A Novel Approach to Boundary Finding

Richard T.Shann & John P. Oakley

Department of Medical Illustration
Department of Electrical Engineering
University of Manchester
M R I
Oxford Road

M13 9WL

A new class of algorithms is described for the analysis of boundaries in a discrete image. The methods are based on well known methods of Gaussian filtering coupled with the use of numerical methods. Instead of the conventional discrete filter and exhaustive evaluation by convolution the filter output is computed only at points of interest. In addition derivative information is gathered at these points which permits rapid convergence onto features of interest. Speed of computation and sub-pixel precision are features of these methods. The directional derivative of Gaussian and Laplacian of Gaussian are discussed and possible applications indicated.

This paper describes work in progress at Manchester University on a novel approach to the problem of finding boundaries in images using Gaussian filtering. The work originated from a study of the optimal method of reconstructing a filtered image from a knowledge of the filter operator and the sensor point spread function. The first working prototypes based on this approach are less than a year old and there seem to be quite a few avenues to explore. There are three areas currently receiving attention:

1) Linear measurement based on directional derivative of Gaussian operators. This work has reached the stage of being incorporated in a practical working system.
2) Boundary analysis using a variable angle directional derivative of Gaussian. Demonstration prototypes of this work are being tested.
3) Boundary analysis using the Laplacian of Gaussian filter.

The paper will present the background to the new method and some of the work done so far.

BACKGROUND

Current methods for locating objects in images typically use an edge extraction stage followed by a model fitting or pattern matching stage. One class of edge extraction algorithm employs Gaussian filtering, both to reduce noise and to define a spatial scale (controlled by the standard deviation of the Gaussian filter function) over which edges will be considered significant. So, for example, Marr and Hildreth [1] used the zero crossing contour of the Laplacian of Gaussian image as an estimate for the boundary.

Haralick [2] used the second order directional derivative, in the direction of maximum gradient, and found the zero crossings in the transformed image. Canny [3] described a scheme for edge analysis which involved evaluating the directional derivative of Gaussian in the direction of maximum gradient and marking local maxima as candidate edge points. Global thresholding (with hysteresis) then yields the edge points. Synthesizing the results obtained from differently sized Gaussians helps overcome the trade-off between positional accuracy and sensitivity. Alternatively, Bergholm [4] suggests tracking edges while changing the scale of the Gaussian.

THE NEW METHOD

In the schemes outlined above the filter is sampled to make a mask which is convolved with the image samples to give a discrete transformed image which is then searched. Feature will, in general, lie in between the pixel positions. The conventional approach to feature detection relies on some sort of interpolation of the transformed image. For example zero crossings are usually detected using linear interpolation. The sort of image being analysed in this approach is illustrated by fig. 1, where a Laplacian of Gaussian convolution has been performed on a coarsely sampled artificial image of a triangle. Clearly determining the zero crossing contour from such an image is no small task.

Instead of doing this we can reconstruct the transformed image between sample points directly in terms of the image samples. In this case the filtered image is a smooth function over the real plane, and we can apply numerical optimisation techniques to search it directly for features of interest. Fig. 2 shows the sort of image being searched in this case. This is the same triangle image used for fig. 1 but transformed using the optimal reconstruction filter discussed below. This image is presented only to give a feel for

AVC 1989 doi:10.5244/C.3.23

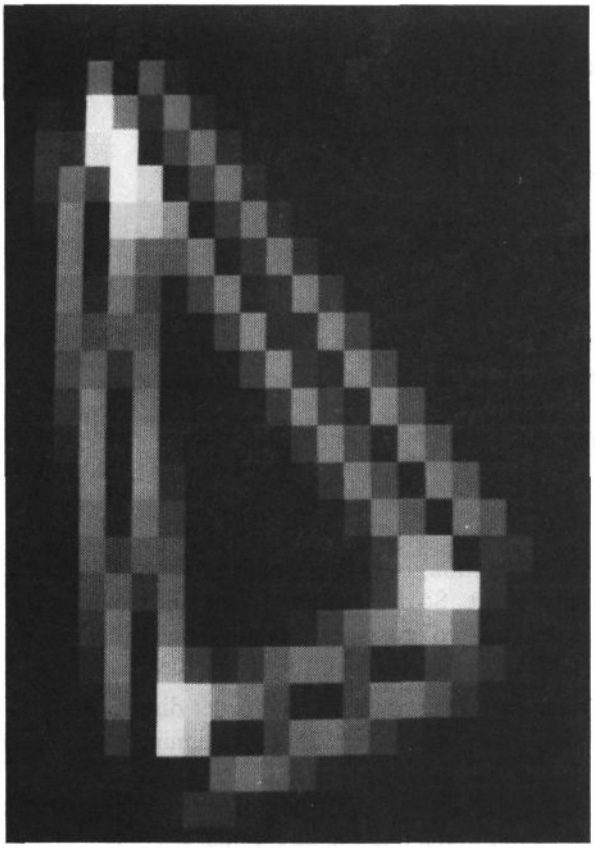

Fig. 1

Discrete Laplacian of Gaussian Image

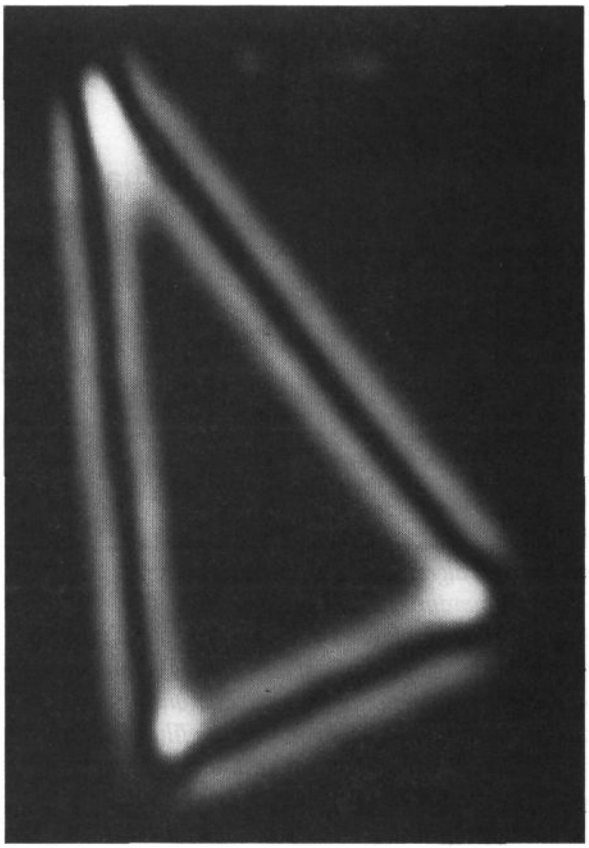

Fig. 2

Reconstructed Laplacian of Gaussian Image

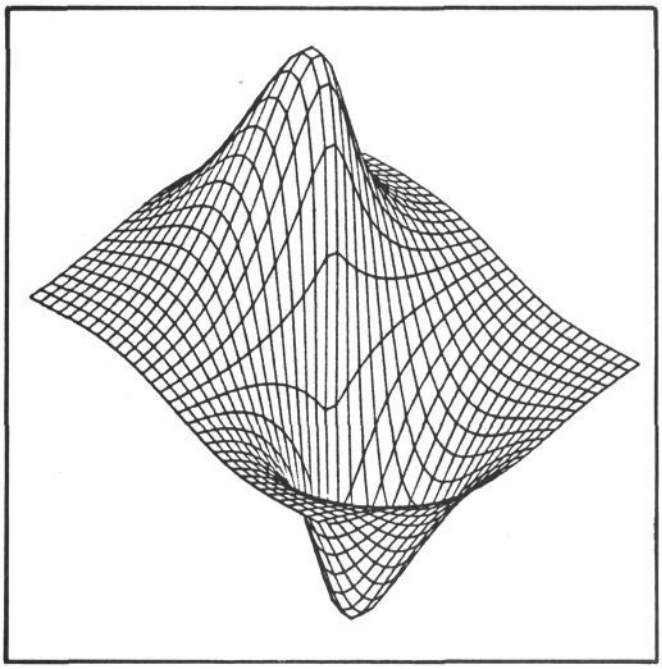

Fig. 3

Derivative of Gaussian Kernel

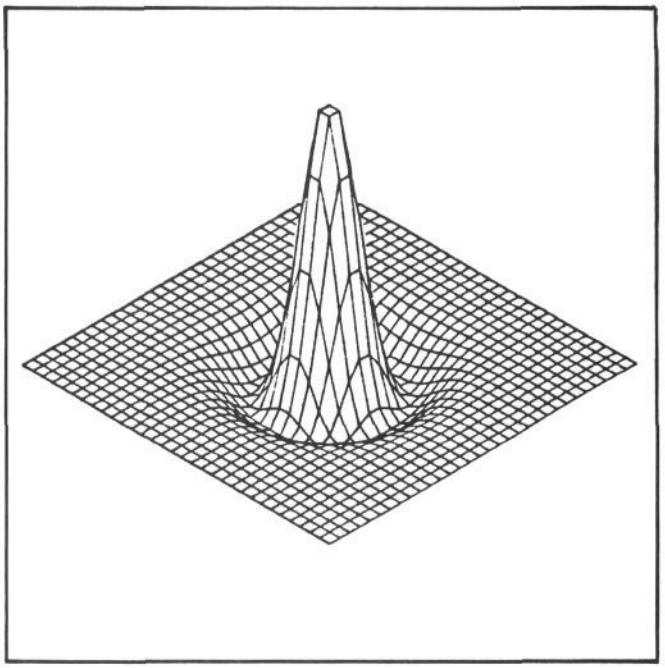

Fig. 4

Laplacian of Gaussian Kernel

what is happening, as we shall see below we avoid actually computing it as a whole.

In conventional techniques a subsequent stage of processing is normally needed after computing the convolution to identify edges and what the edges belong to. For example if a sought for object has a boundary which can be described by a few parameters one could optimise the parameters to fit the edge data found. This might then yield positional or dimensional information - often of sub-pixel accuracy. However it will frequently be the case that weak edges which one threw away in the first stage one would, with hindsight, wish one had retained, so one is tempted into increasingly model dependant algorithms, to-ing and fro-ing between the discrete transformed image data and the floating point world in which real objects reside.

In our new method we stay with the filtered image working all the time with the floating point coordinate system in which we hope to locate objects. The use of numerical analysis techniques to explore the surface of the filtered image means we never have to perform a convolution since we only calculate the surface at a few points, converging on the feature of interest. It is the avoiding of the convolution step which gives the present algorithm its potential for speed.The method is particularly suited to metrology problems, since the question of sub-pixel accuracy can be quite naturally dealt with.

Mathematical formulation

Given an image f(x,y) and a convolution kernel function K(x,y) we can define a filtered image Kf(x,y) by the convolution

$$Kf(x,y) = \int f(x'-x,y'-y)\, K(x',y')\, dx'dy' \qquad ...(1)$$

Since we only have samples fij of the image at the pixel positions (i,j), the usual approach is to sample the operator and calculate the discrete image

$$Kf_{ij} = \sum_{i'} \sum_{j'} f_{i'-i\ j'-j}\, K(i',j') \qquad ...(2)$$

This is time consuming to calculate even using discrete Fourier transforms, and once calculated, is not very pleasant to work with. To give a concrete example, consider for example the case

$$K(x,y) = (2/\sigma^2)(r^2/2\sigma^2 - 1)\exp(-r^2/2\sigma^2) \quad \text{where } r^2 = x^2 + y^2$$

- the well-known Laplacian of Gaussian filter. We know that the zero crossing contours of the filtered image correspond to edges in the image, but in general we won't find a sequence of pixels i,j such that Kfij is zero. Instead we must infer the edge position from adjacent pixels where Kf changes sign.

A more elegant approach is to consider the function

$$Kf(x,y) = \sum_{i} \sum_{j} f_{ij}\, K(x-i,y-j) \qquad ...(3)$$

This is a smooth function (assuming K is smooth) which we can easily calculate for any particular (x,y). We can see that it has the same value on the pixel boundaries as (2). In addition, we can just as easily calculate derivative information at any point of interest, eg

$$\frac{\partial Kf}{\partial x}(x,y) = \sum_{i} \sum_{j} f_{ij} \frac{\partial K}{\partial x}(x-i,y-j)$$

Using this information we can use standard numerical methods to explore the transformed image to perform such tasks as refining an estimate of edge position, tracking a boundary etc. The alert reader will have noticed that (3) is not in general the same as (1), the difference depending on how the fij relate to the f(x,y), ie on the sensor point spread function - for more details see the appendix. Another point is that because the Gaussian operators have infinite support we have to truncate the sum when calculating (3). This causes small discontinuities in the function and derivative values which can upset the more sophisticated numerical analysis algorithms.

THE DIRECTIONAL DERIVATIVE OF GAUSSIAN.

The directional derivative of Gaussian (see fig. 3) gives a filtered image with peaks at the locations of edges normal to the derivative direction. A numerical optimisation routine such as the NAG routine E04LBF will search such a function and find the local maximum. In order to make the search converge rapidly the NAG routine uses the first and second derivatives of the transformed image. In a typical application the direction (or approximate direction) and the approximate position of the edge are specified in advance by the user and the transformed image and its derivatives are computed at that one point for that one direction. On the basis of these values, the algorithm then chooses another point as a (hopefully better) estimate for the boundary location, and the process is iterated until a boundary has been found. This means that the transformed image is only evaluated at a convergent sequence of points, rather than at all points on the image raster, and this results in a huge saving in computer time, especially for large

operators. The angle of the directional derivative can also be treated as a parameter in the optimisation, in which case the derivatives with respect to angle are also computed. This algoritm was implemented by calling the NAG Fortran routines directly from Microsoft Pascal.

An examination of the second order partial derivatives at the final point in conjunction with the size of the peak found yields useful information about the nature of the peak. The first order derivatives will always be close to zero at a peak but some of the second order derivatives will be large if a sharp edge has been found, and the peak will be large if a long edge has been located.

The choice of the scale of the Gaussian used affects the positional accuracy, sensitivity and reliability. The distance to the nearest competing edge, the error in the seed position, the signal to noise ratio, and, finally the pixel size all place limitations on the sharpness of the filter that can be used. The factors involved here are similar to the ones discussed in relation to Canny's scheme, because the underlying filter is the same, but work remains to be done on relating the choice of filter size to the estimated error in the choice of seed point in a systematic way.

In common with all schemes using Gaussian filtering there is the usual distortion of edge position (caused by the curvature of the edge concerned) to be taken into account where necessary. At present there are two fields of interest being actively pursued. In one the application is an interactive measurement scheme. The edges of the object to be measured are indicated by the user, and the distance between the edges is reported by the algorithm typically accurate to 0.1 pixel. This has been successfully used for the measurement of the aspect ratio of drops of molten metal in order to determine the surface tension [5]. The other area of interest lies in the generation of boundary cues, for example to determine the location and orientation of parts in an industrial inspection situation.

THE LAPLACIAN OF GAUSSIAN

The Laplacian of Gaussian filter kernel is shown in fig. 4. The zero crossings of the Laplacian of Gaussian image can be tracked numerically directly in the floating point coordinate system (x,y). The approach taken starts with an initial estimate for a point on the zero crossing contour to be traced. This is refined by minimising the square of the Laplacian of Gaussian image. From this seed point the contour is tracked as follows. A maximum step S and a minimum step s are chosen based on the sigma value of the operator being used. A quadratic approximation Qf(x,y) is formed from the derivative information at the seed point, thus

$$Kf(x,y) \simeq Qf(x,y) \equiv$$

$$K_{xx}\, x^2/2 + K_{yy}\, y^2/2 + K_{xy}\, xy + K_x\, x + K_y\, y$$

where x,y are displacements from the seed point, and the K's are the derivatives evaluated at the seed point ie

$$K_x = \frac{\partial K}{\partial x} \qquad K_{xx} = \frac{\partial^2 K}{\partial x^2} \quad \text{etc}$$

The equation Qf(x,y) = 0 is a conic section which is our local model for the zero crossing contour. We now choose a step size (starting for safety with our minimum s). Then we can determine where the circle of radius s intersects this conic section and thus determine a new point at which to evaluate Kf. This in turn will yield a new quadratic approximation and so on. To simplify the computation in practice a step along the straight line in the direction perpendicular to the steepest descent is made and a straight line intersection is calculated from there. To permit termination we need to determine whether the original seed point lies on the current conic section, if so then we have terminated. (This test is only undertaken when we have returned to within the distance S of our seed point).

We also need to force termination when the boundary becomes to weak or broad. To do this several measures are available, the one used in this implementation comes from monitoring the control of the step size described next. Keeping to a conservative step size s for the whole contour would require more calculation and a more bulky representation than could be had by varying the step size to suit the local curvature. In this implementation we start with the small step size s, increasing it (up to the maximum S) for each successful step. A step is unsuccessful if the intersection with Qf(x,y) = 0 yields only imaginary or distant solutions, and in this case the step size is reduced and another attempt is made. If the step size drops below s then the edge has petered out and the algorithm terminates.

The output of this algorithm is a set of conic sections in the neighbourhood of a set of points. There is no guarantee however that these curves intersect at sensible points between, so that, if an application required it, a further stage of processing (eg weighted averaging) would follow. (This would be equivalent to smoothing out the truncation which occurs when calculating (3) above). As with the directional derivative of Gaussian there are geometric considerations which affect the position of such contours.

Results

This algorithm has been implemented in C on a Tandon AT hosting a dumb image store, and tried on several

different types of image. As expected the maximum step S which yields reliable tracking depends on the σ of the Gaussian.In addition there is a trade off between speed of sketching out a contour and the accuracy of the quadratic approximations for intermediate points. At too large a maximum step size S the time taken to trace out the contour can actually increase due to the back-tracking needed if a step fails. Typical values are shown in tables 1 and 2. The routines have been written with simplicity rather than speed in mind so that there is scope both on the hardware and software side for considerable speed improvements. Applications envisaged for this technique are measurements of area, aspect ratio, projected widths etc on blobs.

CONCLUSION

The preliminary work on these algorithms has been done on PC/AT computers, using a mixture of Pascal, C and Fortran. The results are very promising. Measurements can be taken and contours traced in a matter of seconds even with this relatively simple equipment. This is in marked contrast to the conventional convolution approach where the convolution step itself takes an half an hour on an AT, even using a good FFT algorithm, after which the problem of tracing a contour or determining the position of an edge still remains to be solved.

Max step size S	Number of arcs	Timing (sec)
1	160	90
2	88	54
3	82	61

Outline of image of a coin (perimeter 138 pixels)
Laplacian of Gaussian with σ = 1.6

Table 1

Max step size S	Number of arcs	Timing (sec)
1	153	143
2	91	97
3	55	58
4	45	48
5	36	39
6	45	58
7	57	84

Outline of image of a coin (perimeter 138 pixels)
Laplacian of Gaussian with σ = 2.5

Table 2

APPENDIX

The relation between fij and f(x,y) can be expressed by

$$f_{ij} = \int f(x,y)\, m(x-i,y-j)\, dxdy$$

where m is the sensor point spread function. (Ideally m would be unity over the area covered by the pixel (the unit square centred on the origin) and zero elsewhere). In general this transformation is not reversible, but given a point spread function m we could seek a reconstruction filter R such that

$$Kf'(x,y) = \sum_i \sum_j f_{ij} R(x-i,y-j)$$

is close to Kf(x,y) at all points for arbitrary images f. For the ideal m given above it can be shown (6) that this optimal R is given by the convolution of m with the filter kernel K.

$$R(x,y) = \int m(x',y') K(x-x',y-y')\, dx'dy'$$

Provided the pixel size is small compared with the distance over which K varies R will be the similar K, hence we can use (3) above.

ACKNOWLEDGEMENTS

Support for this work came from the Manchester Central District Research Grants Commitee (R.T.S.) and UK Science and Engineering Research Council (J.P.O.) and from facilities provided by the University of Manchester.

REFERENCES

[1] Marr & Hildreth, "Theory of Edge Detection", Proc Roy Soc Vol 207,1980

[2] Haralick, "Digital Step Edges...",IEEE PAMI VOL 6,1984

[3] Canny, "A Computational Approach to Edge Detection", IEEE PAMI VOL 8, 1986

[4] Bergholm, "Edge Focussing", IEEE PAMI VOL 9,1987

[5] J. P. Oakley & R. T. Shann "An efficient method for finding the position of object boundaries to sub-pixel precision", IEEE PAMI (1989) (to appear)

[6] J.P. Oakley and M.J. Cunningham, "A function space model for digital image sampling and its application in image reconstruction", CVGIP (1989) (to appear)