Available online at www.sciencedirect.com

SCIENCE DIRECT

Pattern Recognition Letters 27 (2006) 556–566

Pattern Recognition Letters

www.elsevier.com/locate/patrec

# Moving average algorithms for diamond, hexagon, and general polygonal shaped window operations

Changming Sun *

*CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde, NSW 1670, Australia*

Received 29 November 2004; received in revised form 9 June 2005
Available online 26 October 2005

Communicated by Prof. L. Younes

**Abstract**

This paper presents fast moving window algorithms for calculating local statistics in a diamond, hexagon, and general polygonal shaped windows of an image which is important for real-time applications. The algorithms for a diamond shaped window requires only seven or eight additions and subtractions per pixel. A fast sparse algorithm only needs four additions and subtractions for a sparse diamond shaped window. A number of other shapes of diamond windows such as skewed or parallelogram shaped diamond, long diamond, and lozenged diamond shaped, are also investigated. Similar algorithms are also developed for hexagon shaped windows. The computation for a hexagon window only needs eight additions and subtractions for each pixel. Fast algorithms for general polygonal shaped windows are also developed. The computation cost of all these algorithms is independent of the window size. A variety of synthetic and real images have been tested.


*Keywords:* Moving average algorithm; Diamond shaped windows; Hexagon shaped windows; Polygonal shaped windows; Local statistics

## 1. Introduction

In most of the image analysis and computer vision applications, the local processing windows are usually square or rectangular shaped. The edges of these windows are aligned with the image rows and columns. Because of the use of such simple shapes, efficient processing of images can be achieved. McDonnell (1981) described a box-filtering procedure for local mean calculation where the window is rectangular shaped. The main advantage of box-filtering is its speed, which approaches four operations for each output pixel and is independent of the box size. The filtering operation is also separable: two-dimensional filtering can be implemented as two 1D filtering.

Other shapes of windows are also used. A circular shaped window gives good isotropic property, but its computational cost is linearly proportional to the radius of the circular window. Glasbey and Jones (1997) presented fast algorithms for moving average and related filters in regular octagonal windows as approximations to circular windows. The algorithm requires twelve additions and subtractions per pixel irrespective of the window size. Ferrari and Sklansky (1984) proposed a two step method for obtaining the mean of an arbitrary shaped window. The number of operations is equal to the total number of concave and convex vertices of the window boundary. Because of the sampling effect, the boundaries of diamond and hexagon windows have many vertices, and the number of vertices also depends on the size of the window. Therefore Ferrari and Sklansky's method will not be very efficient for diamond and hexagon shaped windows. Verbeek et al. (1988) presented min or max filters for low-level image processing. They gave six shapes for the min or max filter, including a full square, a full diamond, a sampled diamond, a discrete approximation of a full circle, the rim and the center,

* Tel.: +61 2 9325 3207; fax: +61 2 9325 3200.
*E-mail address:* changming.sun@csiro.au

0167-8655/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2005.09.020

and eight contour points and the center. The computation cost of the full diamond shaped min or max filter is proportional to the size of the window. Soille and Talbot (2001) presented a decomposition method of morphological operations for diamond shaped and rotated rectangles. van Herk (1992) also developed a fast algorithm for local min or max filters on rectangular and octagonal kernels. van Droogenbroeck and Talbot (1996) presented a general algorithm that performs basic mathematical morphology operations with any arbitrary shaped structuring element in an efficient way.

In some applications such as image processing and stereo matching, the processing window can be diamond or hexagon or general polygon shaped. In stereo matching applications, different shaped windows can be used for calculating correlation coefficients. The shapes of these windows can be adaptive to the orientation of the object boundaries. Diamond shaped window could be used when object boundary are roughly in the diagonal direction as shown in Fig. 1. Square and diamond shaped windows are shown at object boundary in Fig. 1(a) and (b), respectively. The center of the diamond window are closer to the object boundary than that of the square window without intersecting with the boundary. Baaziz and Dubois (1993) used separable diamond shaped filtering for hybrid HDTV image sequence coding. Diamond shaped window can also be imagined as a rotated version of a square window as shown in Fig. 2, although in discrete space the sides of the diamond window have zig zag shapes.

In this paper we will present fast moving windows algorithms for the calculation of local statistics such as mean, variance, skew, and correlation using a diamond or hexagon or general polygonal shaped window. The topic is important for real-time applications. Our algorithms require only seven or eight additions and subtractions per pixel, while a sparse algorithm requires only four additions and subtractions per pixel for local sums calculation for a sparse diamond shaped window. Other variations of diamond windows will also be investigated. Our algorithm for hexagon shaped window requires only eight additions and subtractions. The computational cost for general polygonal shaped window is also given as a simple formula.

The rest of the paper is organised as follows: Section 2 describes three algorithms for local mean calculation in diamond shaped windows. Section 3 gives two algorithms for sparse and multiple-shift diamond windows. Section 4 presents fast algorithms for other variations of diamond window shapes, including skewed or parallelogram shaped window, long diamond shapes, lozenge shapes, and rotated diamonds. Section 5 shows algorithms for hexagon shaped windows. Section 6 gives algorithm for general polygonal shaped windows. Section 7 describes methods for extending the local mean calculation to variance, skew, and correlation calculations. Section 8 shows the experimental results obtained using our fast algorithms applied to a variety of images. Section 9 gives concluding remarks.

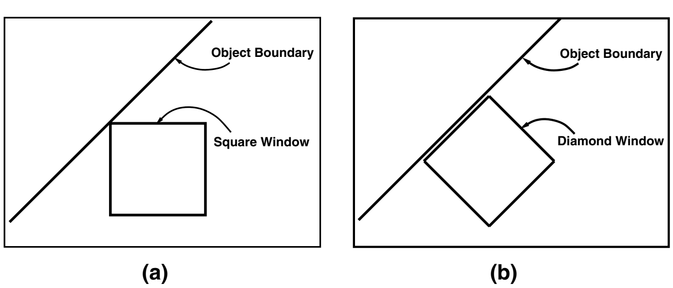


Fig. 1. Different windows at an object boundary. (a) Square window at object boundary and (b) diamond window at object boundary.

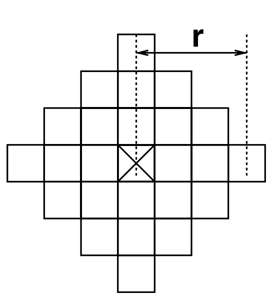


Fig. 2. Diamond shaped window. The cross is the window center, and $r$ is the radius of the window. The value of $r$ in this figure is 3.

## 2. Diamond shaped local sum calculation

In this section we propose three algorithms for obtaining the local sums in a diamond shaped window of an image. Fig. 2 shows the shape of a diamond window, and the cross in the figure indicates the window center. The size of the window is defined by its radius $r$. The size, or area, of the diamond shaped window with radius $r$ is then $2r^2 + 2r + 1$. There will be one division for each pixel on the images for obtaining the mean value from the local sums. Because division operation is usually expensive, one may wish to use just the local sums. In the rest of this paper, we will concentrate on the calculation of local sums.

### 2.1. Edge-Updating algorithm

Assuming the local sum of a diamond window has been obtained at a particular position, when we slide the window horizontally to the right by one pixel to find the new sum, we only need to add in the pixel values from the leading edge with black circles and triangles and subtract out the pixel values from the trailing edge with white circles and triangles as illustrated in Fig. 3. For those pixels marked with circles, they lie on the lines with 45° angles from the horizontal direction. For those pixels marked with triangles, they lie on the lines with −45° angles from the horizontal direction.

Note that the sums of those pixel values on diagonal lines can be obtained using the moving window idea as shown in Fig. 4. The computational cost is only two additions and subtractions for each point on a particular line