A HISTOGRAM METHOD FOR FAST GREYSCALE MORPHOLOGY OPERATIONS

D.M.P. Hagyard, M. Razaz and P. Atkin *

School of Information Systems,University of East Anglia, Norwich, England
* Synoptics Imaging Systems Ltd., Cambridge, England
Email: mr@sys.uea.ac.uk

ABSTRACT

A novel histogram method is presented for fast computation of greyscale morphology operations. This method is compared against the list method, a brute force approach. The histogram method is shown to be considerably faster than the brute force method, and that its computation time is independent of the size of the structuring element. For comparison purposes we have also implemented an alternative fast algorithm based on the van Herk approach for performing maximum and minimum filters on a 1D array of data. The histogram and van Herk methods are compared and contrasted. Both methods are fast but the histogram method is more flexible in dealing with different types of structuring element shapes.

1. BRUTE FORCE METHOD

The brute force implementation of the greyscale morphological primitives converts a structuring element into a list, where each member of the list contains an offset from the origin and a colour value. For each pixel on the image the list is scanned. When performing a dilation, for each member of the list the offset is added to the current pixel location and the list colour value is added to the colour at that location. Once the list has been scanned, the maximum value found after the addition is written into the current location. When performing an erosion, the same process is followed except that the list colour value is subtracted from the image value, and the minimum value found during the list scanning is written into the output pixel. To perform Minkowski addition and subtraction the same methods as dilation and erosion should be followed, except that the offset values should be subtracted from the current pixel location instead of being added.

The brute force method is capable of implementing any structuring element shape. This flexibility comes with a price however. The algorithm is very slow, as its computational complexity is O(n.m), where n is the number of pixels in the image and m the number of pixels in the structuring element. For the binary version of the brute force implementation of dilation it was possible to stop the scanning through the list on locating the first black pixel, since at that point the output pixel had to be black. Similarly for erosion it was possible to cease scanning the list once the first white pixel was found. The fact that the entire list did not always have to be examined allowed the binary algorithms a slight speed up on real images that are not completely black. However, for the greyscale brute force algorithm it is less likely that scanning through the list could be ended early. Halting the scanning can only occur when a pixel value comes to a maximum (for dilation), or to a minimum (for erosion).

2. HISTOGRAM METHOD

There are a number of methods in the literature for calculationg greyscale primitive operations [1,2,3]. We present a new histogram method developed for fast computing of greyscale morphological primitives. The method is designed for operating on 1D arrays, and is extendible to 2D images by performing a primitive operation in a number of directions. The method operates by passing a window, the same width as the structuring element, over the input buffer. A histogram data structure maintains a count of the number of pixels of each colour under the window. As the window is scanned along the input, the histogram is updated with the incoming and outgoing pixel values. The histogram value at the colour of the outgoing pixel is incremented at the colour of the incoming pixel and decremented at the colour of the outgoing pixel. This allows the algorithm to maintain a count of the number of pixels under the window with only two operations per pixel moved. For dilation the maximum value under the window is maintained. This is easy to maintain as long as the maximum value is never the outgoing pixel, since it is only necessary to check that the incoming value is not larger than the current maximum and changing the maximum if it is. If the outgoing pixel value is maximum then the algorithm will have to find the next largest value. The ease with which this can be done depends on the type of data structure used to store the histogram. Performing an erosion requires a similar approach except that the algorithm must maintain the minimum value under the window instead of a maximum.

The histogram method is a 1D maximum or minimum filter, and therefore will perform only flat greyscale morphology. If the values under the window are incremented each time the window is moved then the algorithm will perform a greyscale morphological operation with a sloping or ramp shaped structuring element. By performing the operation in two directions, a 1D cone shaped structuring element can be implemented.

The histogram method was implemented using an array of 256 integers, with the q^{th} array element holding the number of pixels of colour q under the window. The buffer is initialised with zeros. As the window is moved across the line of pixels, the incoming pixel value is added to the window by incrementing the pixels value in the array. The outgoing pixel is subtracted from the window by decrementing the pixel value. For dilation, the procedure keeps a pointer at the array position that contains the current highest pixel value under the window. If the outgoing pixel value is maximum then a new maximum value under the window has to be found. This is done by searching down through the array starting from the outgoing maximum looking for a non zero value. For erosion, if the outgoing pixel is minimum and its removal will zero the array value a new minimum has to be found. This is done by searching upwards through the array starting from the outgoing minimum looking for a non zero value. The number of times that this process must be followed for either erosion or dilation depends on the shape of the input image. At the worst case the image could contain isolated peaks of 255 pixels separated by areas of zero pixels each slightly wider than the window length. This would cause the algorithm to search through the entire array every time the maximum 255 pixel left the window. In the most likely case ‘loosing’ the maximum will happen fairly frequently However the length of array searched through will not be very long because most real greyscale images do not have particularly sharp gradients.

To perform erosion or dilation by a sloping structuring element it is necessary to increase or decrease all the pixel values under the window each time the window is moved. Changing the window values could be performed by scanning through the entire histogram array and shifting the values up or down each time the window moves. The number of operations required for each output pixel is equal to the size of the histogram array. The size of the histogram would have to be equal to the number of grey levels in the image plus the width of the window. The extra size is needed to account for cases during a dilation when a pixel value of 255 enters the window and is shifted past the maximum value in the image. Although the large value cannot be displayed it is still the maximum, its value, clipped to 255, stored and written to the output. Similarly for erosion, zero values entering the window will become negative and will need an array equal to the length of the window plus 256 to store the histogram. When removing pixels from the histogram as they leave the window, it is necessary to decrement the histogram location at the pixel value modified by the width of the window to take account of the fact that the pixel value has been shifted.

To avoid scanning the entire histogram for each output pixel, a slightly more complex data structure was implemented. The histogram array is defined to be circular and with a pointer to the current location of the zero value set. Incoming and outgoing pixels are maintained, shifting the outgoing pixel value by the window width as before. The procedures for carrying this out are slightly more complex due to the maintenance of the circular array. After the incoming and outgoing pixels have been processed the zero location pointer is moved one place down for dilation, and one place up for erosion. This performs in one operation shifting the contents of the entire histogram array up or down.

3. VAN HERK METHOD

The van Herk algorithm performs maximum and minimum filters on a 1D array of data in linear time with respect to the number of pixels in the array [3]. The input to the algorithm is a 1D buffer of pixels, I, which contains n pixels. One end of the array is padded with zeros to make its length, n’, up to a multiple of the structuring element width m. This string of values is used to fill two arrays, G and H which are both n’ long. For dilation, the two arrays are divided into n’/m slices of m pixels each. In each slice of array G the first pixel (counting from the left) is compared with the second pixel and the largest value placed in the second pixel location. In the next step, the new second pixel value is compared with the third pixel value and the maximum written back to the third location. This process continues until the m^{th} pixel in each slice has been written to and tends to ‘smear’ the maximum value to the right. The H array is processed in the same manner except the operation is performed starting at the right hand edge progressing to the left. To produce the final output, the G array is shifted m - 1 pixels to the left. The values in each array are compared and the maximum is put into the output array O, i.e. O[p] = max(G[p], H[p - (m - 1)]). Any values that are not in G or H are assumed to be 0.

For erosion the two arrays, G and H, are divided into n’/m slices of m pixels each. In each slice of array G the first pixel (counting from the left) is compared with the second pixel and the smallest value placed in the second pixel location. In the next step, the new second pixel value is compared with the third pixel value and the minimum is written back to the third location. The process continues until the m^{th} pixel in each slice has been written to and hence it tends to ‘smear’ the minimum value to the right. The H array is processed in the same manner except the operation is performed from right to left. To produce the final output the G array is shifted m - 1 pixels to the left. The values in each array are compared and the minimum put into the output array O, i.e. O[p] = max(G[p], H[p - (m - 1)]). The comparison is run starting from G[m - 1] running for n - (m - 1) locations.

Once the buffers are filled the algorithm requires only 3 operations per image pixel, thus making the algorithm very efficient. The computational complexity of the algorithm is O(n’) and is independent of the size of the structuring element. The algorithm efficiency is reduced by the necessity of filling the buffers, and the buffers may need padding. Memory requirements are low since the algorithm is applied to one row at a time.

4. RESULTS

Both the flat and the sloping histogram methods and van Herk algorithm were implemented for 1D images and tested against the greyscale brute force method. The sloping histogram method was extended to performing the peaked structuring elements by running the algorithm in both forwards and backwards directions. Experiments were carried out to check that the methods produced the correct results and to examine how quickly the algorithms could perform dilation and erosion.

The timing tests were performed on an 864 x 864 greyscale image. The results presented are for the 1D scanning of each of the 864 rows in the image. To preserve the edge information, the output image size is increased for dilation and reduced for erosion by the width of the structuring element minus 1. The results of the brute force method are presented separately in Figure 1 because the computation time for both operations grows, as expected, exponentially with the structuring element size.

Figure 2 displays the timing results for the fast erosion methods. Comparing Figures 1 and 2 shows that the histogram and van Herk methods are much faster than the brute force algorithm. All the methods are linear with respect to the number of pixels in the structuring element. The van Herk algorithm appears to be slightly faster than the histogram method. The extra overhead involved in maintaining the circular histogram data structure makes the sloping histogram method slower than the flat histogram method upon which it is based. The peaked histogram method because it is implemented by performing two sloping erosions in opposite directions takes approximately twice as long as the sloped structuring element. For the peaked histogram method there is a noticeable downward trend as the structuring element size increases. There is a slight downward tendency in the other graphs, but they are not as noticeable as the trend for the peaked histogram.

Figure 3 compares the histogram and van Herk methods for computing the dilation operation. Similar observations to those made for the graph in Figure 2 can be made here. All the methods show a slight upward trend as the structuring element gets larger, this is attributable to the output image size getting larger. For both the erosion and dilation operations the time taken is proportional to the number of pixels in the output array, and independent of the structuring element size.

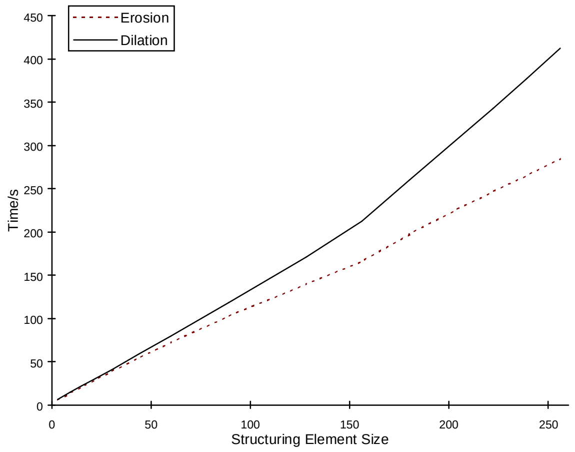

Figure 1. Graph of Time to dilate and erode an 864 black square image, using the Brute Force Morphology method with various sizes of 1D structuring element.

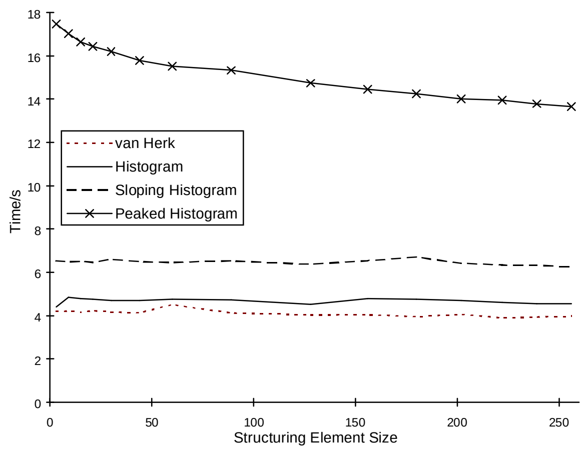

Figure 2. Graph of Time to erode a 864 black square image, using various Morphology methods with various sizes of 1D structuring element.

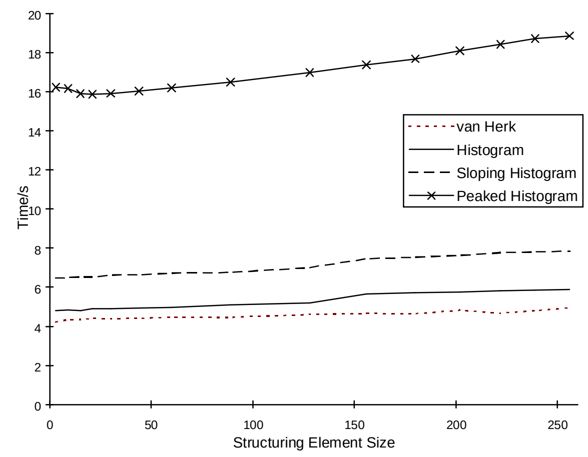

Figure 3. Graph of Time to dilate a 864 black square image, using various Morphology methods with various sizes of 1D structuring element.

5. ACCURACY

The testing of the algorithms was carried out by comparing the results of the new methods with the brute force. These showed that the new methods for using flat structuring elements operated correctly. The methods for using non binary, or non flat structuring elements showed a number of odd effects as described below.

When performing an erosion operation it is easy to see that the pixels around the image can be ignored, since the output should only contain pixels where the entire structuring element has been fitted into the image. When performing a dilation the output will contain pixels that are the result of the structuring element covering both on and off-image pixels. If the structuring element is flat there is no problem since the off-image area will never have a maximum value under the structuring element unless the off-image area is greater than zero. If the structuring element is non flat and the off-image area is defined to be zero, then the off-image value plus the value of the structuring element at that point may have a maximum value (that gets used for the output pixel). If the off-image area is defined to be minus infinity then no pixel in the structuring element can make an off-image pixel more than the value of an on-image value, assuming that the structuring element contains only positive values. A cone shaped structuring element dilating a blank image (with all pixels at zero and the off-image area also defined as zero) will produce a completely flat image at a level equal to the maximum value of the structuring element. If, however, the off-image area is defined as minus infinity the resultant image will have a bevelled appearance. The area in the middle of the output the size of the input image will show the maximum value of the structuring element. Around this area the image values will slope down in a manner defined by the shape of the structuring element. This can be observed in Figures 4 to 6.

If all the pixels in the neighbourhood of the image edge, as defined by the structuring element, are larger than the maximum value in the structuring element, then the definition of the off-image area can have no effect.

With erosion and dilation, all the pixels of the input image will be altered by the height of the maximum value in the structuring element. In the case of the flat structuring elements, all the members of the structuring element will be zero. For a non-flat structuring element the entire image is incremented (for dilation) or decremented (for erosion) by the maximum value of tthe structuring element. For most applications the change in image value may be acceptable, and the problem can be rectified by a simple addition or subtraction.

It should be remembered that the representation of the image in the computer may have a limited number of grey levels available to it, leading to the loss of peaks in an image under dilation, or to the loss of troughs in an image under erosion. This could be less expected under opening or closing, since these operations do not overall shift the height of the image pixels, except in the intermediate steps. Therefore the use of large ball or cone shaped structuring elements could cause problems for erosion and dilation due to a limited number of grey levels.

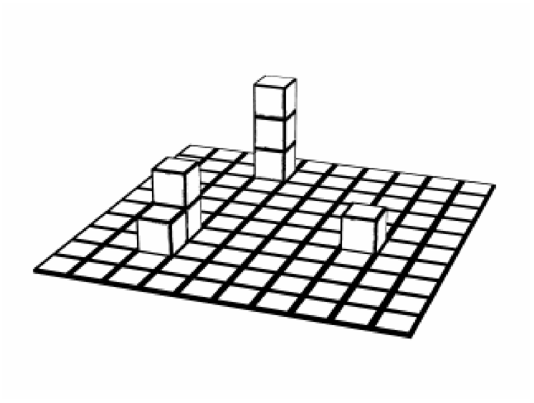

Figure 4. Original image presented as a 3D umbra.

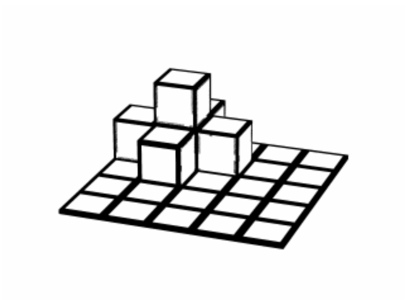

Figure 5. Structuring element presented as a 3D umbra.

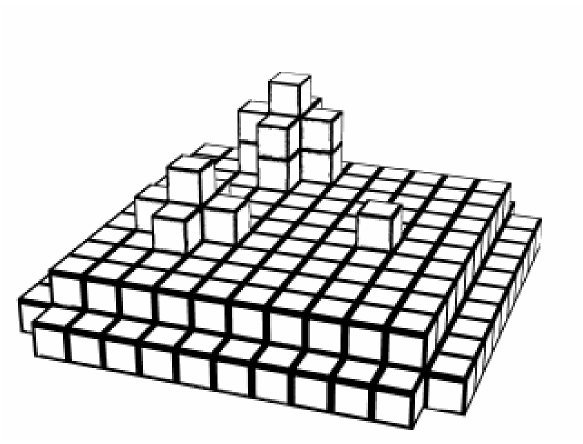

Figure 6. The result of performing a dilation of the image umbra in Figure 4 with the structuring element in Figure 5 presented as an Umbra. Observe the bevelling around the edge of the image.

The peaked structuring element is the result of applying a sloped structuring element in two opposite directions. This is equivalent to using a 1D cone shaped structuring element. The result is not quite as expected in that the combination of the two structuring elements is a peak, but one offset by the height of one of the structuring elements. The structuring element shape will leave the entire image too high if the operation is dilation, and too low if the operation is erosion. The problem can be solved by vertically offsetting the entire image to make the result correct. This may produce incorrect results if the number of grey levels in the image is such that peaks in the image will be truncated on dilation, and troughs in will be flattened on erosion.

6. CONCLUSIONS

We presented a novel and fast histogram method for computing greyscale erosion and dilation operations. It was shown that the computation time is independent of the structuring element size, and varies linearly with the number of pixels in the input image. The method was implemented for flat as well as sloping and peaked structuring elements. The sloping version of the histogram method is slower than the flat version and the peaked version takes twice as long. Using the peaked version allows non-flat structuring elements, but means that attention must be paid to the definitions of off-image pixels on dilation, and to the possible truncation of pixel values due to limitations in the image data structure used.

REFERENCES

[1] Shih, F.Y.C., Mitchell, O.R., “Decomposition of Grey Scale Morphological Structuring Elements”, Pattern Recognition, Vol. 24, No. 3, pp 195-203, 1991.

[2] Soille, P., Breen, E.J., Jones, R.J., “Recursive Implementation of Erosions and Dilations Along Discrete Lines at Arbitrary Angles”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 5, pp 562-567, 1996.

[3] van Herk, M., “A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels”, Pattern Recognition Letters, Vol. 13, pp 517-521, 1992.