EFFICIENT CONVOLUTION BASED ALGORITHMS FOR EROSION AND DILATION

M. Razaz and D.M.P. Hagyard

School of Information Systems

University of East Anglia

Norwich England

ABSTRACT

Morphological operations based on primitives such as dilation and erosion are slow to compute in practice especially for large structuring elements. For direct implementation of these primitives, the computing time grows exponentially with the size of the structuring element used. The latter renders these implementations impractical for large structuring elements due to a rapid increase in computation time. There have been attempts in the literature to develop fast algorithms for implementation of morphological primitive operations. These are mainly restricted to convex and often symmetric structuring element shapes. We have developed a fast convolution-based approach for implementing morphological erosion and dilation and other operations such as opening and closing based on these primitives. The major advantages of this approach are: (i) it can use any structuring element shape including non-convex cases and (ii) it is very fast. This paper briefly introduces the approach and presents timing results for dilation and erosion using three different implementations of the approach. These results are also compared against a direct (brute force) implementation of the primitives.

1. INTRODUCTION

Fast implementation of morphological primitive operations such as dilation and erosion is crucial for their success in image processing applications. Direct implementation of these primitives is a brute force approach and is computationally very slow, the computing time grows exponentially with the size of the structuring element. Previously we presented a fast morphological transform (FMT) method for efficient computation of binary primitives [7]. The major significance of FMT is that its computing time is independent of the structuring element size. The restriction of FMT and other fast implementations in the literature, see for example [5,6,9], is that the structuring elements used are usually confined to convex, and mostly symmetric shapes and certain types of lozenge geometries. In this paper we present a fast convolution based approach for implementing these primitives which overcomes the restriction and can use any arbitrary structuring element.

We first represent dilation and erosion primitives in terms of a thresholded convolution operation. These primitives are then implemented using a fast circular convolution algorithm, which works very efficiently for large structuring elements when the images and structuring elements are of similar sizes. However in practice a structuring element is usually much smaller than the image itself, and therefore the fast circular convolution algorithm would be wasteful and inefficient to use. To overcome these difficulties, we have developed two fast dilation and erosion algorithms using spatial domain *overlap-save* and *overlap-add* convolution methods. Timing results for dilation and erosions for these algorithms are presented and discussed.

2. DILATION AND EROSION AS CONVOLUTION

We now briefly describe how dilation and erosion can be expressed in terms of a thresholded convolution. For simplicity the analysis is presented in one dimension but can easily be extended to multidimensions.

Let us define a nonlinear threshold function $h_t(.)$ to be:

$$h_\tau(x) = \begin{cases} 0, x \le \tau \\ 1, x > \tau \end{cases}$$

Let the dilation of the function f(x) by the structuring element $s(x)$ [1,3] be given by

$$[f \oplus s](x) = \bigcup_{\{y \in Z^n : s(y)=1\}} f(x-y)$$

The linear convolution of $f_1 = f(x)$ and $f_2 = s(x)$ on the other hand is given by

$$[f_1 * f_2](x) = \sum_{y \in Z_n} [f_1(x-y) \bullet f_2(y)]$$

The convolution kernel can now be set up such that the convolution integral will be 0 only when all the members of the convolution kernel are over zero (or white) pixels in the image. If any of the convolution kernel members are over non-zero (or black) pixels of the image then the values will multiply to more than zero and the result of the

convolution must therefore be greater than zero. By thresholding the convolution value using $h_t(.)$ such that every non-zero value is written as black, the combination has the same result as the "hit" operation described in [1] i.e.:

$$[f \oplus s](x) = h_\tau([f * s](x)),$$

$$0 < \tau < 1$$

The erosion operation can also be expressed in terms of thresholded convolution by using duality property:

$$[f \Theta s](x) = (h_\tau[f^c * s](x))^c$$

Thus erosion can be performed by inverting the image in the spatial domain, performing a dilation and then inverting the resultant image back. This is equivalent to performing a dilation on the background of the input image.

3. IMPLEMENTATION

We can see by examining the expressions for the morphological dilation or erosion that the most time consuming operation is the convolution. The thresholding is straight forward to perform and operates in O(n) time where n is the number of pixels in the image. We have implemented the dilation and erosion operations using a fast circular convolution algorithm. This involves performing an FFT on the image and the kernel (structuring element) to convert both into the frequency domain. This is followed by direct frequency domain multiplication, which is equivalent to a circular convolution in the spatial domain. To perform the multiplication correctly, the image and structuring element must be of the same size. If the structuring element is smaller than the image then it has to be padded up with zeros. Once the multiplication has been performed, the result is inverse Fourier transformed into the spatial domain. The image is thresholded and then output to a file.

There are a number of FFT algorithms that can be used for implementation of the fast convolution, most of which have limitations on the sizes of image they can operate on. The particular FFT algorithm used for this implementation was a variation of the mixed-radix FFT transform algorithm [8]. This algorithm, while not the fastest, is capable of processing a wide variety of different sized images. It operates by breaking the width and height of the images down into their prime factors to arrange the image into groups of pixels to operate on. The largest prime factor that can be used is 19. Any image which has a height or width with a prime factor greater than 19 or that is odd, must be increased in size until they are even with prime factors less than 19. If an image has to have its dimensions increased the existing data is written into the top left-hand corner of the area. This occurs when an image is increased to allow its dimensions to be accepted by the FFT, and to bring the structuring element image up to the same size as the image being used by the FFT algorithm.

The mixed-radix FFT algorithm should in theory show approximately logarithmic behaviour [4]. However we found that the operation of the algorithm to accommodate the different sizes of input image introduces a variation in the computing time depending on what prime factors were found in the size of the image. Therefore the size of input image was modified so as to give a minimum FFT computing time.

After the complex multiplication of the images and the inverse transformation of the result back into the space domain, the image is thresholded. This occurs in the same step as copying the image from the floating point array into the byte array. In the byte array off (or white) pixels are represented by 0, and on (or black) pixels are represented by 255. These values are transferred to the floating point array. For thresholding operation, the threshold is set at 125. This value is unimportant as long as values of 0 are represented as white. When performing an erosion the image value is reversed after the thresholding. For erosion, the edges of the output image, where the structuring element would have been partially off the image, are ignored since they contain incorrect information due to the off image areas being read as white.

In the fast circular convolution method just described, the kernel (structuring element) is padded up to the size of the output image for dilation or to the size of the input image for erosion. However when the kernel is significantly smaller than the image as is the case for most practical morphological image processing applications, then the above circular convolution becomes inefficient due to the wasteful zero padding involved.

To overcome this problem we developed the *overlap-add* and *overlap-save* algorithms, that speed up computation time and allow the convolution of a large image with a smaller kernel to be performed [2] without having to pad up the kernel. The two algorithms are similar and are performed in the spatial domain. The input image is subdivided into N segments (slices) of the kernel, the convolution is evaluated for each segment, and then all the separate convolution results are recombined in the spatial domain . In between these two operations any convolution algorithm can be used. In this case, our fast convolution is used, employing the multiplication of two equal sized Fourier transforms in the frequency domain. Multiplication of two discrete Fourier transforms is equivalent to a circular convolution in which information that would be written off the end of the sample due to the 'spreading' effect of the convolution will 'wrap-around' and appear at the beginning of the sample. The management of these wrap-around errors is performed by the overlap-add and overlap-save algorithms in a slightly different manner. The overlap-add algorithm pads the segment of the image so that the spreading of the image under convolution does not fall off the end of the sample and wrap-around on the output. The overlap-save algorithm uses overlapping segments of the image and discards those parts of the output image that contain wrap-around errors. For both algorithms, the segment size chosen is important, and must be at least twice as large as the size of the convolution kernel used. Increasing the segment size will reduce the number of segments needed for the image, and therefore the number of overlap pixels processed is doubled. Reducing the size of segments reduces the overlap at the end of the image. The size can be chosen to minimise the number of pixels processed. We now briefly describe the overlap- save algorithm in the following section. The overlap-add algorithm follows a similar procedure except that segments are padded as explained above.

3.1 Overlap-Save Algorithm

Consider an input image array x(n) and a kernel (structuring element) h(q) containing Q values. The overlap-save algorithm convolves x(n) and h(q) in the spatial domain. The image is split into slices, each N pixels long. The value of N is somewhat arbitrary, but as wrap-around errors extend for Q-1 pixels into the output of each slice, so N should be at least equal to 2Q. The main structure of the algorithm is as follows:

(i) Perform an N-point FFT on h(q) to generate complex H(k). Since N is greater than Q it will be necessary to pad h(q) with zeros.

(ii) Select N pixels from the input array, x(n). Each successive slice of the array should overlap the previous selection by Q-1 pixels. Thus each slice consists of Q-1 pixels from the previous slice and N-(Q-1) new pixels. This slice, $x_m(n)$, has an N-point FFT performed on it to create $X_m(k)$.

(iii) Multiply H(k) by $X_m(k)$ to generate $H(k)X_m(k)$.

(iv) Perform an N-point IFFT on $H(k)X_m(k)$ to find $y_m(r)$ in the space domain.

(v) Discard the first Q-1 points of $y_m(r)$ and write the following N-(Q-1) pixels into the output.

(vi) Goto (ii) and process the next slice.

Note that at each iteration of the loop N-(Q-1) new pixels are processed, and that the FFT of the convolution kernel has to be performed only once.

4. EXPERIMENTAL RESULTS

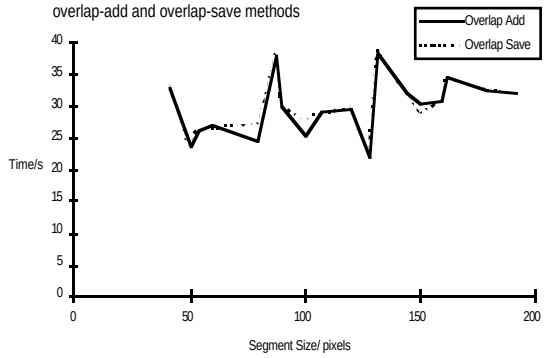
Different experiments were carried out to test the correctness and timing of the algorithms developed. For the overlap-add and overlap-save algorithms the optimum segment size was chosen by calculating the number of pixels that would be input to the FFT algorithm and multiplying that value by the time per pixel to perform the convolution. The segment size to give the smallest estimated time was calculated from Graph 1.

The timing results were produced using a series of octagonal structuring elements ranging from 9x9 pixels to 255x255 pixels. The image used for testing was a greyscale image, 864 by 864 pixels which had been thresholded to produce an approximately equal mix of black and white pixels.

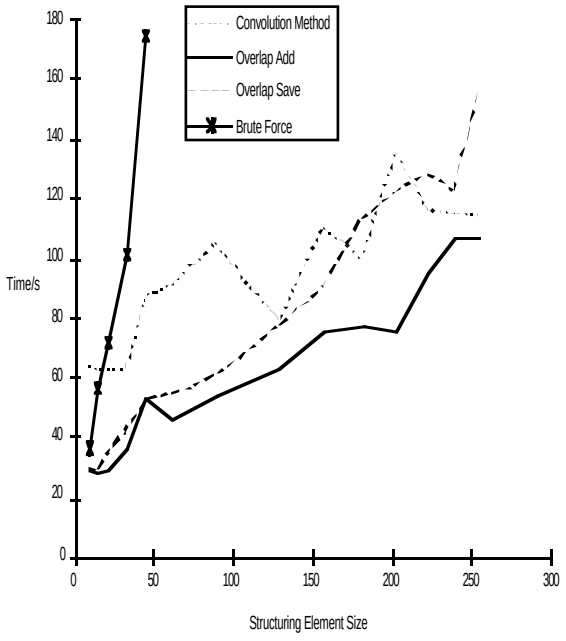
Graph 2 compares the results for dilation using the fast circular convolution, overlap-add, overlap-save and direct brute force algorithms. Both the overlap-add and overlap-save algorithms are superior to the convolution method, with the overlap-add being slightly faster than the overlap-save algorithm. The brute force algorithm is significantly slower than the rest as the computation time grows exponentially with the size of structuring element.

Graph 3 shows the timing results for erosion. The overlap-add and overlap-save algorithms, as can be seen, have a speed advantage over the convolution method, whilst the structuring element remains small compared to the image. Once the structuring element gets above approximately 25% of the size of the input image the direct convolution method is faster. Again as was the case for dilation, the overlap-add algorithm for erosion is faster than the overlap-save algorithm. Notice that the timing for the fast convolution method remains constant as the structuring element size increases. This is because the erosion does not increase or decrease the size of the image that has to be input to the FFT algorithm. The brute

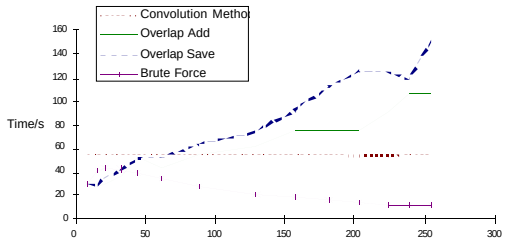
Graph 1. Graph of time to Perform a 5 by 5 dilation on a 864 by 864 image using different values for the size of segment used for the overlap-add and overlap-save methods

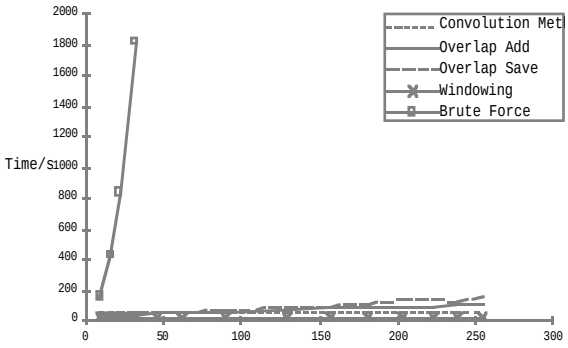


Graph 2. Graph of Time to dilate a 864 square imge with various sizes of octagonal structuring element



Graph 3. Graph of Time to erode an 864 square image using various sizes of octagonal structuring element





Graph 4 Graph of Time to erode a 864 black square i Fast Morphology methods with various sizes of octago element

force algorithm is faster than other algorithms for larger structuring elements. This is purely due to an optimisation procedure implemented in the algorithm that allows the brute force to stop scanning through the list of offsets for the structuring element, once an "off" pixel has been located in the neighbourhood covered by the structuring element. For our test image with an even spread of black and white pixels this means that 50% of output pixels required only one pixel test to be shown to be blank. This situation can be shown to be the reverse if majority of pixels are black as graph 4 shows..

5. CONCLUSIONS

A versatile approach was discussed for fast computing morphological dilation and erosion based on the concept of thresholded convolution. The major advantage of this approach is that it can deal with any arbitrary shaped structuring element. Three different implementations of this approach for erosion and dilation were presented, namely the fast circular convolution, overlap-add and overlap-save algorithms. Typical results for erosion and dilation were discussed and compared against the direct brute force method. When the structuring element used is significantly smaller than the input image then both overlap-add and overlap-save algorithms are the fastest with the former being slightly faster than the latter. Once the structuring element is comparable to the size of the input image (approximately 25% of the image size) then the overhead associated with these two techniques makes them less efficient and the circular convolution is the fastest method to use.

6. REFERENCES

[1] J. Serra, “Introduction to Mathematical Morphology”, Computer Vision, Graphics and Image Processing, Vol. 35, Pages 283-385,1986.

[2] D. J. DeFatta, J.G. Lucas, W.S. Hodgkiss, "Digital Signal Processing: A System Design Approach", pp 305-315, Wiley, 1988.

[3] B. Kisacanin, C. Schonfeld, “A Fast Thresholded Linear Convolution Representation of Morphological Operations”, IEEE Transactions on Image Processing, Vol. 3, No. 4, Pages 455-457, 1994.

[4] A. Peled, B. Liu, “Digital Signal Processing. Theory, Design and Implementation”, Wiley, 1976.

[5] 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.

[6] M. Razaz and Hagyard D.M.P. "Morphological Structuring Element Decomposition: Implementation and Comparison", Signal Processing VIII :Theories & Applications, Vol. 1, pp. 288-291, 1996.

[7] D.M.P. Hagyard, M. Razaz and P. Atkin, “A Fast algorithm for computing morphological image processing primitives”, Proc. IEEE Nonliner Signal & Image Processing, Sept. 1997.

[8] J.W. Cooley, J.W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series”, Math Comp., Vol. 19, Pages 297-301, 1965.

[9] M. Razaz and Hagyard D.M.P. "Structuring element decomposition by tree searching", Proc. IEEE Nonlinear Signal. & Image Processing, 1997.