Using a Mixed Wave/Diffusion Process to Elicit the Symmetry Set

G. L. Scott S. C. Turner* A. Zisserman*

Robotics Research Group
Department of Engineering Science
University of Oxford

We describe a form of damped wave process which appears to have considerable promise in early vision analysis. We demonstrate in particular its use in eliciting a version of the Symmetry Set – the locus of centres of circles bitangent to a curve. The wave process is a method of implementing the "Symmetry Transform" of an image, which we define and discuss briefly. We describe an algorithm to implement this process, and show that sufficient information may be extracted from the algorithm to permit reconstruction of the shape. The algorithm lends itself, not merely to parallel computation, but to analogue implementations which could run to completion in milliseconds.

THE SYMMETRY SET

The symmetry set of a plane curve is defined as the locus of centres of all circles tangent to the curve at two or more distinct points.[1] The symmetry set is a superset of the "symmetric axis transform" (SAT) which is the locus of centres of those bitangent circles contained entirely within the curve ("maximal" circles).

The symmetry set should not be confused with the *evolute* which is the locus of centres of curvature. The evolute and the symmetry set intersect only at points corresponding to extrema of curvature of the curve.

Figure 1 illustrates the three sets for an ellipse. The dotted line ACBD is the evolute, the vertical axis AB constitutes the SAT. The symmetry set contains AB but *also* the horizontal axis CD. A little experimentation will reveal that bitangent circles centred anywhere on CD cut the ellipse (viz. they are not contained by it and are thus not constitutive of the SAT).

Two other "symmetry sets" – Brady's "smoothed local symmetries" (SLS)[2] and Leyton's "process inferring symmetry axis" (PISA)[3] are closely related to the Giblin symmetry set – and the ability to derive one generally implies the ability to derive the others. Figure 2 shows a pair of edgels A and B with a bitangent circle fitted. The "mirror line" CD is the bisector of the angle ADB formed by extending the edgels. The symmetry set point is of course C, the centre of the circle. The SLS point is E, the intersection of the mirror line and the chord connecting the edgels. The PISA point is the point F – the point *closer* to D at which the circle cuts the mirror line.

*S. C. Turner and A. Zisserman acknowledge the support of the Science and Engineering Research Council.

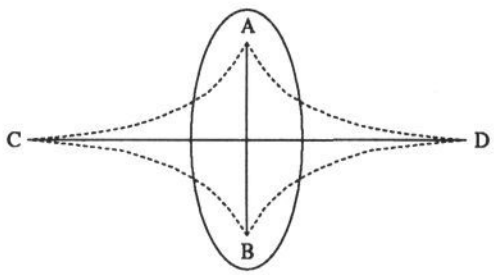

Figure 1: Example of the symmetry axes for an ellipse.

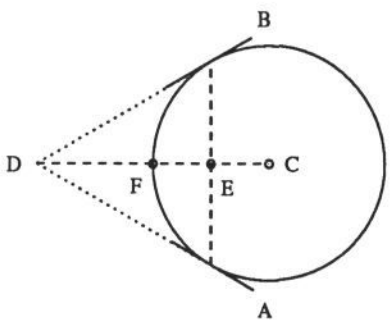

Figure 2: Illustration of the Symmetry Set, SLS and PISA points for a curve.

Uses Of Symmetries

The symmetry set and related structures have already found wide application in computer vision and robotics. Symmetry axes "compress" information and can often provide the basis of some convenient description. This tends to be particularly true for simpler biological shapes. Symmetry axes can also be viewed as a means of "skeletising" a shape (e.g. a solid letterform) prior to forming a symbolic description of it. The Symmetry Set also serves as a useful intermediate step in curve segmentation – since there is an intimate relationship between end-points of the set and points of high curvature. In this role it relates readily to codon theory[4]. It may also assist in *grouping* edgels or line segments together, since two arbitrarily related shapes will tend to have disjoint symmetry sets. Lines of symmetry have been used in robot path planning by Canny[5].

The Importance of the full Symmetry Set versus the SAT

The definitions given above are "purely" geometrical and imply continuous curves of infinitesimal thickness, differentiable save at singularities. For the purposes of computer vision these definitions need to be transposed to the earthy domain

AVC 1988 doi:10.5244/C.2.34

of the digital – a non-trivial task.

The problem of mapping differential geometry to computation is well illustrated by Blum's "brushfire" algorithm[6] which is aimed at eliciting the SAT. In the the most straightforward version of this each edge point is "lit" simultaneously and a "fire" iteratively propagated throughout the surrounding space. Where two "fire fronts" meet the fire extinguishes (the same pixel cannot "burn" twice) and an SAT point is declared. Where the data is noise-free and forms a good approximation to a single well-behaved curve, this algorithm produces a useful output (provided suitable thresholds to determine "extinction" are selected). The time at which extinction occurs at a particular point provides a measure of the *radius* of the associated maximal circle and the original outline can be reconstructed by "reverse time" operation of the brushfire. (Geometrically this corresponds to reconstructing the outline as an envelope of circles).

But in practice, on real images or on images where perceptually distinct shapes are superimposed, the brushfire technique can present severe problems. These stem from the fact that it is "all-or-nothing" (each pixel either burns or it doesn't) and "one-shot" (each pixel burns only once). Figure 3 shows the brushfire axis of a "smile". Figure 4 shows the axis which results when a single false "edgel" is contained within the shape! The original axis has been severely damaged and been replaced by a dominant "ring" which results from the meeting of fire emanating from the noise-point with fire moving inwards from the boundary.

Figure 3: Brushfire axis of a smile.

Figure 4: Brushfire axis of a slightly corrupted smile.

A similar problem is found when multiple shapes are superimposed on one image. The SAT of the combined image gives us a set of symmetry axes due to the interactions between the shapes, which is not obviously related to the individual symmetries of any of the shapes in the image. The symmetry set, however, gives us the symmetries due to the interaction between the shapes (of which the SAT given is a subset) *plus* the individual symmetries of the original shapes, superimposed.

It avails little to complain that noisy and superimposed cases violate conditions which are explicit or implicit in the "pure geometry" definition of the SAT. It is surely not desirable that a symmetry finding process should need to be prefaced by the application of an process so "intelligent" that it removes *all* noise and performs perfect shape segmentation.

A process which yields the full symmetry set – while it may tend to produce too much information on occasion – has the merit that it does not destroy the "true" symmetries under the influence of noise or interaction between different shapes.

CRITERIA FOR A COMPUTATIONAL APPROACH TO SYMMETRY

Apart from the desirability of "superposition" which the brushfire technique lacks there are other criteria which we might want to see met by a computationally robust symmetry finder. By analogy with edge detection (which seeks to extract the "edge set") we might demand:

1. *Quantitative* results – weak responses to "weak" symmetries such as those created by noise, strong responses to perceptually relevant symmetries.
2. An optimal tradeoff between localisation and sensitivity to noise (related to but not identical to (1)).

A third criterion is "nice scaling behaviour". A pixel which is an edge-point at one scale may not be so at another scale. Likewise the symmetry set of a shape at one scale may be quite different from the symmetry set of a "smoothed" (large scale) version of the shape[7]. In contrast to edge detection, however, symmetry points contain their own intrinsic measure of scale – the radius of the associated bi-tangent circle. We will use this to good effect to produce a "self-scaling" procedure.

A Convolution Approach

A fairly obvious approach to digital symmetry detection is to convolve the edge map (which we may regard as a binary image with value 1 in an edge pixel, 0 elsewhere) with a set of "annulus" masks of differing radii. By annulus mask is meant a suitable digital approximation to one which has value 1 where $R - \delta R/2 < r < R + \delta R/2$ and 0 elsewhere (see Figure 5).

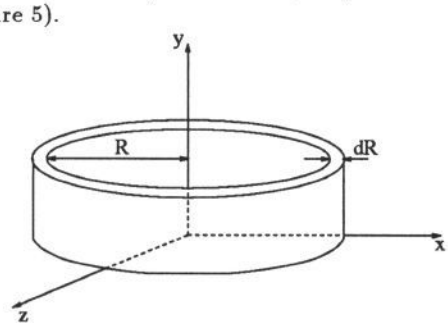

Figure 5: Annulus mask of radius R and thickness δR.

Clearly convolution with such a mask will tend to give a strong response where there are many edgels concentrated within the annulus, and the response will be greater where more edge pixels are within the mask. Note that the response, as we vary the size and position of the masks, will behave differently depending on whether the masks are "unitangent" (tangent to only one section of the curve) or "bitangent" (tangent to two or more separate sections of the curve).

In the unitangent case, the response will be high over an area corresponding to a "blurred" version of the evolute, whereas in the bitangent case the response will be sharply peaked at the symmetry point.

This approach may be analysed by use of the "Radial Distribution Function" for a curve $f(\mathbf{x})$ $(\mathbf{x} = (x, y))$, which gives a measure of the radial organisation about a point:

$$\mathcal{R}(\mathbf{x}, r) = \int_0^{2\pi} f(x + r\cos\theta, y + r\sin\theta)\, r d\theta \qquad (1)$$

Figure 6 shows two arc segments A and B, together with their centre C and an off-centre point D. The R.D.F. for C should show a marked peak for radius $r = 1$, whereas the R.D.F. for point D should have a more diffuse response – D is not bitangent to the curve segments A and B. Figure 7 shows this to be the case.

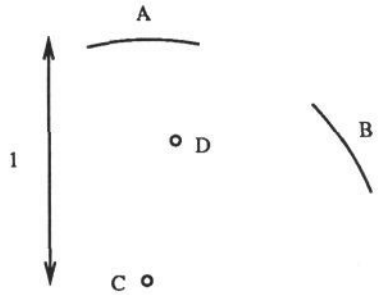

Figure 6: Two arc segments (A,B) with their centre of curvature C and an off-centre point D.

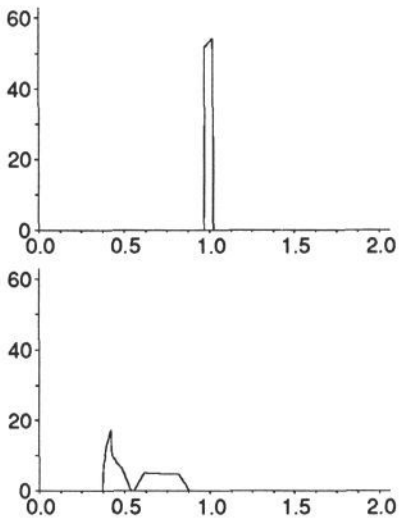

Figure 7: The Radial Distribution Function for points C (upper) and D above.

We may refine our annulus masks in two obvious ways. First we convert the cross-section from a thin "top hat" to a Gaussian-type for noise suppression purposes. Second we "scale" our masks so that the width of the cross-section is proportional to the radius of the mask ($\delta R = aR$). This implies that larger radius masks are more tolerant of "jaggies" and "scatter" along the edges, thereby giving us the "nice scaling behaviour" which we considered would be desirable.

It turns out that we can achieve both refinements simultaneously by applying a diffusion (or "blurring") process in step with the convolution – masks of greater radius are subjected to more blurring. This observation will be returned to in the next section.

THE WAVE/DIFFUSION PROCESS

Blum realised that there is a close relationship between the geometrical definition of the symmetry set and the theory of wave propagation. His "brushfire" arose as an attempt to model some aspects of a wave process.

If we attempt to propagate waves from the edge points by discrete element application of the wave equation, we encounter severe quantisation error. Figure 8 shows a snapshot of the displacement function after 70 iterations of a discrete element simulation of a membrane wave process, initiated with a single displacement "delta" at the centre.

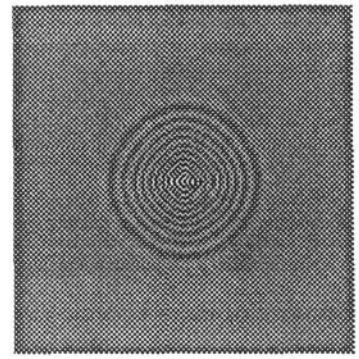

Figure 8: Snapshot of waves emanating from a single edge point.

Three things are worthy of note about this: the function becomes highly anisotropic and is of much higher spatial frequency the closer we move towards the centre. Additionally, much of the energy of the waves is concentrated around the centre. All these are indicative of numerical instability.

Now *theoretically*, none of these features should be present for the simple membrane or shallow-water wave process represented by the classical wave equation. The discrepancy arises entirely from discretisation error which particularly afflicts the higher frequency modes of vibration of the medium. In effect a discrete element simulation of a simple wave process cannot model the higher frequency wave terms – which leads to the highly distorted situation shown above.

Applying a *diffusion* process in step with the wave process clears up the numerical instability of the pure wave process seen above, simply by removing high-frequency displacement. This both removes the anisotropic area in the middle of the figure, and progressively damps out the higher frequencies of the waves – this gives us a single wave "front" (a solitary travelling wave, with no extra perturbations arriving later) which approximates to our "annular" convolution mask. Figures 9 and 10 show the effect of applying diffusion to the discrete element wave process used for Figure 8. Note that the image grey-levels are normalised to lie between 0 and 255 – the enhancement of the wave is *relative*, not absolute. The cross-section of the wave displacement is shown in Figure 11.

A further effect of damping the wave process (one which can not easily be seen from the figures) is that, as the wave moves outwards, it also becomes more diffuse. This means that the wave cross-section becomes broader the further the wave travels – which gives us the scaling behaviour mentioned above, such that distant symmetry axes are more tolerant of "scatter" in the edges than closer axes.

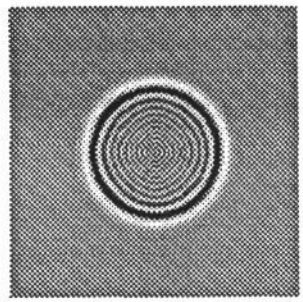

Figure 9: Snapshot of slightly damped wave process.

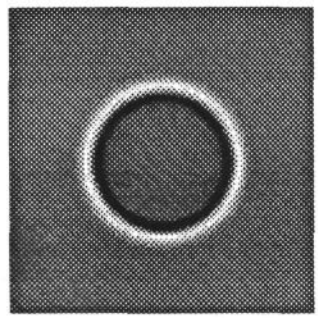

Figure 10: Snapshot of more heavily damped wave process.

Because both the wave and diffusion processes are *linear* the principle of *superposition* applies. The displacement function which results after time T from initiating a number of waves at time 0 is simply the sum of the displacement functions for each wave individually after time T. Furthermore (thanks to rotational symmetry) the total displacement function after time T is the same thing as the *convolution* of the displacement function generated after time T by a single delta at the origin with the *starting* function (a delta everywhere a wave is required to start). In short: provided we accept an annulus cross-section such as that shown in Figure 11, our wave/diffusion process may be regarded as performing the desired convolution for us!

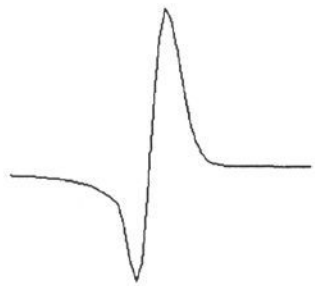

Figure 11: Cross-section of wave/diffusion displacement – the wave is travelling from left to right.

Implementation of the Wave/Diffusion Process

Our wave/diffusion algorithm may be regarded as finding two things simultaneously. It uses the wave/diffusion principle outlined above to propagate wave fronts from the edges of a shape, and gives us two separate outputs. One of these is the accumulation of "energy" (displacement squared) over time of each pixel (this is the Wave Transform, defined below); the other is to note the *maximum* energy to arrive at each point, together with its associated iteration number (which provides a measure of time of arrival). The latter of these outputs gives us, effectively, the radius of the most significant circles making up the symmetry axis – from this, we may reconstruct the shape. This aspect is discussed further below – we shall merely note its existence for now.

Formally the Wave Transform of the signal is given by

$$\mathcal{W}(\mathbf{x}) = \int_0^\infty \phi^2(\mathbf{x}, t)\, dt \tag{2}$$

where $\phi(\mathbf{x}, t)$ obeys the wave equation

$$\nabla^2 \phi = \frac{1}{c^2} \frac{\partial^2 \phi}{\partial t^2} \tag{3}$$

and is also subject to the diffusion equation

$$\nabla^2 \phi = \frac{1}{\kappa^2} \frac{\partial \phi}{\partial t} \tag{4}$$

with initial conditions

$$\phi(\mathbf{x}, 0) = f(\mathbf{x}) \qquad \left.\frac{\partial \phi}{\partial t}\right|_{t=0} = 0 \tag{5}$$

where $f(\mathbf{x})$ is the 2D binary edge image.

The algorithm itself is as follows:

1. Set up starting displacement function (1 for an edge point, 0 elsewhere). Displacement velocity is taken to be zero everywhere initially.
2. Apply *wave* process for 1 iteration. This involves (for each point):
   (a) Computing the Laplacian of the displacement function at time n.
   (b) Multiplying by c^2 (c is the wave constant) to obtain displacement acceleration.
   (c) Adding acceleration to current velocity (time n) to obtain a new displacement velocity (time $n + \frac{1}{2}$).
   (d) Adding the new velocity to the displacement to obtain the the new displacement at time $n + \frac{1}{2}$. This is the standard discrete element approximation for a wave process obeying the wave equation.
3. Apply 1 iteration of *diffusion* to *both* the displacement and the displacement *velocity* functions (using the same diffusion constant). This involves:
   (a) Computing the Laplacian of the displacement function at time $n + \frac{1}{2}$.
   (b) Multiplying by κ^2 (κ is the diffusion constant) to obtain a displacement increment.
   (c) Adding this into the displacement function to obtain displacement at time $n + 1$.
   (d) Carrying out the three steps above for the *velocity* function.
4. Add current displacement^2 (energy) for each pixel into accumulated energy store.
5. Update and record energy maxima (with associated iteration number i.e. radius).
6. Return to step 2 unless some preselected number of iterations has been executed.
7. Output accumulated energy store.

8. Output record of displacement maxima and associated iteration number.

Some details of the actual implementation, as used to generate the examples below, are not part of the basic process and so are discussed separately.

There is one modification to this basic process, demanded by the need for satisfying pictures! In order to obtain output imagery in which high-radius symmetry points are not lost when compared to low-radius points, it is necessary to take some account of the decay in wave height due to the spread caused by diffusion. For this reason we "boost" late-occurring displacement maxima relative to those that occur earlier by some amount. Note that for simplicity, we apply this boost to the stores of energy and maximum energy rather than to the process itself – although the two are theoretically equivalent. A suitable formula for the amount of boost applied – which we have used on the images in this paper – is:

$$\phi_{max,recorded} = \phi_{max} \frac{\sum \phi_{n=1}}{\sum \phi_{n=N}}$$

where ϕ_n is the displacement after n iterations and the summations are performed over the entire image. Other weighting functions may be more suitable for particular applications – in particular it might be considered desirable to keep the wavefronts moving off a straight line of constant height, so that all interference shows up equally. Our weighting function was selected because it appears to provide suitable pictures.

The laplacian mask used to compute the laplacians was:

1	2	1
2	–12	2
1	2	1

Our convolution "wraps round" – as the leftmost column of the mask extends off the leftmost edge of the image, it reappears on the rightmost column of the image etc. This is to avoid the waves reflecting off the edges – they wrap round toroidally instead. This is merely a matter of personal preference. The values of c^2 and κ^2 (the wave and diffusion constants) used were $c^2 = 0.05$ and $\kappa^2 = 0.01$.

It should be noted that a faster version of the algorithm may be realised by only applying diffusion to the displacement function – this is more difficult to analyse theoretically, since physical diffusion damps both displacement *and* velocity (by damping the displacement). We mention it here because it offers a considerable improvement in speed over the basic algorithm and seems to work quite well – the results given by the "fast" algorithm are of broadly comparable quality to those from the ordinary version. Since we only apply diffusion to the displacement, we need only compute the laplacian of the displacement $\nabla^2\phi$ once. We then use this to give us the new velocity and displacement functions: $\frac{\partial\phi}{\partial t}$ increases by $c^2\nabla^2\phi$, and ϕ increases by $\frac{\partial\phi}{\partial t} + \kappa^2\nabla^2\phi$. This is equivalent to running the wave part and the diffusion part (on the displacement only) at the same instant, rather than sequentially as above. Accumulation of energy and noting of energy maxima/times of arrival are performed as above. We note in passing that the "fast" version of the algorithm requires more damping (i.e. a larger value of κ) than the ordinary version to remove the numerical instabilities (since we are only applying the diffusion equation once in each iteration).

EXAMPLES OF THE PROCESS

The examples shown below were generated using the basic wave/diffusion algorithm as stated above, with the numbers of iterations for which the examples were run indicated in the figure captions. The first four images are examples of the two output types (accumulated energy and maximum energy/time of arrival) for a smile. Note that the edges of the original shape show up strongly on the maximum energy image – this is because the edges have high energy at the start (the starting function has all edge points at value 1, with 0 elsewhere) and this energy shows up in the store of maxima. The edges may be removed by simply ignoring the results of the first few iterations for the purposes of energy maximum storage – we will return to this point later. Note also that the time of arrival image is *not* normalised to lie between grey-levels of 0 and 255, and hence appears rather dark! The remainder of the images shown are the accumulated energy output of the algorithm for the shapes shown, subjected to non-maximal suppression to extract the symmetry axes from the surrounding background noise.

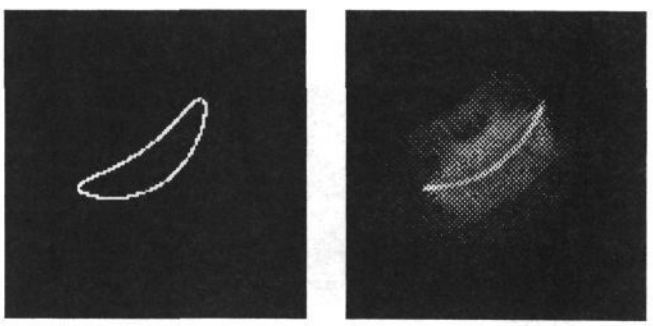

Figure 12: Smile (left) and accumulated energy output after 70 iterations.

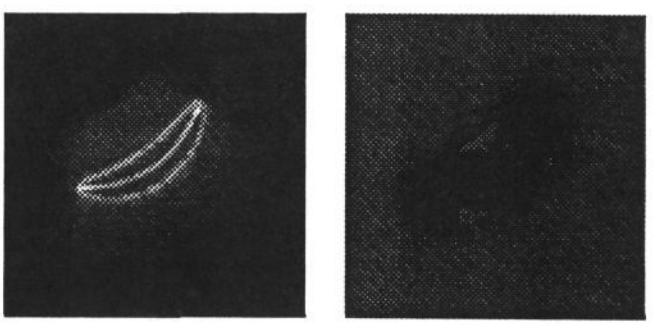

Figure 13: Maximum energy output (left) and time of arrival data for smile.

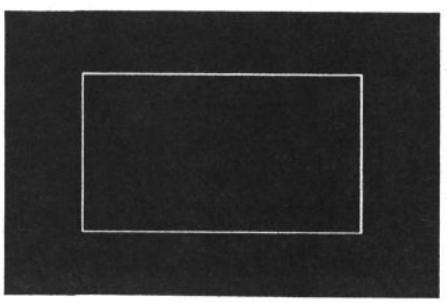

Figure 14: Rectangle image.

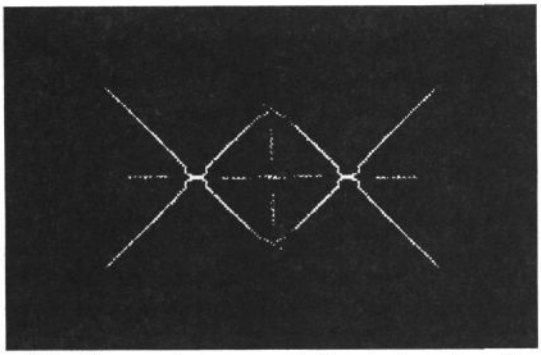

Figure 15: Symmetry axes of the rectangle after 200 iterations.

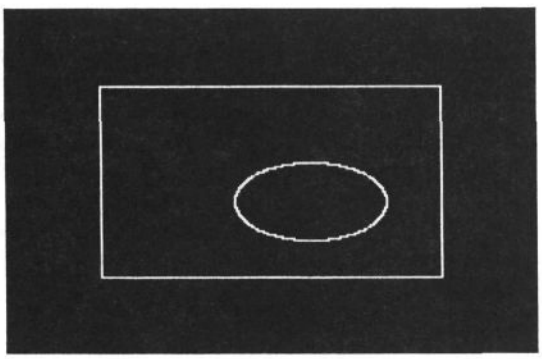

Figure 16: Superimposed rectangle and ellipse.

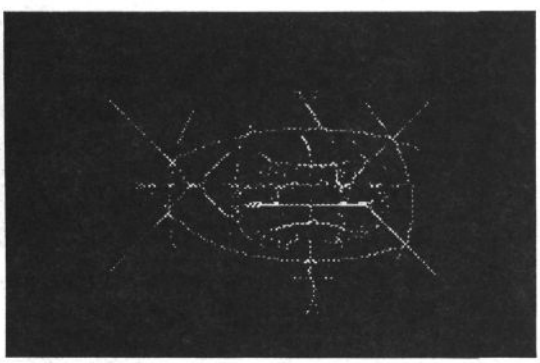

Figure 17: Symmetry axes of the rectangle/ellipse after 200 iterations.

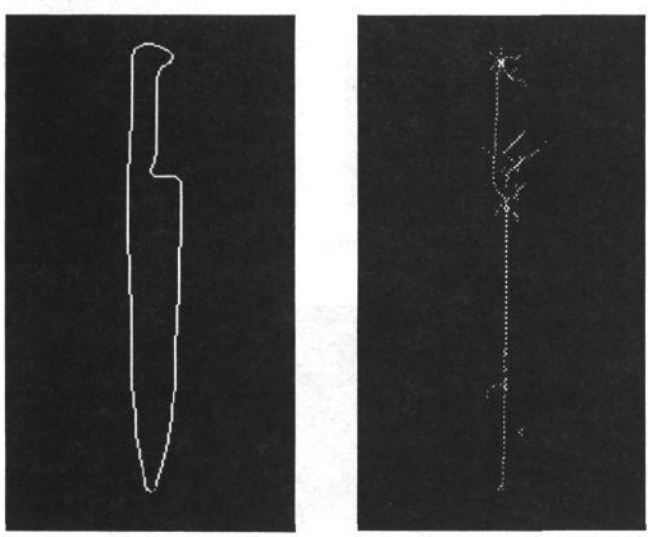

Figure 18: Carving knife and symmetry axes after 50 iterations.

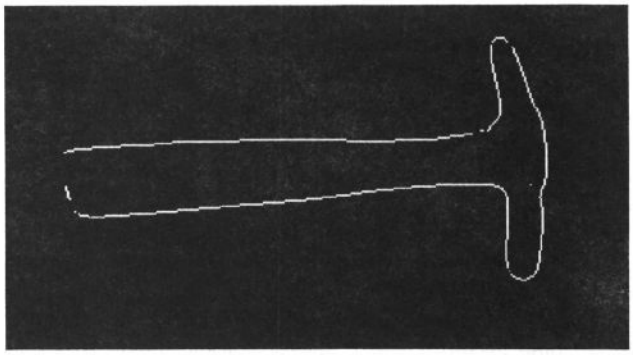

Figure 19: Tack hammer image.

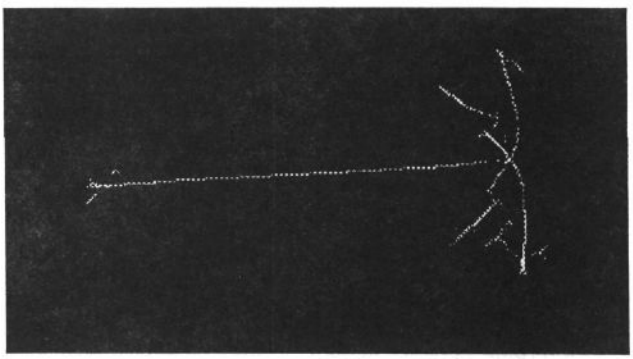

Figure 20: Symmetry axes of the tack hammer after 70 iterations.

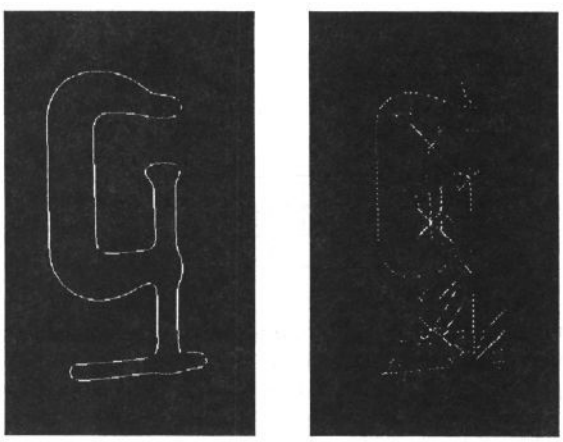

Figure 21: G clamp and symmetry axes after 120 iterations.

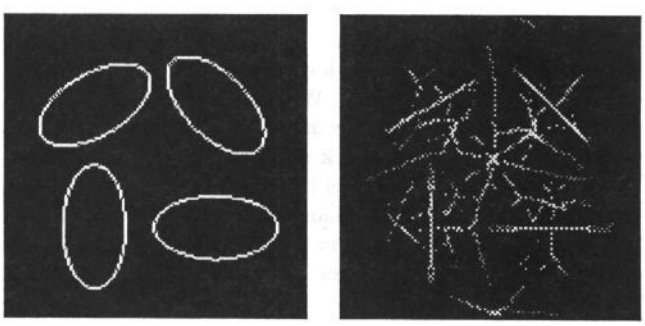

Figure 22: Group of ellipses and their symmetry axes after 70 iterations.

RECONSTRUCTION

We noted above that by recording the *maxima* of energy and their associated iteration numbers, we could effectively store the radius of circles on the symmetry axes and hence reconstruct the shape. Certainly the reconstruction of shape as an envelope of circles, based on the SAT or similar symmetry axis, is well known. The axes we store as the maxima of energy are not quite the same as those given by the Symmetry Transform – the S.T. implementation relies on the accumulation of energy over time – but are the axes relating to the most "significant" symmetries of the shape. We can reconstruct a somewhat simplified version of the original shape by running the wave process *backwards*, from the final iteration number back to iteration number 1. At any time that we have a record of a maximum arriving, we "fire off" a wave from that point, of "strength" (initial displacement) related to the height of the maximum energy at that pixel. When we reach iteration number 1, the overall wave displacement should have reached the edges of the original shape. The outline will, of course, be somewhat blurred and simplified due to the spread of the travelling wavefronts. Non maximal suppression should restore sharp edges and leave us with a simplified edge contour.

If we examine Figure 13, the edges of the shape show up strongly. This is because the edges have high initial displacement, and hence have high energy during the first few iterations. The reverse process, run off these images, would fire off waves at the edges at iteration 1, thereby ensuring that we get the shape back! If we run the original algorithm and ignore the maxima that arrive during the first few iterations, we do not store the edge waves in the first place. In addition, thresholding the resulting maximum displacement image to remove background noise and parts which are not obviously symmetry axes will further eliminate "spurious" parts of the image. Having done this, we may now attempt to reconstruct the shape without "cheating".

The actual algorithm used to reconstruct the shapes shown below was identical to the algorithm to find the original axes, with the reversing rationale outlined above; when the iteration number corresponding to the arrival of a maximum at a point is reached, a displacement equal to that maximum displacement is added in to the current displacement array. This causes waves to be "fired off" from that point at that time. The final wave displacement when we reach iteration number 1 is our reconstructed shape.

Examples of shape reconstruction from symmetry axes

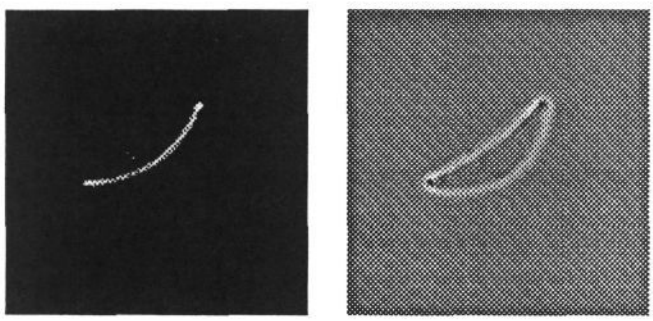

Figure 23: Symmetry axis and reconstructed outline of rotated smile.

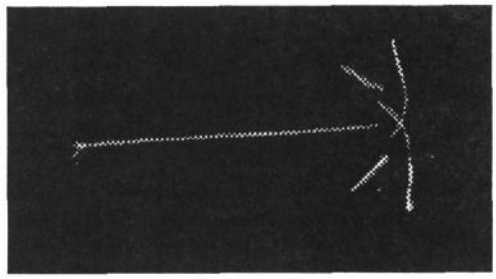

Figure 24: Symmetry axes of tack hammer.

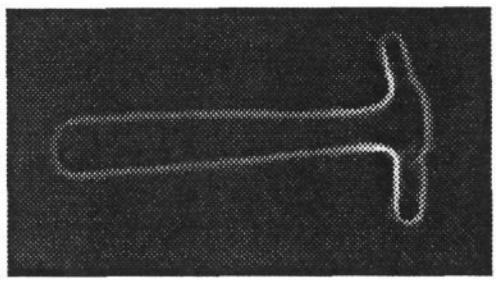

Figure 25: Reconstructed outline of tack hammer.

THE SYMMETRY TRANSFORM

We define the 2D Symmetry Transform $\mathcal{S}(\mathbf{x})$ of a signal $f(\mathbf{x})$ (where $\mathbf{x} = (x, y)$) as

$$\mathcal{S}(\mathbf{x}) = \int_0^\infty r^2 w(r) \left\{ \int_0^{2\pi} f(x + r\cos\theta, y + r\sin\theta)\, d\theta \right\}^2 dr \tag{6}$$

where $w(r)$ is a suitable weighting function (we use $w(r) = \frac{1}{r}$) Maxima in $\mathcal{S}(\mathbf{x})$ are points of "high bi-tangent symmetry" in the signal. This can be seen as follows:

At each point $\mathbf{x}$ in the image calculate the Radial Distribution Function

$$\mathcal{R}(\mathbf{x}, r) = \int_0^{2\pi} f(x + r\cos\theta, y + r\sin\theta)\, r d\theta \tag{7}$$

This will be strongly peaked wherever there are many points at the same radial distance from $\mathbf{x}$ (e.g. compare the R.D.F. for point C with that at point D in Figure 7). Now for any point the integral

$$\int_0^\infty \mathcal{R}(\mathbf{x}, r)\, dr = \int_{-\infty}^\infty \int_{-\infty}^\infty f(\mathbf{x})\, d^2x$$

will be the same. However, the integral of the *square* of the radial distribution function will be significantly larger at C than D (see Figure 26).

Finally,

$$\mathcal{S}(\mathbf{x}) = \int_0^\infty w(r) \left\{ \mathcal{R}(\mathbf{x}, r) \right\}^2 dr$$

gives a total measure of the radial organisation about $\mathbf{x}$. The weighting function gives progressively less emphasis to distant points.

It can be shown that

$$\mathcal{W}(\mathbf{x}) = \frac{1}{2c} \mathcal{S}(\mathbf{x}) \tag{8}$$

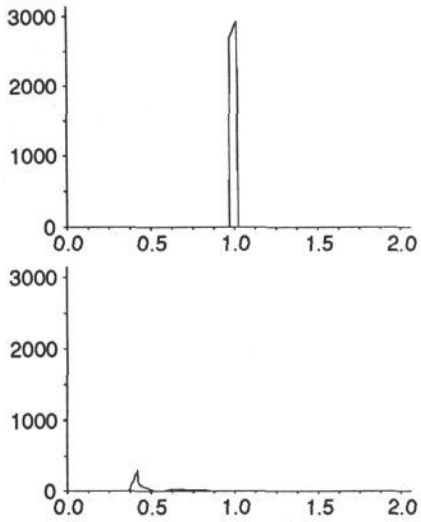

Figure 26: The square of the R.D.F. for points C (upper) and D of Figure 6.

That is, carrying out the Wave Transform calculates the Symmetry Transform.

ACKNOWLEDGEMENTS

We would like to thank Professor Mike Brady for his enthusiasm and the many invaluable conversations we have had with him. Alison Noble's many discussions with us have helped us to formalise our ideas. We are also grateful to Dr. Paul Leyland, who worked wonders with some of the routines in our code and brought about a large and very welcome decrease in processing time!

BIBLIOGRAPHY

1. Giblin, P. J. and the late Brassett, S. A. "Local Symmetries of Plane Curves" *American Mathematical Monthly* Vol. 92 No. 10 (1985).
2. Brady, J. M. and Asada, H. "Smoothed Local Symmetries and Their Implementation" *I.J.R.R.* Vol. 3 No. 3 (1984).
3. Leyton, M. "A Process-Grammar for Representing Shape" *Artificial Intelligence* 34 (1988).
4. Richards, W. and Hoffman, D. D. "Codon Constraints on Closed 2D Shapes" *Compute Vision, Graphics, and Image Processing* 31 (1985).
5. Canny, J. F. and Donald, B. R. "Simplified Voronoi Diagrams" *Proc. ACM Symposium on Computational Geometry* (1987).
6. Blum, H. "Biological Shape and Visual Science" *I. J. Theoret. Biol.* 38 (1973).
7. Pizer, S. M., Oliver, W. R. and Bloomberg, S. H. "Hierarchical Shape Description via the Symmetric Axis Transform" *Univ. of North Carolina Technical Report 86-016* (1986).