MORPHOLOGICAL STRUCTURING ELEMENT DECOMPOSITION: IMPLEMENTATION AND COMPARISON

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

School of Information Systems, University of East Anglia, Norwich, U.K.

Email: mr@sys.uea.ac.uk, dmph@sys.uea.ac.uk, phil.atkin@synoptics.co.uk

* Synoptics Ltd., 271 Cambridge Science Park, Cambridge, U.K.

ABSTRACT

Structuring element decomposition is used to reduce computation time in performing morphological image processing operations by breaking down a structuring element into simpler components. This paper classifies decomposition algorithms into two broad categories, namely morphological combination and set theoretic combination classes. Two important structuring element decomposition methods, the tree search and arbitrary shape decomposition algorithms, are discussed and their performances are compared using a series of different structuring element shapes. We found that the tree search decomposition algorithm is restricted to mainly symmetric and convex structuring elements, and its computation time for performing a morphological operation grows exponentially with the size of the element used, whereas the arbitrary shape decomposition algorithm performs the same operation in linear time, and can deal with any structuring element shape.

1 INTRODUCTION

There are a variety of structuring element decomposition (SED) methods in the literature, see for example [1-6]. Generally, SED algorithms can be broadly divided into two classes according to the manner in which their resulting series of structuring elements are applied to the subject image. The first class combines the output decomposed components morphologically and the second class does the same but set theoretically. The morphological combination class relies on the chain rule's applicability to erosion and dilation. This implies that if a structuring element (SE) can be generated by dilating a single point at the origin consecutively by a series of shapes, then applying the same series of operations to an image will have the same effect as applying the original SE.

In the set theoretic combination class, addition or dilation can be represented as an ORing together of a series of translates of the subject image, where each translate represents a member of, or a pixel in, the structuring element set. Similarly, subtraction or erosion can be represented as an ANDing together of a series of translates of the image. The structuring element is represented by a series of decomposed elements that can be ORed together to produce the original element. This set theoretic decomposition will not reduce the number of pixels required to perform the operation, and it could even require more pixels due to overlapping members of the decomposition. The approach, however, can be used to simplify the set of shapes that must be applied to the image structuring elements. It can be tailored to breaking the SEs down into simple shapes which are easily decomposable by another algorithm or shapes for which an improved implementation is available. This is how this set theoretic decomposition produces a performance improvement.

Two important algorithms, one from each class, based on tree search decomposition and arbitrary shape decomposition are discussed and implemented. Experimental results are presented and compared with a brute force approach, which implements directly a morphological operation without using any decomposition.

2 TREE SEARCH DECOMPOSITION

The tree search algorithm (TSA), which belongs to the morphological combination class, is based on the work reported in [2]. In this algorithm, an element S is decomposed into a series of shift and OR operations which are equivalent to dilating by SEs containing two members, one at the origin and the other at the location that defines the translation. The decomposition of S is found by a combinatorial search process that constructs a tree of possible shift

and add operations from a start node. The search is recursive. There is a partial decomposition at each node, and the child nodes add one more translation such that the result of morphologically combining the members of the decomposition still remains within the confines of the SE to be decomposed. A forward checking mechanism is applied to improve the efficiency of the algorithm by identifying translations that cannot produce valid children and thus excluding them from the search.

3 ARBITRARY SHAPE DECOMPOSITION

The arbitrary shape decomposition (ASD) algorithm, which belongs to the set theoretic combination class, is based on and extends the work reported in [5]. This algorithm decomposes an structuring element S into a series of basis shapes which, when appropriately scaled and translated, can be ORed together to produce the original S. Each element of the decomposition is a triple holding the basis shape (square, diamond or octagon), its size and location. A list of triples is generated by finding those triples which have maximal disks, each disk being one of the basis shapes, within the boundaries of the SE, S. Once an initial list of triples has been generated, the algorithm creates an optimal list of triples by successively removing the best one from the list and adding it to the optimal list. The best triple is the one remaining in the list which covers up the most new pixels whilst taking the least time to implement. The process ends when no triple can be found which is more efficient than using the brute force method for the new pixels covered. Once the list of triples is generated, it is applied to the image. This requires the implementation of procedures to perform the morphological primitives on an image using a series of triples and then combining the translated results of each operation.

4 RESULTS AND DISCUSSION

The tree search decomposition algorithm was implemented and tested using a series of structuring element shapes. Despite its theoretical elegance, we found that the completion time for this algorithm increased exponentially with the size of the structuring element. This was because the forward checking algorithm was unable to prevent the number of nodes in the tree rising exponentially with the number of pixels in the structuring element. Also the SE shapes that could be decomposed generally had to be symmetric and convex. A limited group of more complex shapes could be decomposed, where the shapes were constructed by successively shifting and ORing the current shape onto itself. The speed of performing morphological operations using the decompositions was considerably improved over the use of undecomposed SEs. The times to perform morphological operations were found to be generally proportional to the number of shift and add operations, which is approximately log2 of the lengths of the edges of half the structuring element. These results are shown in Figures 1 and 2.

The arbitrary shape decomposition algorithm was implemented using a simplified method of identifying the initial list of triples. The median positions within the structuring element were identified in the four scanning directions and a distance transform was used to find the largest size of each shape that could be placed in the location. The approach we used to optimise the list of triples is similar to that in [5]. The implementation of the morphological operations was performed using logarithmic decomposition as well as our own fast morphological transform (FMT) algorithm [7]. FMT is a windowing algorithm and allows 1D erosion and dilation operations to be performed in constant time with respect to the structuring element width. FMT can be applied in 4 directions allowing very fast implementation of squares, diamonds and octagons.

We found that ASD algorithm operates in linear time with respect to the number of pixels in the structuring element, and it could decompose any SE shape. This algorithm is considerably faster than TSA and is also more flexible in dealing with different shapes. One slight problem with ASD algorithm was that all the triples had to have a centre pixel, and therefore had to be an odd number of pixels wide and tall. The CPU times taken to perform dilation were variable as shown in Figure 4, although the windowing and logarithmic methods produced average speedups of two to three times over the brute force method for a selection of 30 by 30 pixel SEs. The CPU times taken to perform erosions were found to be more variable (see Figure 5), including several decompositions for which the brute force method was faster. These results were obtained on images with approximately even numbers of black and white pixels. On

completely black images we found that erosion by the brute force method was considerably slower than the two decompositions, frequently by a factor of 4 to 8. The speedup produced by ASD algorithm was maximised when the decomposition included a few large triples as opposed to a large number of small triples. We found that the structuring elements shapes 2, 4, 5, 9 and 10 (see Figure 3) produce the most efficient decompositions as shown in Figures 4 and 5.

Figure 1. Time taken to decompose a square structuring element by the tree search algorithm.

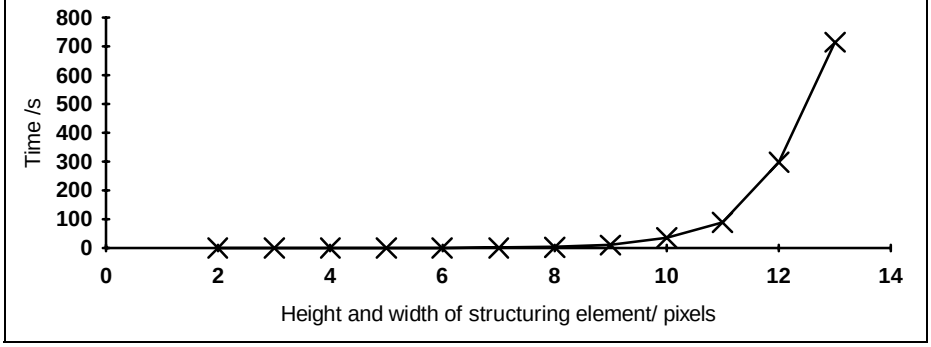


Figure 2. Dilation of a 256 by 256 image using versions of the structuring elements decomposed by the tree search algorithm.

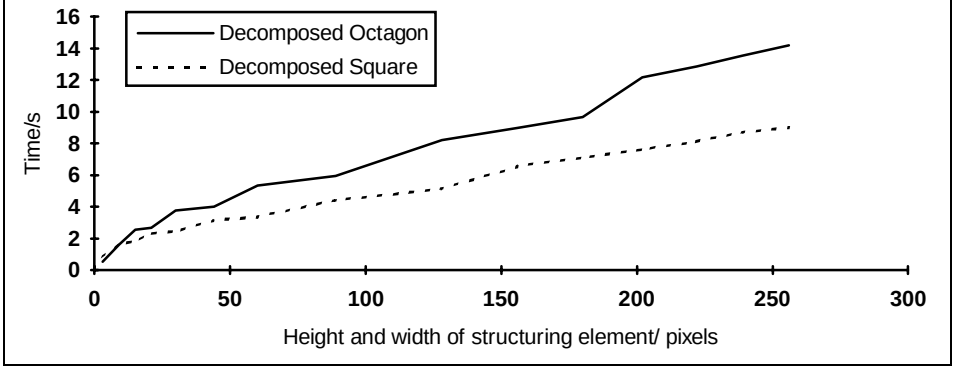


Figure 3. The structuring elements used for the tests on the image, each structuring element bitmap is 30 pixels square.

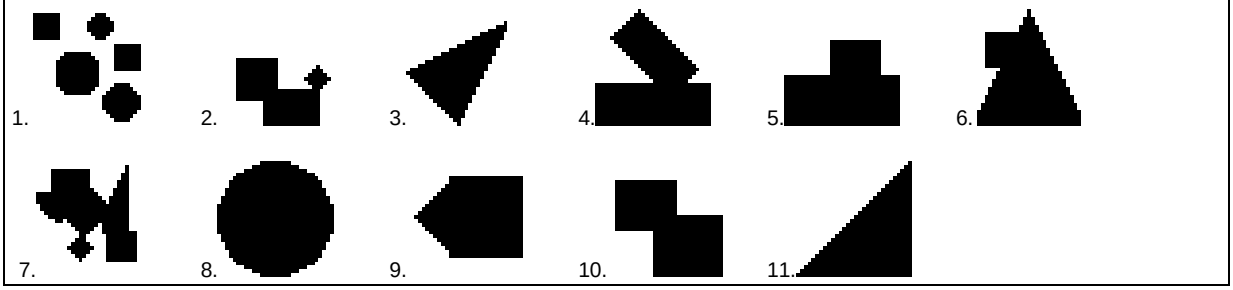


Figure 4. Time to perform dilation on 256 by 256 binary image using the arbitrary shape decomposition algorithm.

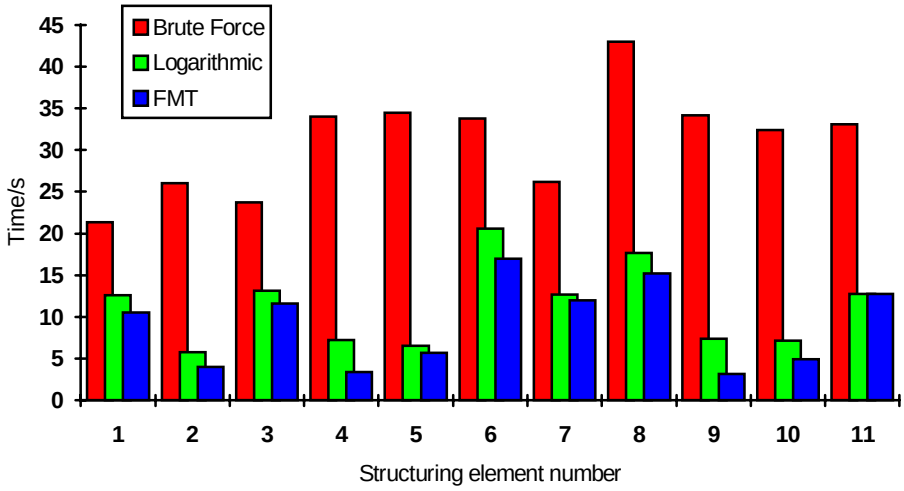
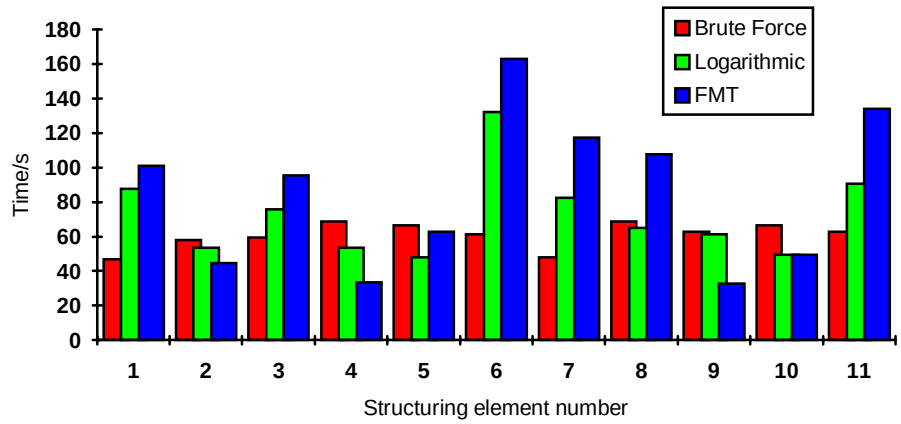


Figure 5. Graph of time to erode using the decompositions on an 864 pixel square image using the arbitrary shape decomposition algorithm .



References

[1] J. Pecht, Pattern Recognition Lett., Vol. 3, pp.113-117, 1985

[2] X. Zhuang and R. M. Haralick, Computer Vision, Graphics and Image Processing (CVGIP), Vol. 35, pp 370-382, 1986.

[3] I. Pitas et al, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Vol. 12, pp 38-45, 1990.

[4] J. Xu, IEEE Transactions on PAMI, Vol. 13, pp 153-162, 1991.

[5] R. van den Boomgaard and R. van Balden, CVGIP, Vol. 54, pp. 252-258, 1992.

[6] H. Park and R.T. Chin, IEEE Transactions on PAMI, Vol. 17, pp.2-15, 1995

[7] D.M.P. Hagyard, M. Razaz and P. Atkin "A fast algorithm for computing morphological image processing primitives", Tech. Rept., School of Info. Systems, UEA, Jan. 1996.