

Fast computation of moving average and related filters in octagonal windows

*C.A. Glasbey**

Biomathematics and Statistics Scotland

JCMB, King's Buildings, Edinburgh EH9 3JZ, Scotland

and *R. Jones*

CSIRO Mathematical and Information Sciences

Locked Bag 17, North Ryde NSW 2113, Australia

Abstract

A method is proposed for efficiently computing moving average filters in regular octagonal windows, as approximations to circular windows. The algorithm requires thirteen operations per pixel irrespective of window size and is faster than existing Fourier-based and recursive methods for all but the smallest windows. Further, new variance-weighted nonlinear filters are introduced, based on moving average filters. They are shown to smooth speckly images more effectively than existing filters.

Keywords: Fourier transform; Image processing; Isotropic filters; Nonlinear filter; Polygonal windows; Speckle; Synthetic aperture radar

*Corresponding author. E-mail: chris@bioss.sari.ac.uk

1 Introduction

Moving average filters are effective in smoothing noisy images, can be computed very quickly, and have well-understood mathematical properties. They are also building blocks from which other filters can be constructed. For example, Wells (1986) approximated Gaussian filters as convolutions of moving averages, Ferrari et al (1987) used B-splines to construct recursive implementations of linear filters, and Tomita and Tsuji (1977) and Lee (1981) proposed adaptive, nonlinear filters based on the moving average.

In one-dimensional signal processing, the moving average filter operates by replacing each data point by the arithmetic average of data values in an interval. It can be efficiently computed using a recursive algorithm: as the interval slides along the series of data points, the sum is

updated by adding to it the next data point encountered and subtracting the data point at the trailing end of the interval. Therefore, only two additions and one division are required per observation, irrespective of interval length. In two-dimensional image processing the arithmetic average is computed within a window. If this window is square or rectangular in shape then the filter is separable into two orthogonal one-dimensional moving sum filters: first the moving sum is applied along each row of the image and then it is applied down each column of the resulting image. The division by the area of the window takes place during the second pass. Only four additions and one division are required per observation, irrespective of window size (see McDonnell, 1981; Haralick and Shapiro, 1992, p304).

In some image processing applications it is preferable to compute the moving average filter in a non-rectangular window. In particular, for a rotationally invariant (i.e. isotropic) filter, the window should be circular, at least to a pixel approximation. Isotropic filters have several theoretical and practical advantages. For example, Davies (1984) showed the benefits of circular windows, and octagonal approximations, in the context of edge detection. Moving average filters in non-rectangular windows are non-separable and so take longer to compute. It is possible to use a recursive algorithm – a two-dimensional version of that above in which all pixels on the leading edge of the window are added to the sum and those on the trailing edge are subtracted. However, the number of operations is not size invariant; it is twice the diameter of the window. Ferrari and Sklansky (1984) proposed a method which involves taking sums and differences of partial sums of pixel values. The number of operations is two greater than the number of corners in the pixel representation of the boundary of the window. This algorithm is faster than the sliding window if the window has few corners, such as being L-shaped, but will be slow if the window is circular or has diagonal edges on its boundary. An alternative approach is to make use of the convolution theorem for linear filters. If an image and filter are both Fourier transformed and their element-by-element complex product is formed, then the inverse transform gives the required result. The algorithm complexity is independent of filter size and proves to be faster than convolution in the spatial domain for computing general linear filters in rectangular windows of size in excess of 7×7 (Glasbey and Horgan, 1995, p69).

A major drawback with all linear filters is that edges in an image are blurred in the smoothing process. Nonlinear filters are able to smooth while still preserving edges. However, most nonlinear filters are slow to compute and have less well understood properties than linear filters. (For recent reviews, see Fong et al, 1989; Wu et al, 1992.) Some nonlinear filters can be implemented efficiently because they use output from a moving average filter. For example, Lee's (1981) additive filter produces as output at position (i, j)

$$\eta_{ij} = \frac{\frac{f_{ij}}{\sigma^2} + \frac{\mu_{ij}}{V_{ij}}}{\frac{1}{\sigma^2} + \frac{1}{V_{ij}}},$$

where f_{ij} is the pixel value in row i , column j of an $n \times n$ image, σ^2 is the speckle, or observation error, variance (which is assumed known, but can alternatively be estimated from pixel variability in homogeneous regions of an image), μ_{ij} is the moving average output for a

$(2r + 1) \times (2r + 1)$ square window:

$$\mu_{ij} = \frac{1}{(2r + 1)^2} \sum_{k=i-r}^{i+r} \sum_{l=j-r}^{j+r} f_{kl} \quad \text{for } i, j = (r + 1), \dots, (n - r),$$

and V_{ij} is the variance of pixel values within the window, after subtraction of the speckle variance:

$$V_{ij} = \frac{1}{(2r + 1)^2 - 1} \left(\sum_{k=i-r}^{i+r} \sum_{l=j-r}^{j+r} f_{kl}^2 - (2r + 1)^2 \mu_{ij}^2 \right) - \sigma^2.$$

V_{ij} is also constrained to be positive, say no less than unity if pixel values have been scaled to lie in the range 0 to 255. (A value of unity rather than zero has been used because V_{ij} is a divisor in the filter.) Note that V_{ij} can be computed efficiently by applying a moving average filter to an image with (i, j) th pixel value f_{ij}^2 . In an edge-free neighbourhood, $V_{ij} \ll \sigma^2$ and $\eta_{ij} \approx \mu_{ij}$, whereas near edges, $V_{ij} \gg \sigma^2$ and $\eta_{ij} \approx f_{ij}$. The filter is derived as the minimum mean-square-error predictor of η_{ij} , if f_{ij} has mean η_{ij} , variance σ^2 , and η_{ij} has mean μ_{ij} , variance V_{ij} .

Computer timings can be very different for two similar filters, one of which is better suited to an efficient implementation. For example, Tomita and Tsuji (1977) proposed a filter which takes as its output the mean of the least variable subwindow among a set of square subwindows within the main filter window, i.e. the output at (i, j) is μ_{kl} where

$$(k, l) = \operatorname{argmin} \{V_{ij}, V_{i-r, j-r}, V_{i+r, j-r}, V_{i-r, j+r}, V_{i+r, j+r}\}.$$

Nagao and Matsuyama (1979) proposed a very similar filter, but using other shapes of subwindows with better corner-preserving properties. However, because there is no efficient algorithm for computing the averages in these subwindows, their filter is much slower to compute than Tomita and Tsuji's one.

In §2 we present a new algorithm for computing moving average filters in windows which are regular octagons, as approximations to circular windows, and show that it is faster than existing Fourier-based and recursive methods for all but the smallest window sizes. Then, in §3 we propose new nonlinear filters based on moving average filters. They are syntheses of Lee's (1981) and Tomita and Tsuji's (1977) filters, are fast to compute and smooth speckly images more effectively than existing filters. We illustrate their use on synthetic aperture radar (SAR) data. Finally, in §4 we discuss how the moving average algorithm can be generalised to other shapes of polygonal window.

2 Moving average filters

Fig 1 illustrates the superiority of an octagonal window over a square window as an approximation to a circular window. Fig 1(a) shows a simple binary image, and Fig 1(b) shows the result

of applying a moving average filter in a circular window. To the eye, results for octagonal and square windows with the same areas look very similar to Fig 1(b). To highlight the differences, Fig 1(c) shows the difference between Fig 1(b) and applying a moving average filter in a square window, employing a stretched grey-scale to emphasise the differences. On the same stretched scale, Fig 1(d) shows the difference between Fig 1(b) and applying a moving average filter in an octagonal window. It is clear that a moving average filter in an octagonal window is much closer to being isotropic than that in a square window, and filtering artifacts along rows and columns are avoided.

The sides of an octagonal window are piecewise linear. Therefore, pixel sums in the leading and trailing edges of a sliding window can be computed using one-dimensional moving sums, and the number of arithmetic operations per pixel is independent of window size. We give details of the algorithm below, which requires twelve additions and subtractions, together with one division, per pixel.

We define a lattice representation of a regular octagon centred at the origin and with radius r to be the set

$$S = \{(k, l) : |k|, |l| \leq r, |k + l|, |k - l| \leq r + p\},$$

where p is an integer chosen so that the Euclidean length of the vertical and horizontal sides of the octagon are approximately equal to the lengths of the diagonal sides, i.e.:

$$p \approx \frac{\sqrt{2}(r + 1) - 1}{\sqrt{2} + 2}.$$

An example is shown in Fig 2(a) for $r = 4$ and $p = 2$.

The output from a moving average filter is given by

$$\mu_{i,j} = \frac{1}{\chi(S)} \sum_{(k,l) \in S} f_{i+k,j+l},$$

where $\chi(S)$ denotes the number of elements in the set S (in Fig 2(a), $\chi(S) = 69$). If $\mu_{r+1,r+1}$ is computed by summing all pixel values in the appropriate window, then the filter can be recursively updated along the row by:

$$\begin{aligned} \mu_{i,j+1} = \mu_{i,j} + \frac{1}{\chi(S)} & \left(\sum_{k=-r}^{-p} f_{i+k,j+r+p+k+1} + \sum_{k=-p+1}^{p-1} f_{i+k,j+r+1} + \sum_{k=p}^r f_{i+k,j+r+p-k+1} \right. \\ & \left. - \sum_{k=-r}^{-p} f_{i+k,j-r-p-k} - \sum_{k=-p+1}^{p-1} f_{i+k,j-r} - \sum_{k=p}^r f_{i+k,j-r-p+k} \right). \end{aligned}$$

Similarly, the first value in each row can be computed recursively by sliding the window down a column. Fig 2(b) illustrates the pixels in the summation terms above for the octagon in Fig 2(a). The limits of the first three (additive) terms are shown outlined in bold on the right hand side of the octagon and the limits of the last three (subtractive) terms are shown in bold on the left hand side. The position of (i, j) is indicated by the black square.

All the six summations can themselves be computed recursively. Define three arrays as follows, for appropriate ranges of i and j :

$$A_{i,j} = \sum_{k=0}^{r-p} f_{i+k,j+k} , \quad B_{i,j} = \sum_{k=0}^{2p-2} f_{i+k,j} , \quad C_{i,j} = \sum_{k=0}^{r-p} f_{i+k,j-k} .$$

Then, if the A -array is computed in its first row and first column, the remaining elements can be computed as:

$$A_{i+1,j+1} = A_{i,j} + f_{i+r-p+1,j+r-p+1} - f_{i,j} .$$

Similarly, if the first row of the B -array is computed, the remaining elements are given by:

$$B_{i+1,j} = B_{i,j} + f_{i+2p-1,j} - f_{i,j} ,$$

and the remaining elements of C can be computed from elements in its first row and final column:

$$C_{i+1,j-1} = C_{i,j} + f_{i+r-p+1,j-r+p-1} - f_{i,j} .$$

These arrays can be used to recursively apply the moving average filter by:

$$\mu_{i,j+1} = \mu_{i,j} + \frac{1}{\chi(S)} (A_{i-r,j+p+1} + B_{i-p+1,j+r+1} + C_{i+p,j+r+1} - C_{i-r,j-p} - B_{i-p+1,j-r} - A_{i+p,j-r}) .$$

The six array positions in this equation are shown numbered consecutively in Fig 2(b), using the convention that the origin is in the top-left corner of the figure.

Table 1 gives CPU times in seconds for the new algorithm, implemented in C running on a DECstation 5000/200, for a range of octagon sizes on a 1024×1024 image. For comparison, times are also given for two alternative algorithms, a sliding window and the Fourier method. As expected, times for the sliding window increase linearly with window size whereas those for the Fourier approach are constant. Times for the new algorithm increase slowly with window size because of the time taken to initialise the arrays. It can be seen that the new algorithm is much faster than both the other two for all but the smallest window sizes.

3 Nonlinear filters

We propose a new nonlinear filter, a synthesis of Lee's (1981) and Tomita and Tsuji's (1977) filters, with output defined in terms of moving average filters in square windows, as follows:

$$\frac{\frac{f_{ij}}{\sigma^2} + \frac{\mu_{i-r,j-r}}{V_{i-r,j-r}} + \frac{\mu_{i+r,j-r}}{V_{i+r,j-r}} + \frac{\mu_{i-r,j+r}}{V_{i-r,j+r}} + \frac{\mu_{i+r,j+r}}{V_{i+r,j+r}}}{\frac{1}{\sigma^2} + \frac{1}{V_{i-r,j-r}} + \frac{1}{V_{i+r,j-r}} + \frac{1}{V_{i-r,j+r}} + \frac{1}{V_{i+r,j+r}}} .$$

This is the minimum-variance linear combination of f_{ij} and the means in $r \times r$ square sub-windows, if (small) covariances among the means are ignored. It can be evaluated very quickly

using output from the moving average algorithm (in particular, implementation time is independent of r). As with Lee's filter, it is assumed that σ^2 is known. Alternatively, it could be estimated from pixel variability in homogeneous regions of an image.

For octagonal windows, one way to modify the filter is as follows:

$$\frac{\frac{f_{ij}}{\sigma^2} + \frac{\mu_{i-r,j}}{V_{i-r,j}} + \frac{\mu_{i+r,j}}{V_{i+r,j}} + \frac{\mu_{i,j-r}}{V_{i,j-r}} + \frac{\mu_{i,j+r}}{V_{i,j+r}}}{\frac{1}{\sigma^2} + \frac{1}{V_{i-r,j}} + \frac{1}{V_{i+r,j}} + \frac{1}{V_{i,j-r}} + \frac{1}{V_{i,j+r}}}.$$

The change is necessary to ensure that (i, j) lies within all subwindows. We also propose a version of Tomita and Tsuji's (1977) filter based on octagonal windows. The output at (i, j) is μ_{kl} where

$$(k, l) = \operatorname{argmin} \{V_{ij}, V_{i-r,j}, V_{i+r,j}, V_{i,j-r}, V_{i,j+r}\}.$$

To compare the new filters with existing ones, artificial images were generated by segmenting a 250×250 image using a Poisson line process. This involved randomly positioning lines across the image, then allocating an intensity value, chosen randomly from a normal distribution with variance 100, to pixels lying in each of the polygons bounded by the lines. To this was added speckle noise with variance σ^2 , independently for each pixel. This method of simulation was used because it generates images similar to synthetic aperture radar images (see Fig 3(a)). If the number of lines intersecting the image is Poisson distributed with mean L , then on average $L/2$ lines will intersect a side of length 250 pixels. So there will be one intersection every $500/L$ pixels, on average, and the covariance between pixels in the simulated image will be

$$\operatorname{cov}(f_{ij}, f_{kl}) = \begin{cases} \sigma^2 + 100 & \text{if } (i, j) = (k, l), \\ 100 \exp \left[-\frac{L}{500} \sqrt{(i-k)^2 + (j-l)^2} \right] & \text{otherwise.} \end{cases}$$

All the previously discussed filters, together with the moving median (Huang et al, 1979) which is the most commonly used nonlinear filter, were applied to the simulated images using a range of window sizes. In each case the result was summarised by a root-mean-square error (RMSE), which is the square-root of the average squared difference between the filter output and the simulated image before the addition of noise, for the central 200×200 pixels. For each image and filter, the window size which minimised the RMSE was considered to be optimal. Table 2 summarises the results. For the moving average, moving median and Lee's filters, the RMSEs were very similar in square and octagonal windows, so only results for square windows are given. Except when σ^2 is large, corresponding to very low signal-to-noise levels (0.25:1), both forms of the new filter outperform the other filters by a comfortable margin. The octagonal window can also be seen to be superior to the square one for both Tomita and Tsuji's and the new filter.

To illustrate the use of the new filters, Fig 3(a) shows a log-transformed synthetic aperture radar (SAR) image of an area near Thetford forest, England, obtained as part of the Maestro-1 airborne campaign (Joint Research Centre, Ispra, report IRSA/MWT/4.90), and previously analysed by Glasbey and Horgan (1995). SAR is a form of remote sensing in which data have a

large noise component due to speckle, and therefore need smoothing before manual or automatic interpretation (Oliver, 1991). Durand et al (1987) found Lee’s (1981) filter particularly effective in smoothing SAR data. Horgan (1994) derived optimal linear filters for this image, and found the signal-to-noise ratio to be about 2:1 and correlations to decay exponentially at a rate $e^{-0.1}$. Therefore, this matches the simulations with $L = 50$ and $\sigma^2 = 50$. Figs 3(b)-(f) show the filters applied to the SAR image. For each filter, the size of window used was that previously identified as being optimal in the simulations. The new filter can be seen to combine the strengths of both Tomita and Tsuji’s and Lee’s methods: homogeneous regions are smoothed while boundaries are retained.

4 Discussion

The moving average algorithm presented above may be generalised to any shape of polygonal window. However, care must be taken because it will only be translation-invariant if the edges of the polygon are periodic lines (Jones and Soille, 1996). The algorithm is based on a recursive procedure where the sum is updated along an interval by adding to it the next data point encountered and subtracting the data point at the trailing end of the interval. Such a procedure will only be translation-invariant (and thus equivalent to the true moving sum filter) if the shape of the interval does not vary as the procedure progresses. For example, consider the interval shown in Fig 4(a) (a Bresenham line (Bresenham, 1965) of length 9 at angle 18.4°), where the origin is indicated by a black square and A , B and C are values in the image. The shape of the interval varies as the recursive procedure progresses: if the interval slides from left to right, the moving sum at position A is computed by adding B and subtracting C from the existing sum. However, this will in fact be the sum over the interval shown in Fig 4(b), which does not have the same shape as the original interval in Fig 4(a).

The only way to ensure that the shape of the interval remains invariant as the procedure progresses is to use an interval defined by a periodic line:

$$P_{\lambda, \vec{v}} = \bigcup_{i=0}^{i=\lambda} i\vec{v} .$$

Here $(\lambda + 1)$ (which must be positive) is the number of points in the periodic line and \vec{v} is a constant vector. A periodic line consists of a set of collinear points at a fixed distance apart; an example of a periodic line $P_{2,(3,-1)}$ is shown in Fig 4(c). To implement a moving sum filter defined in this interval using the recursive procedure, the interval must now slide left to right through steps of the vector v . For example, referring to Fig 4(c), once the value at the origin has been found the next value to be found will be A . The values in between the origin and A are computed during subsequent passes in the image (note however that the total processing time for all the passes combined will be the same as that for one pass with a non-periodic line).

All the edges of the octagon are periodic lines, and for this reason the algorithm that we presented in §2 for an octagon was translation-invariant. Unfortunately, the requirement that edges be defined as periodic lines becomes somewhat restrictive for other polygons. This problem can be overcome, with some sacrifice to processing speed, by using edges that are separable

into cascades of two orthogonal periodic lines. To illustrate the procedure, we will consider the example of a dodecagon. The top-right quarter of one is shown in Fig 5(a). If the dodecagon moves from left to right through the image, there are three leading edges in the quarter, as indicated by the numbers in the figure.

The edges marked by the numbers 1 and 3 in this figure are instances of periodic lines; the recursive procedure using these intervals will be translation-invariant. In contrast, the recursive procedure will not be translation-invariant for the edge marked by the number 2, as it is not a periodic line. However, the interval is separable into a cascade of two orthogonal periodic lines, as illustrated in Fig 5(b). The moving sum over interval 2 can therefore be implemented as a cascade of two moving sums using the periodic intervals shown, where each moving sum is implemented using the recursive procedure. In such case, four additions are required to compute the moving sum for interval 2, as opposed to the two additions required for each of intervals 1 and 3. This value is unaffected by the lengths of the periodic lines in the decomposition.

Acknowledgements

The first author's work was supported by funds from the Scottish Office Agriculture, Environment and Fisheries Department. This paper was started while he was visiting CSIRO's Division of Mathematics and Statistics.

References

- J.E. Bresenham (1965) Algorithm for computer control of digital plotter. *IBM System Journal*, 4, 25-30.
- E.R. Davies (1984) Circularity – a new principle underlying the design of accurate edge orientation operators. *Image and Vision Computing*, 2, 134-142.
- J.M. Durand, B.J. Gimonet, and J.R. Perbos (1987) SAR data filtering for classification. *IEEE Transactions on Geosciences and Remote Sensing*, 25, 629-637.
- L.A. Ferrari, P.V. Sankar, S. Shinnaka, and J. Sklansky (1987) Recursive algorithms for implementing digital image filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9, 461-466.
- L.A. Ferrari and J. Sklansky (1984) A fast recursive algorithm for binary-valued two dimensional filters. *Computer Vision, Graphics, and Image Processing*, 26, 292-302.
- Y. Fong, C.A. Pomalaza-Raez, and X. Wang (1989) Comparison study of nonlinear filters in image processing applications. *Optical Engineering*, 28, 749-760.
- C.A. Glasbey and G.W. Horgan (1995) *Image analysis for the biological sciences*. Wiley, Chich-

ester.

R.M. Haralick and L.G. Shapiro (1992) *Computer and Robot Vision*, volume 1. Addison-Wesley, Readings Massachusetts.

G.W. Horgan (1994) Choosing weight functions for filtering SAR. *International Journal of Remote Sensing*, 15, 1053-1064.

T.S. Huang, G.J. Yang, and G.Y. Tang (1979) A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27, 13-28.

R. Jones and P. Soille (1996) Periodic lines: definition, cascades and application to granulometries. *Pattern Recognition Letters*, 17, 1057-1063.

J.S. Lee (1981) Refined filtering of image noise using local statistics. *Computer Graphics and Image Processing*, 15, 380-389.

M.J. McDonnell (1981) Box-filtering techniques. *Computer Graphics and Image Processing*, 17, 65-70.

M. Nagao and T. Matsuyama (1979) Edge preserving smoothing. *Computer Graphics and Image Processing*, 9, 394-407.

C.J. Oliver (1991) Information from SAR images. *Journal of Physics D: Applied Physics*, 24, 1493-1514.

F. Tomita and S. Tsuji (1977) Extraction of multiple regions by smoothing in selected neighborhoods. *IEEE Transactions on Systems Machines and Cybernetics*, 7, 107-109.

W.M. Wells (1986) Efficient synthesis of gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8, 234-239.

W. Wu, M.J. Wang, and C. Liu (1992) Performance evaluation of some noise reduction methods. *CVGIP: Graphical Models and Image Processing*, 54, 134-146.

Algorithm	radius = 5	11	21	41	61
Fourier transform	104	104	104	104	104
sliding window	9	17	31	73	99
new algorithm	11	11	12	12	14

Table 1: CPU times (seconds) for existing and new algorithms for applying a moving average filter in an octagonal window to a 1024×1024 image

Expected no. of lines	σ^2	Moving average	Moving median	Lee's filter	Tomita/Tsuji's		New filter	
					square	octagon	square	octagon
12	25	1.98	1.67	1.62	1.65	1.49	1.24	<u>1.15</u>
	50	2.29	2.11	2.10	2.13	1.93	1.67	<u>1.56</u>
	100	2.42	2.43	2.46	2.71	2.29	2.20	<u>1.94</u>
	200	2.87	2.95	3.00	3.37	2.87	2.76	<u>2.42</u>
	400	3.10	3.27	3.35	3.99	3.42	3.34	<u>2.97</u>
25	25	2.55	2.20	2.05	2.21	2.00	1.74	<u>1.56</u>
	50	2.75	2.58	2.47	2.68	2.38	2.14	<u>1.96</u>
	100	3.24	3.24	3.17	3.53	3.21	2.91	<u>2.70</u>
	200	3.62	3.77	3.75	4.38	3.86	3.64	<u>3.35</u>
	400	4.01	4.26	4.33	5.10	4.51	4.35	<u>3.94</u>
50	25	3.08	2.78	2.47	2.64	2.56	<u>2.08</u>	2.08
	50	3.59	3.49	3.17	3.48	3.36	2.88	<u>2.73</u>
	100	3.94	3.97	3.80	4.46	3.91	3.74	<u>3.32</u>
	200	4.45	4.63	4.55	5.26	4.85	4.53	<u>4.18</u>
	400	<u>4.91</u>	5.21	5.24	6.22	5.65	5.31	4.99
100	25	3.95	3.51	2.98	3.52	3.30	2.66	<u>2.57</u>
	50	4.40	4.33	3.83	4.33	4.08	3.57	<u>3.42</u>
	100	4.99	5.20	4.68	5.33	5.06	4.58	<u>4.40</u>
	200	5.54	5.82	5.60	6.67	5.99	5.78	<u>5.30</u>
	400	<u>6.21</u>	6.56	6.49	7.49	7.07	6.65	6.23
200	25	5.49	5.10	3.54	5.23	4.88	3.70	<u>3.46</u>
	50	5.62	5.53	4.56	5.66	5.31	4.62	<u>4.32</u>
	100	6.12	6.42	5.65	6.52	6.17	5.70	<u>5.38</u>
	200	6.99	7.37	6.78	7.73	7.40	6.89	<u>6.62</u>
	400	<u>7.59</u>	8.15	7.80	8.89	8.38	8.19	7.64

Table 2: Root-mean-square errors of filters for optimal window sizes, based on 20 simulations of 250×250 images in each case. (The smallest value in each row is underlined)

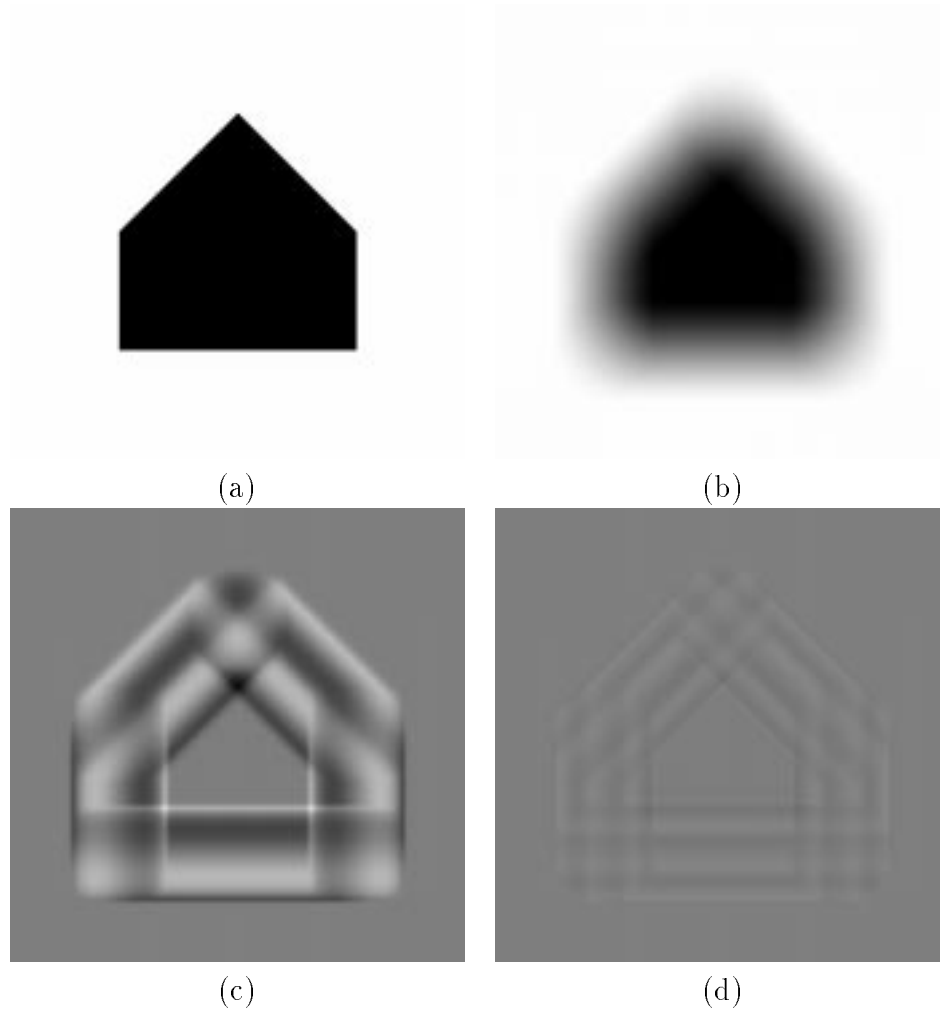


Figure 1: (a) Simple binary image, (b) result of applying a moving average filter in a circular window, (c) the difference between moving average filters in circular and square windows of the same area (zero values are displayed as mid-grey, positive values as lighter shades of grey and negative values as lighter shades of grey), (d) the difference between moving average filters in circular and octagonal windows of the same area (with the same grey-scale as in (c)).

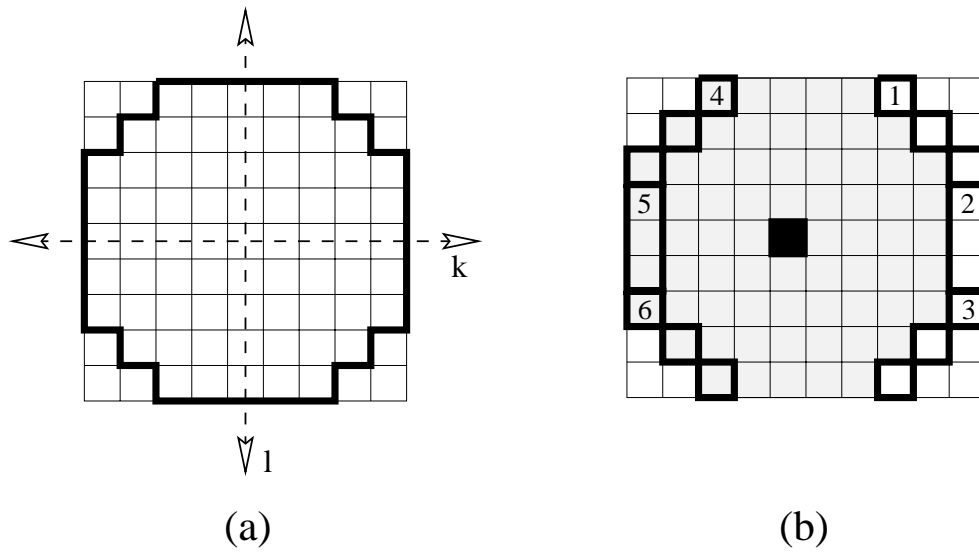


Figure 2: (a) Octagon constructed using $r = 4$ and $p = 2$. (b) Figure illustrating the recursive algorithm of the moving sum in this octagonal window (see text).

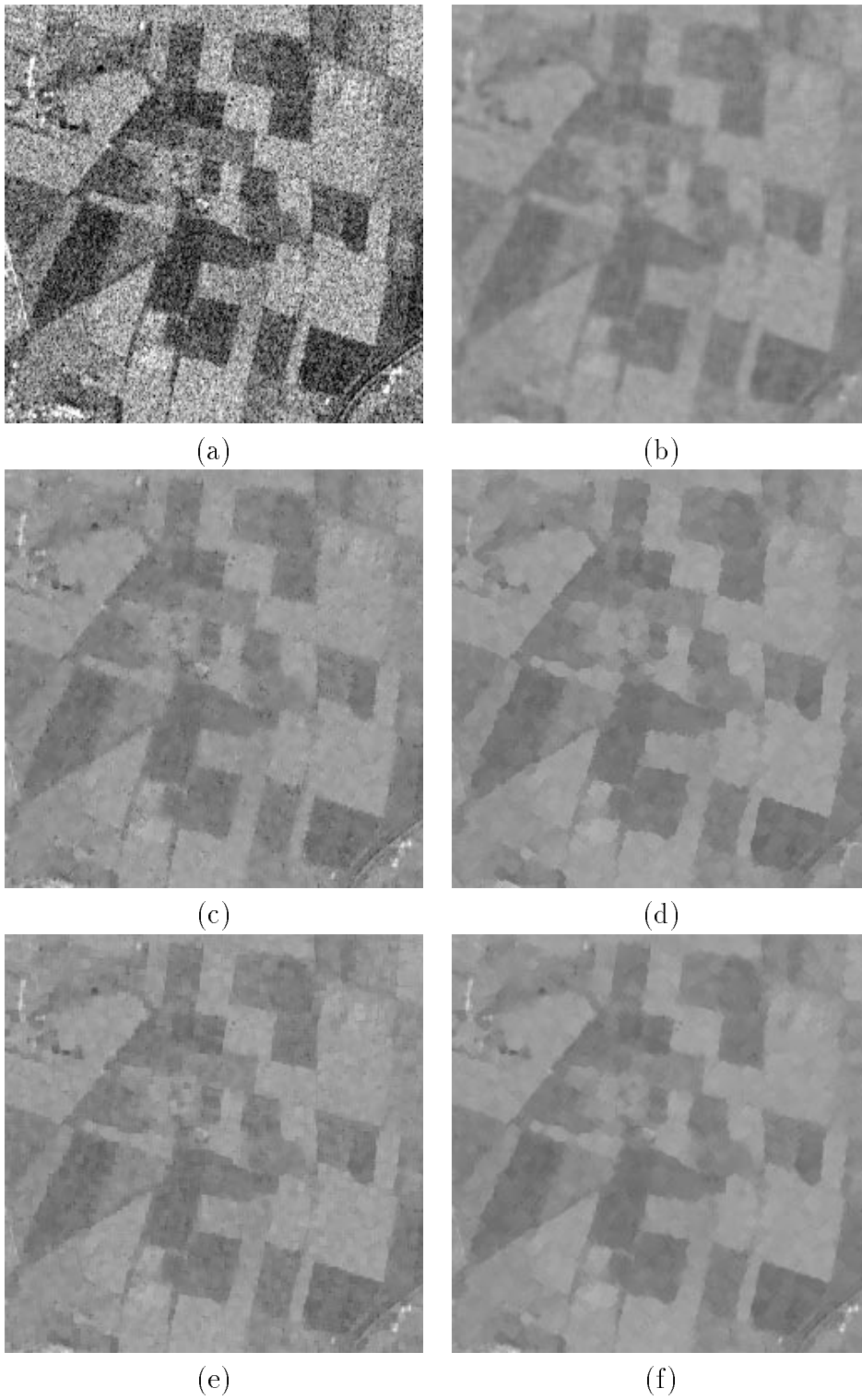


Figure 3: (a) SAR image, (b) moving median filter in square window ($r = 2$), (c) Lee's filter in square window ($r = 2$), (d) Tomita and Tsuji's filter in octagonal sub-windows ($r = 2$), (e) new filter in square sub-windows ($r = 1$), (f) new filter in octagonal sub-windows ($r = 2$).

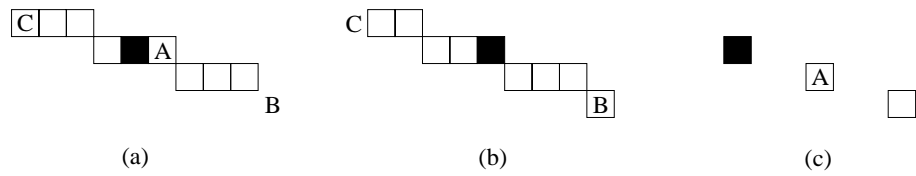


Figure 4: (a) Bresenham line of length 9 at angle 18.4° . (b) Effective Bresenham line at point A using the recursive procedure. (c) Periodic line.

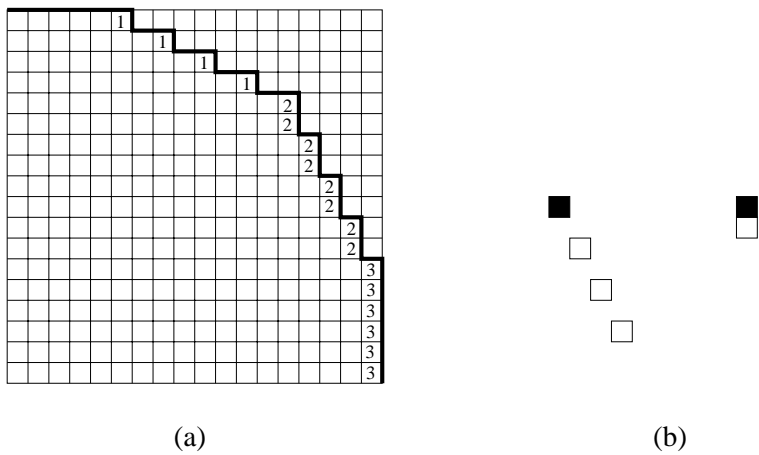


Figure 5: (a) Top right hand quarter of a dodecagon. (b) Interval 2 as a cascade of two periodic lines.