use the above ideas as a basis for Type 2 and 3 Algorithms. The former is simple—we iterate over all convex hull faces and count the number of stable faces, dividing by the total number of faces.

This measure is crude, however, and a better approach, using a Type 3 Algorithm, takes account of the probability that the object would be placed on any particular face when put in a random orientation—we must allow for the size of each face. (Wiegley *et al.* [40] use this idea to assess the probability of the object finally lying on any given face when put down in a random orientation.) For face F_i this may be computed by projecting from the polyhedron's centre of mass onto a sphere, and determining its projected area A_i. We then use

$$\textit{result-of-this-decision-test} = \begin{cases} 0 & \text{if this face is unstable} \\ A_i & \text{otherwise} \end{cases}$$

To get the correct normalisation, we must use a sphere whose total surface area is n, the number of faces in the convex hull of the object.

16 Conclusions

We have given a meta-method for deriving shape measures from definitions of Boolean shape properties. We have used it to construct a variety of example shape measures, and classes of shape measures, some of which are well-known, and others apparently new. It is possible to produce many different measures for a single shape property, as our examples show. We have not compared such alternatives, nor have we evaluated the usefulness of our measures in any particular application. Our approach is intended to be generic, and users must determine for themselves the most appropriate measures for a given problem. The measures which may be constructed are not limited to our examples, and we have given a principled way in which an application user may derive new ones.

Acknowledgments

We would like to thank the members of the Geometric Modelling Society for various helpful discussions, including suggesting some shape properties for measurement, and pointers to references.

References

[1] E. M. Arkin, L. P. Chew, D.l P. Huttenlocher, K. Kedem, J. S. B. Mitchell "An Efficiently Computable Metric for Comparing Polygonal Shapes" IEEE Transactions on Pattern Analysis and Machine Intelligence 13(3), 209–216, 1991.

[2] L. Boxer "Computing Deviations from Convexity in Polygons" Pattern Recognition Letters 14, 163-167, 1993.

[3] L. G. Brown "A Survey of Image Registration Techniques" ACM Computing Surveys 24, 325–376, 1992.

[4] L. da Fontoura Costa, R. Marcondes Cesar "Shape Analysis and Classification: Theory and Practice" CRC Press, 2001.

[5] T. J. G. Davies, R. R. Martin, A. Bowyer "Computing Volume Properties using Low-Discrepancy Sequences" Computing [Suppl] 14, 55–72, 2001.

[6] A. Efrat, A. Itai, M. J. Katz "Geometry Helps in Bottleneck Matching and Related Problems" Algorithmica 31(1), 1–28, 2001.

[7] V. N. Gudivada, V. V. Raghavan "Content-based Image Retrieval Systems" IEEE Computer 28 (9), 18–22, 1995.

[8] J. H. Han, M. Pratt, W. C. Regli "Manufacturing Feature Recognition from Solid Models: A Status Report" IEEE Trans. Robotics and Automation, 16 (6), 782–796, 2000.

[9] R. M. Haralick "A Measure for Circularity of Digital Figures" IEEE Trans. on Systems, Man and Cybernetics 4, 394–396, 1974.

[10] A. Held, K. Abe "On the Decomposition of Binary Shapes into Meaningful Parts" Pattern Recognition 27 (5), 637–647, 1994.

[11] P. T. Highnam "Optimal Algorithms for Finding the Symmetries of a Planar Point Set" Information Processing Letters 22, 219–222, 1986.

[12] C. Hoffmann "Geometric and Solid Modeling" Morgan Kaufmann, 1989.

[13] M. K. Hu "Visual Pattern Recognition by Moment Invariants" IRE Trans. Information Theory IT-8, 179–187, 1962.

[14] Y. S. Kim "Recognition of Form Features using Convex Decomposition" Computer Aided Design 24 (9), 461–476, 1992.

[15] D. Knuth, J. Morris, V. Pratt "Fast Pattern Matching in Strings" SIAM J. Computing 6, 322–350, 1977.

[16] I. Lakatos "Proofs and Refutations" Cambridge University Press, 1976.

[17] V. F. Leavers "Shape Detection in Computer Vision Using the Hough Transform" Springer-Verlag, London, 1992.

[18] D. T. Lee, F. P. Preparata "An Optimal Algorithm for Finding the Kernel of a Polygon" JACM 26, 415–421, 1979.

[19] V. Levenshtein "Binary Codes Capable of Correcting Deletions, Insertions and Reversals" Soviet Physics—Doklady 10 (10), 707-710, 1966.

[20] X. Li, W. Wang, R. R. Martin, A. Bowyer "Using Low-discrepancy Sequences and the Crofton Formula to Compute Surface Areas of Geometric Models" To appear, CAD, 2003.

[21] G. Manacher "An Application of Pattern Matching to a Problem in Geometric Complexity" Information Processing Letters 2, 6–7, 1976.

[22] R. R. Martin "Fuzzy-set-theoretic Solid Modelling" Proc. CSG'94, Ed. J. R. Woodwark, 73–98, Information Geometers, 1994.

[23] R. R. Martin, D. Dutta "Tools for Asymmetry Rectification in Shape Design" J. Systems Engineering 6, 98–112, 1996.

[24] R. R. Martin, P. C. Stephenson "Putting Objects into Boxes" Computer Aided Design 20, 506–514, 1998.

[25] K. Mehlhorn, T. Shermer, C. Yap "A Complete Roundness Classification Procedure" Proc. 13th Symp. Computational Geometry, 129–138, 1997.

[26] B. I. Mills, F. C. Langbein, A. D. Marshall, R. R. Martin "Approximate Symmetry Detection for Reverse Engineering" Proc. Sixth ACM Symposium on Solid Modelling and Applications, Eds. D. C. Anderson, K. Lee, 241–248, ACM Press, 2001.

[27] J. O'Rourke, A. Aggarwal, S. Maddila, M. Baldwin "An Optimal Algorithm for Finding Minimal Enclosing Triangles" J. Algorithms 7, 258–269, 1986.

[28] O. Owodunni, S. Hinduja "Evaluation of Existing and New Feature Recognition Algorithms Part 1: Theory and Implementation" Proc. IMechE Part B: J. Engineering Manufacture 216 (6), 839–852, 2002.

[29] H. K. Pao, D. Geiger, N. Rubin "Measuring Convexity for Figure/Ground Separation" Proc 7th Int. Conf. Computer Vision (2), 948–955, 1999.

[30] S. Parry-Barwick, A. Bowyer "Is the Features Interface Ready?" In: Directions in Geometric Computing, Ed. R. R. Martin, Information Geometers, 1993.

[31] M. Peura, J. Iivarinen "Efficiency of Simple Shape Descriptors" Advances in Visual Form Analysis, Eds. C. Arcelli, L.P. Cordella, G. Sanniti di Baja, 443–441, World Scientific, 1997.

[32] R. J. Prokop, A. P. Reeves "A Survey of Moment-based Techniques for Unoccluded Object Representation and Recognition" CVGIP: Graphics Models and Image Processing 54 (5), 438–460, 1992.

[33] A. G. Requicha "Toward a Theory of Geometric Tolerancing" Int. J. Robotics Research 2 (4), 45–60, 1983.

[34] F. Reuleaux "The Kinematics of Machinery: Outlines of a Theory of Machines" Macmillan, 1876.

[35] P. L. Rosin, "Measuring Rectangularity", Machine Vision and Applications 11, 191–196, 1999.

[36] P. L. Rosin, "Measuring Shape: Ellipticity, Rectangularity, and Triangularity" To appear, Machine Vision and Applications.

[37] H. I. Stern "Polygonal Entropy: a Convexity Measure" Pattern Recognition Letters 10, 229–235, 1989.

[38] R. C. Veltkamp "Shape Matching: Similarity Measures and Algorithms" Proc. Shape Modelling International 2001, 188–197, 2001.

[39] A. Verri, C. Uras, P. Frosini, M. Ferri "On the Use of Size Functions for Shape Analysis, Biological Cybernetics 70, 99–107, 1993.

[40] J. Wiegley, A. Rao, K. Goldberg "Computing a Statistical Distribution of Stable Poses for a Polyhedron" 30th Annual Allerton Conf. Communications, Control, and Computing, 1992.

[41] J. D. Wolter, A. C. Woo, R. A. Volz "Optimal Algorithms for Symmetry Detection in Two and Three Dimensions" The Visual Computer 1, 37–48, 1985.

[42] T. C. Woo "Feature Extraction by Volume Decomposition", Proc. Conf. CADCAM Technology in Mechanical Engineering, 76–94,1982.

[43] J. Zunic and P.L. Rosin "A Rectilinearity Measurement for Polygons", Proc. European Conference on Computer Vision, 746–758, 2002.

[44] J. Zunic and P.L. Rosin "A Convexity Measurement for Polygons", Proc. British Machine Vision Conference, 173–182, 2002