An Efficient Strategy for Implementing Iterative Area Openings Using the Max Tree

Xiaoqiang Huang, Mark Fisher, Yanong Zhu, Richard Aldridge, Dan Smith,
School of Computing Sciences, University of East Anglia
Norwich, United Kingdom, NR4 7TJ
Email: xq.huang@uea.ac.uk, {mhf, yz, rva, djs}@cmp.uea.ac.uk

Abstract

Area opening is an important morphological connected set operator that features in removing upper level sets from an image whose area properties are smaller than a threshold λ. Existing algorithms found in the literature that implement the area opening operator are based on either priority queues, the max tree or the union-find approach. In this paper we explore the advantages of using the max tree based approach for iterative area opening. Iteratively applying an area opening is the central idea underpinning all scale based image decompositions. An efficient implementation strategy for iterative area opening is therefore very important if scale based image processing algorithms are to be successfully applied in real time computer vision applications. This paper builds on recently published work comparing approaches for implementing area openings, and improves on the method proposed for image reconstruction via a max tree. Experimental results are presented that show the new approach proposed in this paper obtains a performance gain of 25% with images of reasonable big size (320×256).

1 Introduction

Connected set operators [9, 8, 1] are fundamental to many non-linear morphological image filters having a number of desirable properties, most importantly preservation of shape or the scale casualty [4]. A connected set operator only deals with flat zones (largest connected set of the space where the image grey-level is constant) in the image. When a connected operator is applied to an image, each flat zone will either be merged to one of its neighbouring flat zones or be left intact. As this does not introduce any new contour, it simplifies the image as well as preserving the scale casuality.

Area opening [11] (area closing as its dual operator) is one such operator that features in removing connected upper level sets (brighter intensity image objects) whose area properties are smaller than a threshold λ. Examples of applying the area opening operator to both a binary and a gray scale image are shown in figure 1 and figure 2, respectively. Figure 1 illustrates the area opening operator removes only those flat zones whose area properties are smaller than seven pixels, leaving the other larger regions untouched. From a signal processing point of view, area opening and area closing operators form a pair of non-linear filters.

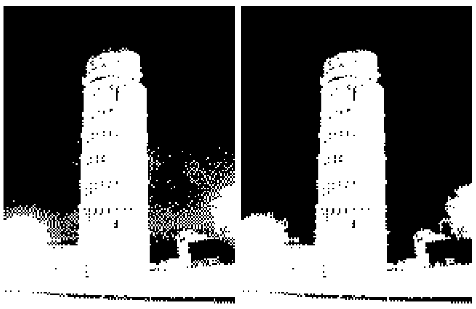
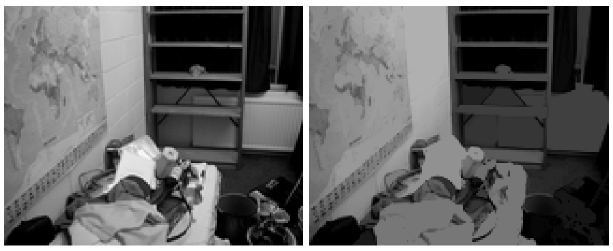

(a) the original image (b) the image after area opening

Figure 1. An example of binary area opening: image size = 144 $\times$ 192, λ = 7

Binary area opening is based on binary connected opening. Let the set $X \subseteq M$ denote a binary image with domain M. The binary connected opening $\Gamma_x(X)$ of X at point $x \in M$ yields the connected set of X containing x if $x \in X$ and $\emptyset$ otherwise. Thus Γ_x extracts the connected set to which x belongs, discarding all others.

The binary area opening can now be defined as:



(a) the original image (b) the image after area opening

Figure 2. An example of gray scale area opening: image size = 160 $\times$ 128, λ = 100

Definition 1. Let $X \subseteq M$ and $\lambda \geq 0$. The binary area opening of X with scale parameter λ is given by

$$\Gamma^a_\lambda(X) = \{x \in X | Area(\Gamma_x(X)) \geq \lambda\}. \quad (1)$$

The binary area closing can be defined by duality

$$\Phi^a_\lambda(X) = [\Gamma^a_\lambda(X^c)]^c. \quad (2)$$

The definition of an area opening of a gray-scale image f is usually derived from binary images $T_h(f)$ obtained by thresholding f at h. These are defined as

$$T_h(f) = \{x \in M \mid f(x) \geq h\}. \quad (3)$$

Definition 2. The area opening for a mapping $f : M \to \bar{R}$ is given by

$$(\gamma^a_\lambda(f))(x) = sup\{h | x \in \Gamma^a_\lambda(T_h(f))\}. \quad (4)$$

The gray-scale area closing ϕ^a_λ is defined by using a duality relationship similar to equation (2).

$$\phi^a_\lambda(f) = -(\gamma^a_\lambda)(-f). \quad (5)$$

The above definitions of area opening and closing are both adopted by Vincent et. al. [11] and Meijster et. al. [5]. Although morphological opening and closing are usually applied in the context of level sets the concept has been extended and generalised by Breen et. al. [2] to operate on other image attributes including length, diameter, radius, perimeter and other shape features.

Existing algorithms found in the literature that implement the area opening operator are based on either priority queues [11], the max tree [7] or the union-find approach [10]. Recently, Meijster et. al. [5] published a comprehensive comparison of these three algorithms and concluded that the union-find approach outperformed the other two algorithms in terms of CPU execution time and memory efficiency. However, this conclusion was based on the assumption that only one area opening/closing (at predefined λ) needs to be performed. In another words, the threshold λ should be known a-priori. In most scale based image decompositions one is interested in applying the area opening operator iteratively at varying scale λ to the same image. The experiments reported in section 4 concludes that in this case the method of area opening used by the max tree is a better option. This is because no matter how many times one wants to apply the area opening operator to the same image , the tree only needs to be constructed once. However, with the other two approaches area openings need to be re-computed whenever λ changes.

This paper discusses several issues relating to how to implement more efficiently iterative area openings via the max tree. The paper is organised as follows: Section 2 briefly describes the max tree and introduces an efficient way of building a max tree from an image. Section 3 describes two different strategies for iterative area openings in the context of a max tree and compares these two strategies theoretically. Experimental results, shown in section 4, confirm that the strategy adopted in this paper is more efficient than that proposed by other researchers. Conclusions are drawn in the final section.

2 The Max Tree

The max tree proposed by Salembier et. al. [7] is a multiscale image representation formed by considering a hierarchy of connected upper level sets of pixels in an image. This representation was developed to deal with classical anti-extensive connected operators (such as area opening), as well a new ones, in an efficient manner. The max tree has been applied to image filtering, segmentation, tracking and information retrieval [6].

To understand the tree, the image is considered to be a 3D relief. The nodes of the tree represent the connected upper level sets for all possible gray-level values. The leaves of the tree correspond to the regional maxima of the image. The links between the nodes describe the inclusion (father-child) relationship of the connected sets. An example of a max tree created from a simple image is shown in figure 3 [1].

Three sub-problems need to be addressed before the max tree can be successfully constructed:

A. Find all possible tree nodes from the original image

B. Create the father-child relationships (links) between each possible pair of tree nodes

C. Create an efficient data structure to store information (region attributes) at nodes in the max tree.

[1]This figure is reproduced from [6]

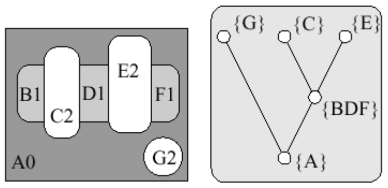


Figure 3. left: a simple image only with seven flat zones, right: the max tree for the image on the left

The first sub-problem has been successfully addressed by a recursive flooding algorithm proposed in [6], however a number of approaches to the remaining sub-problems have been described. The max tree built by Meijster et. al. adopted an array as the underlying data structure for the tree. The advantage of using the array is that it provides direct access to its elements i.e. the complexity of visiting a random element in an array is $O(1)$. However, there are two major problems in using an array as the underlying data structure. Because the size of a max tree is unknown before the tree is built using a static array is not memory efficient. For example, Meijster el. al. used the image size (number of pixels) to parameterise the length of their array. This approach is very wasteful of memory as in most cases the number of max tree nodes is far less than the number of pixels in the image. Secondly, when a node is deleted from the tree its corresponding element must also be removed from the array. Maintaining the array structure can become very cumbersome.

A more efficient data structure using a linked list associated with a hash table has recently been proposed by the authors of this paper [3]. The linked list provides dynamic allocation of computer memory thus the memory allocated for the max tree is just enough. However, the nodes in a linked list must be accessed serially so the complexity of visiting a random node in a linked list is $O(N)$. To overcome this problem a hash table is built immediately after the recursive flooding step. Using this table, the elements in the linked list are accessed directly so that the complexity is reduced to $O(1)$. The linked list structure also allows more flexible management of the tree nodes, they can be easily deleted and the memory recovered. Further details and experimental results comparing the efficiency of the technique to other published implementations may be found in [3].

Once the underlying data structure of the max tree has been decided, one needs to consider what information should be stored at a tree node. A minimal set of attributes should include

A. ID: node identification number

B. father's ID

C. a list of children IDs

D. a list of pixel coordinates

E. area: number of pixels belonging to a node

Obviously, each node in a tree needs a unique identification number. As a tree is a hierarchical data structure, each node is linked to its father and children nodes. The area attribute records the total number of pixels belonging to a node (i.e. the size of a granule or region). A node would also need to store information of pixels belonging to its support region so that an image can be reconstructed from a tree. Note that the list of children's ID and the list of pixel coordinates are also implemented by the linked list.

During the recursive flooding step of the creation of the max tree, Salembier et. al. use a number of arrays to store information important in later steps. The remainder of this section provides a brief overview of these attributes, however we refer the reder to [7] and [6] for a more complete description.

Salembier et. al. has proposed a very efficient way of finding scale tree nodes using a recursive flooding algorithm. The flooding algorithm begins at the root node of the tree (this is, the lowest gray value of the image) and recursively constructs each of the branches of the tree. They use hierarchical first-in-first-out queues of NG levels, with NG the possible number of gray levels (usually $NG = 255$). These queues are used to define the scanning and processing order of pixels comprising the image. An array $STATUS$ of the same size as the image is used to determine to which node a pixel belongs to. A pixel p with gray-level h belongs to the node C_h^k if $STATUS[p] = k$. Initially, all elements in $STATUS$ are set to $NOTPROCESSED(< 0)$. One array of G integers called $Number - Nodes(h)$ is used to store the number of max tree nodes detected so far at each gray level h. The values of Number-Nodes is updated whenever new nodes are created at gray-level h. A further array of G Boolean $Node - at - Level(h)$ stores at which levels below the current gray-level nodes have been detected in the path from current node to the root.

A node in a max tree is presented uniquely by C_h^k. For instance, a node presented by C_{20}^3 means this node's gray value is 20 and it is the third node found in the level of 20 during the recursive flooding algorithm. Note that a node C_h^k corresponds to a connected component P_h^k. However, C_h^k contains only those pixels in P_h^k which have gray-level h for the purpose of memory efficiency. In the next section, we proceed to explain why and how efficient iterative area openings can be achieve using the max tree.

3 Iterative Area Opening via the Max Tree

Both Salember [7] and Meijster [5] claimed that once a max tree of an image has been constructed, computing an area opening at scale λ reduces to removing all nodes representing regions which have an area smaller than λ from the tree, and then reconstructing the image at the new scale.

The area opening operator is often applied as a non-linear filter for preprocessing an image. In most cases, one is interested in applying an area opening operator at different scales λ to the same image. In this case, it is beneficial to retain the max tree structure unchanged after computing an area opening operation. Thus, once a max tree of an image has been constructed, computing an area opening reduces to analysing the max tree nodes and reconstructing the opened image only by those tree nodes which have an area greater than or equivalent to λ.

The strategy proposed by Meijster et. al. to reconstruct the opened image is described as follows. Here, as stated earlier in section 2, a node C_h^k corresponds to a connected component P_h^k. However, C_h^k contains only those pixels in P_h^k which have gray-level h. For each node, Meijster et. al. check whether its area is smaller than λ. If so, its output gray level is set to that of its parent (which has already been assigned the correct gray). The output image O is made by visiting all pixels in the image, determining its node from the input image $I[p]$ and $STATUS[p]$, and assigning the output gray level of that node to $O[p]$. Their strategy actually consist of two steps. First, they assign the correct value for the output gray level of each node according to their area properties. Second, they assign each pixel the correct gray value according to the node the pixel belongs to. These two steps are processed serially.

We use different strategy to reconstruct the output image. The area property of a node is always smaller than that of its father (except the root node as it does not have a father node). Thus if a node is determined to have a smaller area value than the threshold λ, there is no need to check its descendant nodes at all. Therefore, we start checking the root node and proceed to recursively check each of its children nodes. During each check, the pixel in the output image O is assigned the gray level value of the node it belongs to as long as the area property of the current node is smaller the λ. When a node whose area property is greater than the λ is found, the current checking stops and all the pixels belonging to this node and its descendant nodes are output directly into the reconstruction image with the gray level value of the father node of the current node. With the help of the father-child relationships embedded in the tree, the descendant nodes of the current node can easily be visited in a recursive way.

The main differences between our strategy and that of Meijster et. al. 's is firstly not all max tree nodes are checked and secondly the checking and the reconstruction of the output image are done in a parallel way. Also the strategy is more memory efficient. Both strategies are data structure independent. In another words, both the strategies could work either using the array [5] or using the linked list combined with hash table [3] as the underlying data structures.

4 Experiments

Both the above strategies are evaluated in our experiments. The max tree is implemented by combining the linked list and the hash table as the underlying data structure [3]. The max tree and the iterative area opening operator are implemented in C++ using Microsoft Visual C++ IDE (version 6). The specification of the computer in which all experiments are carried out is as follows: OS: Windows 2000, CPU: Pentium 1.7GHZ, Memory: DDR 1.0G Bytes. The multi-resolution test images used in our experiments are shown in figure 4. Here, image 5 in figure 4 is the original image and the others are the resized versions of it. The time costs associated with area openings of these images that are reported as mean values found by repeating the experiment 100 times.

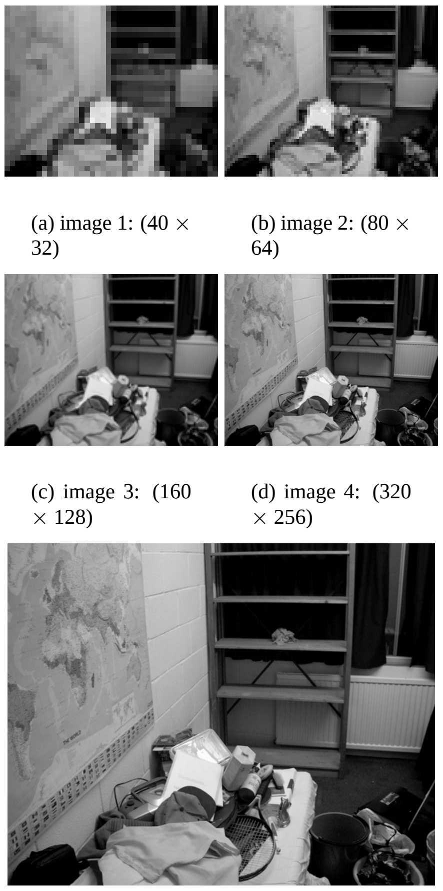

(a) image 1: (40 × 32) (b) image 2: (80 × 64) (c) image 3: (160 × 128) (d) image 4: (320 × 256) (e) image 5: (640 × 512)

Figure 4. The first group of our test images

Three sets of experiments have been conducted. The aim of the first experiment is firstly to compare the performance of the two filtering strategies described in section 3 and secondly to find out if the iterative area opening via the max tree is scale independent. Image 4 in figure 4 is chosen for this experiment as its size ($320 \times 256 = 89920$) is reasonably large. The time cost for building the max tree and then computing an area opening ($\lambda = 100$) is 265 milli-seconds and 12.82 milli-seconds respectively (approximately 4 milli-seconds faster than that (16.7) by the strategy from Meijster et. al).

For a more comprehensive evaluation of the two strategies all the possible values for λ (from 1 to 81920) are computed (in steps of 10). For each of the 8191 different values of λ, area openings are found using both strategies and 9 time cost averages computed from blocks of 1000 results (the final group contains 191 samples). The nine values from each strategy are plotted in figure 5.

Each point in the plot stands for the average time cost of applying the area opening to the image with a limited range of λ. The range is defined between the values of a pair vertical lines, one is the line that the point just sits on, another is the nearest line to the left side of the first line. The mean time cost is shown in the Y axis of figure 5.

These experiments clearly show the strategy for implementing the iterative area opening reported in this paper runs about 25 percent faster than that from Meijster et. al. This is because the Meijister's strategy requires extra work to assign the correct output gray level to a pixel in the output image. In the new strategy the checking and reconstruction of the output image are done in a parallel way. In addition, we used a recursive approach to identify the descendant nodes of the current node, which is also very efficient. The iterative area opening implemented by both strategies is *almost* scale independent. However, there is a small (almost unnoticeable) variation of the time cost using either of the two strategies (see figure 5. Investigating the underlying reasons for these variations is part of our further work.

The comparison work done by Meijster et. al. showed that the union-find approach is the fastest algorithm to implement a one-off area opening and runs no more than two times faster than the max tree based approach (using their filtering strategy) for real images. However, when a max tree is built up, the time cost of recovering an area opening using the tree is almost negligible. For example it costs 265 milli-seconds to build the max tree for image 4 and the average time cost of recovering an area opening to image 4 takes no more than 13 mil-seconds. Thus we conclude that the max tree approach outperforms the union-find approach is the case of iterative area opening.

The aim of the second experiment is to see, if the value of λ is kept the same, how the time cost varies when the image size changes. Table 1 showed the results of our second experiment using the test images from figure 4 and λ set

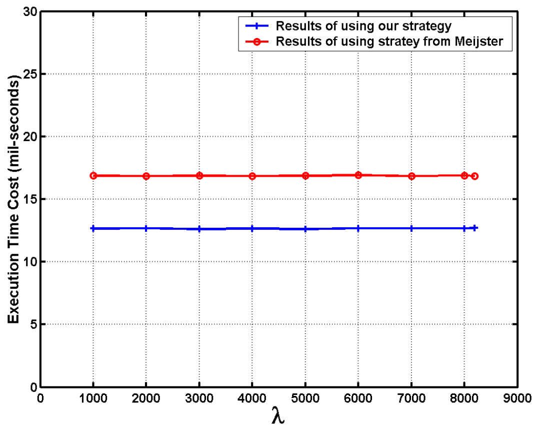


Figure 5. Our first experimental results

equal to 100. The results clearly show that both the strategies are not linear in terms of image size. However, our strategy is more linear than that from Meijster. This means that the performance gain of our strategy compared to that from Meijster et. al. should increase against image size. For instance, 25% performance gain is achieved with image 4 and 31% with image 5 using our filtering strategy (see table 1). Investigating the non-linearity is also part of our future work.

Strategy	Image Number				
	1	2	3	4	5
Ours	N/A	N/A	2.97	12.82	59.37
Meijster's	N/A	N/A	3.75	16.70	86.40

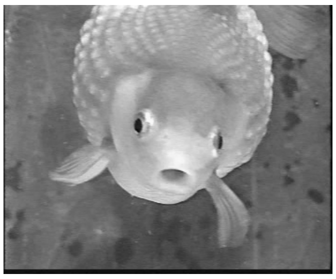
Table 1. Time cost (in term of milli-seconds) of applying area opening with $\lambda = 100$ to all images from figure 4.

More images (see figure 6) have been evaluated with different value of λ. The results also show that our strategy runs about 25% faster than Meijster et. al's strategy.

5 Conclusions and Further Work

In this paper, the term 'iterative area opening' has been developed to describe the operation that is at the heart of many morphological scale based decompositions. Based on our experiments, we proposed a new strategy for reconstructing the image at scale λ and compared this to that recently published by Meijster. The experiments show that our strategy runs about 25 percent fast than that from Meijster with images of reasonable big size.

Fast implementation of iterative area opening is impor-


(a) image 1: (352 × 288)

(b) image 2: (384 × 256)

(c) image 3: (320 × 240)

Figure 6. The second group of our test images

tant for a number of real time application in the field of computer vision. These might include non-linear image filtering, image segmentation, and object based image coding.

So far only a small number of image sets have been tested in our experiments. We are planing to use more synthetic and real images of different complexities for our further experiments. Investigating the underlying reasons for those small variations in figure 5 and the non-linearity in table 1 is also worthy of further work.

References

[1] J. Bangham, R. Harvey, and P. Ling. Morphological scale-space preserving transforms in many dimensions. *Electronic Imaging*, 5(3):283–299, 1996.

[2] E. J. Breen and R. Jones. Attribute openings, thinnings, and granulometries. *Computer Vision and Image Understanding*, 64(3):377–389, 1996.

[3] X. Huang, M. Fisher, and D. Smith. An efficient implementation of max tree with linked list and hash table. In *Proceedings of International Conference on Digital Image Computing-Techniques and Applications (Accepted)*, Macquarie University, Sydney, Australia, December 2003.

[4] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[5] A. Meijster and M. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):484–494, 2002.

[6] L. G. Ostermann. *Hierarchical Region Based Processing of Image and Video Sequences: Application to Filtering, Segmentation and Information Retrieval*. PhD thesis, Department of Signal Theory and Communications, Universitat Politecnica de Catalunya, Barcelona, Spain, April 2002.

[7] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4):555–570, 1998.

[8] P. Salembier and J. Serra. Flat zones filtering, connected operators and filters by reconstruction. *IEEE Transactions on Image Processing*, 3(8):1153–1160, 1995.

[9] J. Serra and P. Salembier. Connected operators and pyramids. In *Proceedings of SPIE Conference on Image Algebra and Mathematical Morphology*, volume 2030, pages 65–76, 1993.

[10] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22:215–225, 1975.

[11] L. Vincent. Grayscale area openings and closings: Their efficent implementation and applications. In *Proceedings of the International Workshop on Mathematical Morphology and its Applications to Signal Processing*, pages 22–27, 1993.