AN EFFICIENT ALGORITHM FOR ATTRIBUTE OPENINGS AND CLOSINGS

Jérôme Darbon[1,2] *and Ceyhun Burak Akgül*[2,3]

[1]EPITA Research and Development Laboratory (LRDE)
14-16, rue Voltaire F-94276 Le Kremlin Bicêtre, France

[2]Département Signal-Images, Ecole Nationale Supérieure des Télécommunications (ENST)
46, rue Barrault F-75643 Cedex 13 Paris, France

[3] Electrical and Electronics Engineering Department, Bogazici University
34342 Bebek, Istanbul, Turkey

{jerome.darbon, akgul}@enst.fr

ABSTRACT

In this paper, we present fast algorithms for area opening and closing on grayscale images. Salembier's max-tree based algorithm is one of the well known methods to perform area opening. It makes use of a special representation where each node in the tree stands for a flat region and the tree itself is oriented towards the maxima of the grayscale image. Pruning the tree with respect to some attribute, e.g., the area, boils down to attribute opening. Following the same approach, we propose an algorithm for area opening (closing) without building the max-tree (min-tree). Our algorithm exhibits considerable performance compared to the state-of-the art in this domain.

1. INTRODUCTION

In the domain of morphological image processing, connected operators have received considerable interest within the last decade especially after their theoretical foundations have been demonstrated in [11]. Their well known quality, namely simplifying an image without moving any of its contours [1] have been used in many applications such as image filtering [13], restoration and segmentation [4, 12]. Attribute openings (closings) are versatile filters in the sense that they can be used in diverse applications. In [1], the authors have demonstrated how to exploit area morphology in image classification. Sequence processing constitutes another niche of application for these filters [10]. Area opening (closing) [13] constitutes a benchmark instance of connected operators. In this paper, we present a new technique for its implementation.

An attribute opening can be realized by using a tree-based image representation where each node in the tree stands for a flat region or a connected component. The tree itself is oriented towards the maxima of the grayscale image and hence is called a max-tree [10]. Once the max-tree is constructed using a hierarchical queue data structure, in order to filter a grayscale image one needs to measure a specific attribute associated with the connected component of each node and to decide whether to keep the component or merge it to its father. Decision is based on comparing the value of the attribute against a prescribed threshold. By duality, an attribute closing can be performed similarly but with a slight modification: this time the tree is oriented towards the minima of the image and is called a min-tree. The filtering process for an attribute closing, i.e., successive attribute measurements and decisions, is identical to the case of an attribute opening.

Direct implementations of these filters that do not rely on trees are also possible [2, 8, 13]. In [13], Vincent introduced a priority-queue algorithm for area opening and closing. Later in [2], Breen and Jones extended Vincent's priority-queue algorithm to the general case of attribute openings and closings. In [8] which is also a good survey of connected set openings and closings, Meijster and Wilkinson presented the union-find approach in which case regional extrema are processed simultaneously in contrast to queue-based algorithms where the processing is performed sequentially.

In this paper, we present an implementation based again on the tree pruning strategy, using area as the attribute. Nevertheless, our algorithm does not build any tree in contrast to those that first explicitly build the tree and then prune it for filtering [5, 6, 7, 9, 10].

In that sense, the present work can be considered as a direct approach which proves to be considerably fast. The rest of the paper is organized as follows. In section 2, we describe our algorithm for area opening (closing). Section 3 is devoted to the experiments we carried on natural and synthetic images. We compare our area opening (closing) approach to the original tree-based algorithm [10] and to the union-find approach [8]. Finally in section 4, we discuss possible extensions and draw some conclusions.

2. ALGORITHMS FOR AREA OPENING AND CLOSING

In [10], Salembier presents a three-stage algorithm which consists of tree creation, filtering and image restitution stages. The crucial part of the algorithm is tree creation using a recursive flooding procedure. We briefly describe it in order to introduce our approach. In Salembier's algorithm, the hierarchical first-in-first-out (FIFO) data structure is of great importance. It is made of L queues each of which corresponds to one of the L gray levels. For instance, such a queue at level h determines the processing order of the pixels of grayscale value h. A `status` array of the same size as the image stores the information regarding the pixels. Accordingly, a pixel p at level h can be in one of the following three states: `NOT_ANALYZED`, `IN_THE_QUEUE` or assigned the value k. The latter determines which h^{th} level node, i.e., which connected component at level h, the pixel

belongs to. Another auxiliary array of length L can be referred as NUMBER_NODES[h] and stores the index of the connected component, at level h, which is being processed. Depending on whether an opening or a closing is desired, the flooding procedure starts at lowest or highest gray level respectively and it consists of two successive steps: first a propagation step, where all the pixels of a connected component at the current level are explored, and then a parent resolving step where explicit construction of tree branches occurs. Further details concerning the dynamics of the algorithm can be found in [10].

Our algorithm for area opening relies on the flooding process of the max-tree but recall that we are not interested in building this tree. So, in a sense, we combine the filtering stage with the flooding process. Before describing our algorithm, we present some data structures we need. The idea is to use the array status to store different type of information: it can store points which act as pointers to points in order to implement a hierarchical queue or to keep track of disjoint sets; or it can store gray-level values. We store pointers to pixels with a non-negative integer whose value is $y * width + x$ where x and y are pixel coordinates, and $width$ is the width of the original image. In contrast, we use a negative number in order to store a gray-level value. For our algorithm a gray-level value h is stored as $-h-1$. Thus pointers are encoded as non-negative values, and gray-level values as negative values.

Basically, during the flooding process, once a pixel has been removed from a queue, we make it point to an arbitrary pixel which represents its component. Components are merged while keeping track of their areas until a component has an area large enough to fulfill the criterion. Areas of currently extracted connected components are stored in an extra array of length L, called area.

To implement a queue using the array status we need an extra array, last, of length L. It contains the last element added to the queue and initialized to a special value NONE which means that no element is in the queue. An illustration of simulating a queue with these arrays is depicted in Figure 1.

In order to have a representative element for each of the extracted components at a given level, we need again an extra array, representative, of length L. Each time a new component is being extracted, its first encountered element is stored in this array. The latter is initialized to a special value NONE which means that there is currently no element to represent the component.

We are now ready to describe the core of our algorithm. Like for the Salembier's algorithm, we launch the flooding with a pixel which has the lowest gray-level. The array status is initialized to NOT_ANALYZED. The whole algorithm is depicted in Figure 2. Once an element p has been dequeued (lines 4-5), it cannot be added to the queue anymore. Thus status[p] is set to the representative element of the current extracted component (line 7). If a pixel in the neighborhood of p (denoted by $\mathcal{N}_p$) which has not been processed is encountered, then it is added into the hierarchical queue (lines 13-14); and if it is a new component, then we update the array representative (lines 11-12). If the level of this new pixel has a higher gray-level value than the current extracted component, then we launch again the flooding with that level (lines 16-18).

Once the propagation of a level h is over, we have to check whether this extracted component will keep its gray-level value h or not. This is determined by the value in status associated with representative[h]. First we get, if it exists, the component at level m with the highest lower level than h (lines 23-25). If it does not exist, then it is the darkest level of the image and its gray-level is stored (line 35). If such a component exists then we check if the current extracted component at level h has a large enough area. If not, we merge it with its parents, i.e, the component at level m (lines 28-29). If it is large enough then the final gray-level value is kept (line 33). In both cases, the area is updated (lines 38 and 40). Finally, since the flooding at level h is over, we re-initialize information associated with this level (lines 38-40).

At the end of the flooding process, every pixel in the array status stores either pointers to representative nodes or gray-levels (in the negative form). Thus we perform a resolving phase such that all pixels stores its associated gray-level value (still in negative form). This process is depicted in Figure 3. A final pass on the image sets the pixels to their real gray-level values.

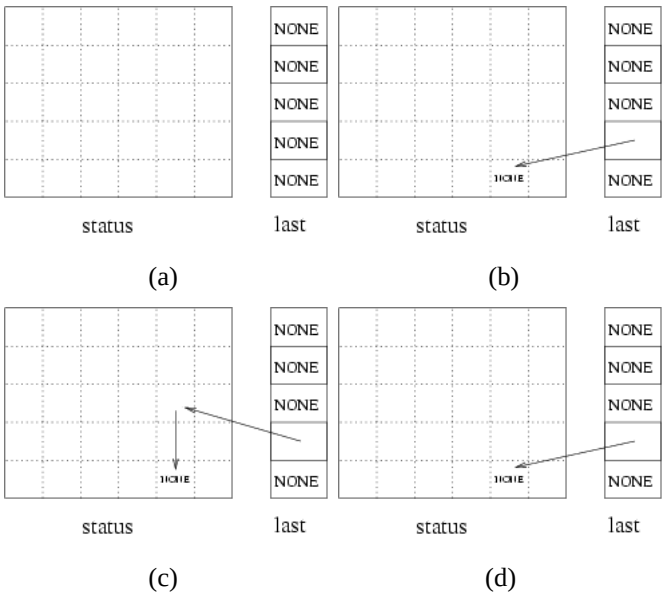

Figure 1: Illustration of our queue. The states of status and last are depicted after different operations on points which are at the same gray-level value: in (a) the original state, in (b) after pushing the point(5,5), in (c) after pushing the point(3,5), and in (d) after a pop.

3. COMPLEXITIES AND EXPERIMENTS

In what follows, N and L respectively denote the number of pixels of the original image and the number of gray-levels. Typically L is equal to 256.

From a theoretical point of view, the computational complexity of the union-find method is $\Theta(N \log N)$ as shown in [8]. The complexity of Salembier's and our algorithm is $\Theta(N)$ because it is mainly dominated by the flooding process. The resolving process is also linear.

Considering the use of memory, our algorithm is the best one. Indeed, apart from the original image, the union-find

```
flood(h)
   while (last[h] == NONE)
      // propagation at level h
      p = last[h]
      last[h] = status[last[h]]
      // set to its representative element
      status[p] = representative[h]
      for all q ∈ N_p such that q ∈ Ω
         if (status[q] == NOT_ANALYSED)
         // set representative element if none
            if (representative[h] == NONE)
               representative[h] = p
            status[p] = last[h]
            last[h] = p
            val = I[q]
            if (val > h)
               m = q_val
               do {m = flood(m)} while (m > h)
      area[h] +=1

// parent settings
m = h−1
while ((m >= 0) and
       (representative[m] == NONE))
   −−m
if (m>=0)
  if (area[h] < criteria)
     status[representative[h]]=
        representative[m]
     area[m] += area[h]
  else
     area[m] = criteria
     status[representant[h]] = −h − 1
else
   status[representant[h]] = −h − 1
// reset attribute of extracted connected component
// at level h
area[h] = 0
last[h] = NONE
representative[h] = NONE
return m
```

Figure 2: Generic flooding process for an area opening

```
for all p ∈ Ω
   root = status[p];
   while (status[root] >= 0)
      root = status[root];
   val = status[root];
   while (p != root)
      tmp = status[p]; status[p] = val; p = tmp;
```

Figure 3: Resolving process after the flooding process.

method needs two arrays of N integers. The first one is used to sort the pixels while the second one is required to maintain the disjoint sets and to store the area. So its required memory is $2N$ integers. The max-tree approach is the method with the most pronounced memory requirement. It needs a hierarchical queue (N integers), an array of N integers for the status array, and finally N nodes for the tree. Each node contains a pointer to its parent, its area, and its output gray-level value (so 3 integers for each node). In total, the max-tree algorithms requires $5N$ integers. Our algorithm needs only an array of N integers. Note that since $L \ll N$, we neglect the memory used by arrays of size L for both the max-tree and our approach. If the attribute is not based only on the area anymore, more memory is required: An extra array of size N is needed for both the union-find and the max-tree approaches, while our algorithm requires only an extra array of size L.

All of the algorithms cited above have been implemented in C/C++ and compiled with full optimizations. We have used different images to perform our experiments. The first image is a synthetic image which depicts a chessboard (512x512) whose cell size is 2. The other ones are natural images: the well-known lena (512x512), a highly textured image of a carpet (715x1024) and finally a satellite image (1780x1380). Due to space restriction, we only present the results for these images. For each image, time results (in seconds) for area opening with different area thresholds are shown in Figure 4. Time results were obtained by taking the mean CPU times on a Pentium 4 3.0GHz (1024 Kb of cache memory) over 20 runs. Note that same performances were obtained using a Celeron 2.6GHz with 256 Kb of cache memory. As seen on Figure 4, our algorithm outperforms the other approaches for natural images. The unique instance where the union-find algorithm is superior to ours happens for the synthetic chessboard image which contains only two graylevels. We observe that the superiority of our algorithm depends on the content of the image (synthetic versus natural), but not on the image size. It is interesting to see that for natural images the union-find algorithm outperforms the max-tree only for the satellite image which has 30 graylevels. It seems that the fewer the graylevels are, the faster the union-find behaves. Execution time of our algorithm is independent of the area threshold chosen. The same observation holds for the max-tree approach while only a slight dependence to area is observed for the union-find approach. Similar results for attributes other than area are available at `http://www.perso.fr/~darbon/eusipco`.

Figure 5 illustrates the use of an area-closing for a segmentation task. A classical approach using morphological tools consists of computing the norm of the gradient of an image and to apply the watershed transform on it [12]. However, it yields an over-segmentation due to many spurious minima in the gradient. So, we filter the gradient image by applying an area-closing. As seen in Figure 5, when the area threshold is increased, the image is significantly simplified while the countours are preserved [1].

4. CONCLUSION

In this paper we have presented an efficient algorithm to perform attribute openings and closings. It performs very satisfactorily on natural images. Moreover, it requires much less memory than any other algorithm available to our knowl-

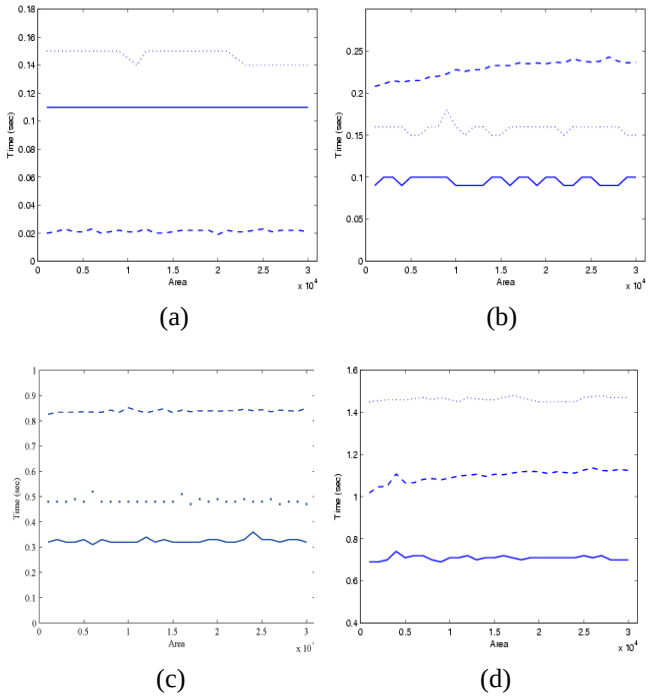

Figure 4: Time results of an area-opening using the max-tree approach (dots), the union-find-method (dashed) and our algorithm (solid) as a function of area (Results for chessboard, lena, carpet and satellite images in (a), (b), (c) and (d) respectively).

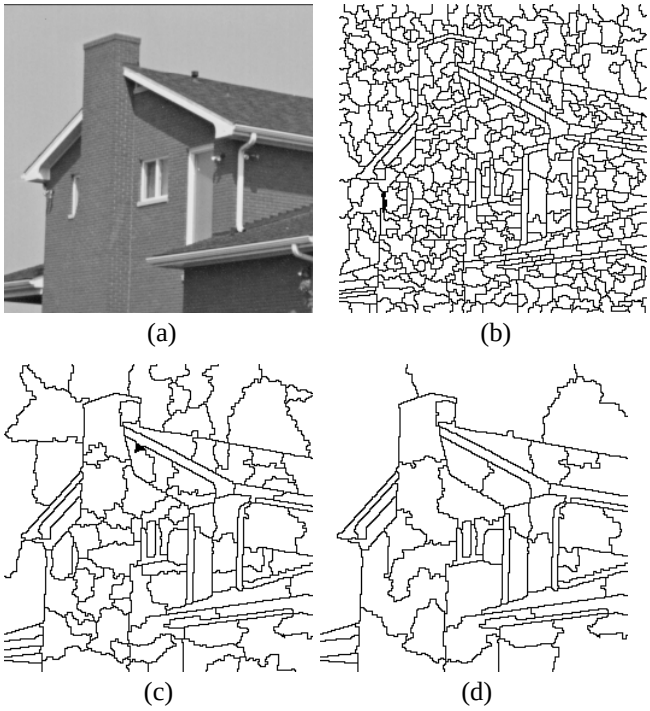

Figure 5: Illustration of the effect of an area closing. The original image is depicted in (a). An area closing is performed on the norm of the gradient and the watershed transform is applied. Results for area closings of area 10, 50 and 100 are respectively depicted in (b), (c) and (d).

edge. Further investigation on the behavior of the union-find approach with respect to image content (especially, the number of available graylevels) must be explored.

We are currently working on the extension of our algorithm to non-increasing attributes which correspond to thinnings and thickenings [2], and on computing pattern spectra. Finally, extension of this approach to the grain filter [3] is currently under investigation.

Acknoledgement

The authors would like to thank M.H.F. Wilkinson (University of Groningen) for providing the code of the union-find approach [8].

REFERENCES

[1] S.T. Acton and D.P. Mukherjee. Scale space classification using area morphology. *IEEE Transactions on Image Processing*, 9(4):623–635, April 2000.

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

[3] V. Caselles and P. Monasse. Grain filters. *J. of Math. Imaging and Vision*, 17(3):249–270, 2002.

[4] J. Crespo, R. Schafer, J. Serra, C. Gratin, and F. Meyer. The flat zone approach: A general low-level region merging segmentation method. *Signal Processing*, 62(1):37–60, 1998.

[5] W.H. Hesselink. Salembier's min-tree algorithm turned into breadth first search. *Information Processing Letters*, 88(5):225–229, December 2003.

[6] X. Huang, M. Fisher, and D. Smith. An efficient implementation of max tree with linked list and hash table. In *Proc. of the Int. Conf. on Digital Image Computing: Techniques and Applications*, pages 299–308, 2003.

[7] R. Jones. Connected filtering and segmentation using component trees. *Computer Vision and Image Understanding*, 75(3):215–228, September 1999.

[8] A. Meijster and M.H.F. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Trans. on PAMI*, 24(4):484–494, Avril 2002.

[9] L. Najman and M. Couprie. Quasi-linear algorithm for the component tree. In *SPIE Symposium on Electronic Imaging*, pages 18–22, 2004.

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

[11] J. Serra and P. Salembier. Connected operators and pyramids. In *Proc. SPIE Image Algebra Math. Morphology*, volume 2030, pages 65–76, 1993.

[12] P. Soille. *Morphological Image Analysis Principles and Applications*. Springer-Verlag, 1999.

[13] L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In *Proc. EURASIP Mathematical Morphology and Its Application to Signal Processing*, pages 22–27, 1993.