CONCURRENT COMPUTATION OF CONNECTED PATTERN SPECTRA FOR VERY LARGE IMAGE INFORMATION MINING

Michael H. F. Wilkinson, Ugo Moschini

Intelligent Systems Group,
Johann Bernoulli Institute for
Mathematics and Computer Science,
University of Groningen, P.O. Box 800,
9700 AV Groningen, Netherlands.

Georgios K. Ouzounis, Martino Pesaresi.

Global Security and Crisis Management Unit,
Institute for the Protection and Security of
the Citizen, Joint Research Centre,
European Commission, Via E.Fermi 2749,
I-21027 Ispra (VA),Italy.

ABSTRACT

This paper presents a shared-memory parallel algorithm for computing connected pattern spectra from the Max-Tree structure. The pattern spectrum is an aggregated feature space derived directly from the tree-based image representation and is a powerful tool for interactive image information mining. An application example along with timings on experiments with Gpixel input imagery are given. On images of 0.87 to 1.29 Gpixel, wall-clock times of 8.13 to 15.17s, and a speed up of between 27.5 and 33.5 were achieved on a single 2U 64 core rack server.

***Index Terms*—** Pattern spectrum, Max-Tree, parallel algorithm, image information mining.

1. INTRODUCTION

Connected pattern spectra [1] are effective feature vectors in many pattern recognition and image information mining tasks. These are univariate or multivariate histograms of the image features, in which each bin denotes how much image content lies in a particular size or shape class. Pattern spectra can be computed efficiently from hierarchical image representations like the Max-Tree/Component Tree [1–3] and the Alpha-Tree [4]. These tree structures allow interactive retrieval and management of the component attributes and have been used extensively in image information mining [5,6], and interactive, explorative visualisation in 3D [7,8].

An example in the domain of remote sensing image analysis is the mining of building footprint candidates. The process is often driven by selections of target examples. Interpreting the example attributes into bin IDs allows identification of pattern spectrum entries describing best the targets. In a pass through the tree structure, all nodes associated to components that satisfy the selected bin criteria, are marked to be preserved while all others to be removed.

Fig. 1 demonstrates this application on a panchromatic Quickbird image of Sana'a, Yemen, courtesy of DigitalGlobe.

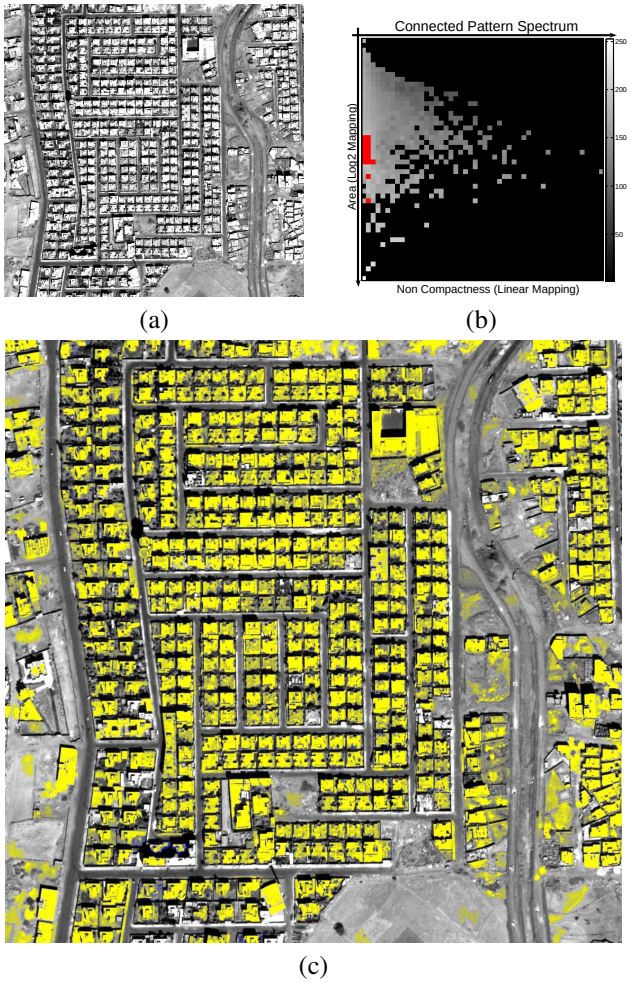


Fig. 1. (a) A sample tile from the input image; (b) the connected pattern spectrum with a set of bins selected (in red); (c) the resulting segmentation.

The image in Fig. 1(a) is a sub-tile of a larger data set covering a test area of 10.6×26.6 km^2. Figure 1(b) shows a selection of pattern spectrum bins driven by a set of positive examples, i.e. ROIs on the image each fully containing a target instance; a building in this case. The 2D pattern spectrum shown, was computed using non-compactness as shape information (horizontal axis) and area as size information (vertical axis), as in [1]. Fig. 1(c) shows the filtering result, which is the input image overlayed with the identified objects in yellow.

Because image sizes are increasing with increasing sensor resolution, a parallel approach for computation is needed. In this paper, we present a new algorithm for the concurrent computation of pattern spectra from the Max-Tree structure, based on the shared-memory parallel Max-Tree algorithm in [9]. The current version is capable of handling images up to 4Gpixels, the limit of the Geo-TIFF format.

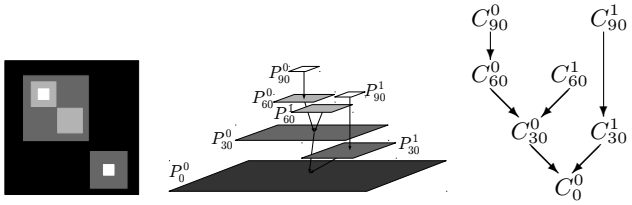
The structure of this paper is organised as follows. Section 2 gives a brief introduction on the Max-Tree structure and on the method for computing connected pattern spectra from it. Section 3 introduces the new algorithm and gives a pseudo-code listing. Section 4 presents timings and performance figures of the proposed algorithm on images of 870 Mpixels to 1.295 Gpixels from both remote sensing and astronomy. Section 5 discusses the new method, gives overview of its features and finishes off with a summary of conclusions.

2. CONNECTED FILTERS

In connected mathematical morphology [10], all image operations are carried out at the level of *connected components*, i.e. connected subsets of foreground pixels of maximal extent. The pattern spectra discussed here are based on connected attribute filters [11]. In the binary case, attribute filters compute some property or attribute, like area, compactness, elongation, moment of inertia, entropy, etc, of each connected component. Based on these attributes, it is decided which connected components to keep, and which to remove. This can be generalised to the grey-scale case by thresholding at all possible grey levels, computing the binary filters, and stacking the results into a new binary image. In practice, the Max-Tree [2] can be used to speed up this process and derive more versatile operators.

2.1. The Max-Tree Structure

In a Max-Tree, each node of the tree represents a *peak component*, which is a connected component of a threshold set of the image. An example is shown in Fig. 2. As can be seen, we can represent the image in Fig. 2(a) as a set of nested peak components P_h^k shown in Fig. 2(b), with h the grey level and k an index number. Using the nesting relationship, we can assign a single parent to each peak component at a grey level just below its own. These parent-child relationships are the edges in the Max-Tree graph, whereas each node C_h^k correspond to



(a) 2D image (b) Peak components (c) Max-Tree

Fig. 2. A simple 2-D image, its peak components P_h^k, and corresponding nodes C_h^k in the Max-Tree

set of pixels in P_h^k at level h. Quite obviously, the number of Max-Tree nodes is smaller than or equal to the number of pixels in the image. Usually it is much smaller.

The Max-Tree is used for efficient grey-scale attribute filtering by computing an attribute per node, applying some criterion to that value (e.g. by thresholding), and deciding which nodes to retain and which to remove. New grey levels are then assigned to each node, and the output image is computed by assigning the new grey level to each pixel in each node.

As an example, consider the grey-scale area opening [12, 13]. In the binary case, this removes connected foreground components smaller than an area λ and leaves the rest unchanged. We can compute the area opening in grey level by thresholding the image, performing the binary area opening on each and stacking the result. Using the Max-Tree, we can compute the area of each node efficiently, and then simply traverse the tree. Applying an area opening with $\lambda = 5$ to Fig. 2(a), removes the small peak components P_{90}^0 and P_{90}^1. This is achieved by setting the new grey levels of these nodes to the grey level of first ancestor with an area $\geq \lambda$, as we traverse the tree from leaf to root. In this example the relevant nodes are their respective parents, so the new grey level for C_{90}^0 becomes 60, and that of C_{90}^1 becomes 30.

Here we use the adaptation by [9] of Salembier et al's algorithm [2] to build the Max-Tree and compute the attributes. In the representation used by [9] each pixel is considered as a node in the tree. Each node contains fields for the parent pointer, area, and attribute data. All members of a given C_h^k point directly or indirectly to a single member r_h^k of C_h^k as parent. This node r_h^k, called the *level root* of C_h^k , points to a member of the parent node $C_{h'}^m$ of C_h^k. Thus, only the level roots really represent the Max-Tree, and the rest just indicate which node they belong to. Level roots can be recognised by the fact that they point to a node at a lower grey level.

2.2. Connected Pattern Spectra

Pattern spectra [14] are traditionally computed by applying a series of morphological openings to an image, each with a larger structuring element, e.g. a series of disks with increasing radii r_i. By taking difference images between the results

of scale r_i and r_{i+1}, we obtain those features in the image with a radius between those two values. By summing the grey levels of all pixels in such a difference image, we obtain the amount of image content at that scale. This allows us to summarise the image content in a 1D feature vector which tells us how much content is in the image at each scale i. Other opening, such as area openings can be used instead of structural morphological openings.

A key feature of Max-Trees is that it is possible build the tree once, which is the costly part, and use it to filter an image many times, which is relatively cheap. Suppose we compute the area of each node, as before. It then contains all the information you need to compute any area opening, and even more complex filters, like the difference between two area openings directly, simply by applying the right criterion to each node. We can also compute the area pattern spectrum not by repeated filtering, but by a single traversal of the tree [1].

We first set all bins in the spectrum to zero. We then traverse the tree in arbitrary order. As we visit each node, we decide into which scale class it falls based on its area. We then add the product of its area and the grey level difference with its parent to the appropriate bin in the spectrum. This method is about as fast as the computation of a single filtered image [1]. It can also readily be extended to multiple attribute to compute multi-D pattern spectra [1]. An example is shown in Fig. 1.

Algorithm 1 Procedure ccaps (combined construction and pattern spectrum): parameters are thread p, MaxTreeNodes $nodes$, and private pattern spectrum *threadPatSpec*

```
process ccaps(p, nodes, threadPatSpec)
    LocalMaxTreeBuild(V^p) ;
    var i := 1 , q := p ;
    while p + i < K ∧ q mod 2 = 0 do
        P(sa[p + i])   (∗ wait to glue with
                          right-hand neighbour ∗) ;
        FuseTrees(p, (p + i)) ;
        i := 2 ∗ i ; q := q/2 ;
    end ;
    if p ≠ 0 then
        V(sa[p])   (∗ signal left-hand neighbour ∗) ;
    end ;
    Barrier(p)   (∗ use Barrier synchronisation type ∗) ;
    MaxTreePatSpectrum2D(p, nodes, threadPatSpec[p]) ;
    Barrier(p)   (∗ use Barrier synchronisation type ∗) ;
    if p = 0 then
        for i := 1 to K − 1 do
            SumPatSpecs(0, i) ;
        end ;
    end ;
end ccaps .
```

3. THE PARALLEL ALGORITHM

The key to parallelizing the computation of connected pattern spectra lies in parallelizing the construction of the Max-Tree. This has been achieved previously in [9]. The solution consists of building Max-Trees for disjoint sections V^p of the image for each thread p, and then stitching them together in a hierarchical fashion. Assuming we have 8 threads building the trees, we first combine pairs of adjacent trees using 4 threads, then combine these results using 2, and finally merge the last result using a single thread. This process is shown in the while loop in Alg. 1. Provided the number of different grey levels is small ($\leq$ 16 bits per pixel) this is efficient. In the original code the thread doing the final merge would signal the others that they could proceed with the filtering phase. The top half of the pseudo-code shown in Algorithm 1 until the first barrier is essentially the same as in [9]. The reader is referred to that paper for a more detailed discussion.

We adapted the code from [9] to compute two attributes: area as a size attribute and a shape attribute of the user's choice. In the experiment we used non-compactness as shape attribute, i.e., the inverse of Hu's first moment invariant [15] normalised to scale between 0 and 1. Once built, we replaced the filtering phase with a part performing pattern spectrum computation.

Computation of the pattern spectra, shown in Alg. 2 consists of traversing part of the tree in the private section V^p of each thread p, and for every *level root* decide which bin in the spectrum it should be assigned to, based on the two attributes. Once the bin indices *sArea* and *sAttrib* have been computed, the product of the node area and the grey level difference with its parent is added to the appropriate bin of the spectrum. To do this in parallel, we could use one of two strategies. In the first we allocate a single pattern spectrum array, and ensure through mutexes that no two threads try to increment the spectrum at the same time. In the second we allocate as many pattern spectrum arrays as there are threads, and let each compute a partial pattern spectrum of their segment. These partial pattern spectra are added together, to form the final result.

We chose the latter approach because the use of so many mutexes in the former slows computation down. The latter approach does incur an extra memory burden, but as the size of the pattern spectrum (50×50 doubles in this test) is tiny compared to the size of the images, this is hardly a problem. Thus each instance of *MaxTreePatSpectrum2D* is handed a private spectrum array *threadPatSpec*.

Once all private pattern spectra are computed in Alg. 1 the individual partial spectra are summed by a single thread. We experimented with a hierarchical merge of the pattern spectra in a similar vein as the merge of the trees, but it turns out this is more costly, as synchronisation overhead outstrips speed gains on this minute part of the computation.

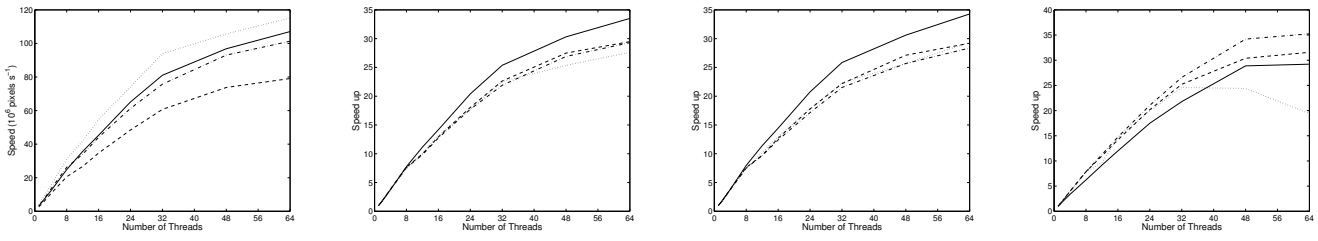


Fig. 3. Results: (a) total speed, in 10^6 pixels per second; (b) total speed up; (c) speed-up for the max-tree computation; (d) speed-up for the pattern-spectrum computation. In all three cases the solid line represents the 870 Mpixel image of Sana'a, the dashed the 1.2 Gpixel image of Port-au-Prince, and the dashed dot and dotted lines are the two 1.27 and 1.295 Gpixel astronomical images respectively.

Algorithm 2 *MaxTreePatSpectrum2D* - parameters are thread p, MaxTreeNodes $nodes$, and private pattern spectrum *threadPatSpec*. Variable f denotes the grey-scale input image.

process *MaxTreePatSpectrum2D*$(p, nodes, threadPatSpec)$
  for all pixels $v \in V^p$ **do**
    if *isLevelRoot*$(nodes, v)$ **then**
      sArea := *getScalesArea*$(nodes[v])$;
      sAttr := *getScalesAttr*$(nodes[v])$;
      if not *isTreeRoot*$(nodes[v])$ **then**
        threadPatSpec$[sArea][sAttr]$:=
          threadPatSpec$[sArea][sAttr]+$
            $(f[v] - f[nodes[v].parent])\times$
            nodes$[v]$.*Area* ;
      end ;
    end ;
  end ;
end *MaxTreePatSpectrum2D* .

4. RESULTS

The algorithm was implemented in C and tested on a Dell R815 compute server with four 16-core AMD Opteron processors, and total memory of 512 GB. Though the server has 64 cores, it has only 32 floating point units, each shared by a pair of cores. Tests were run on the full 870 Mpixel panchromatic image of Sana'a, courtesy of Digital Globe, and a 1.2 Gpixel image created by replicating a section of a 3000×4000 pixel aerial photography image of Port-au-Prince. Two tiled astronomical images of 1.27 and 1.295 Gpixel were created, to study the behaviour of the algorithm on images with very different statistics. The tilings were made by mirroring the images in such a way that no artificial edges emerge. The results are shown in Fig. 3.

To correct for differences in image size Fig. 3(a) shows the increase of compute speed in millions of pixels per second as a function of the number of threads. Speeds increase rapidly between 1 and 32 threads, from about 3 to 4.5 million pixels per second to between 60 and 90 million pixels per second. From 32 to 64 threads, a slower increase is seen to about 80 to 115 million pixels per second. Wall-clock times reduce from between 272 to 447 s to between 8.13 and 15.17 s. The bulk of this is computation of the Max-Tree, costing between 7.22 and 12.26 s on 64 threads, whereas computation of the pattern spectrum took between 0.85 and 2.82 s. By comparison, on a single thread, computation of the pattern spectra took between 24.84s for the smaller Sana'a image to 89.04 s for the large Haiti image.

As can be seen in Fig. 3(b)-(d), up to about 32 threads the efficiency is quite good, but above that the increase in speed-up drops off, and does not rise about a factor of roughly 32. Initially we suspected this was due to the use of floating point operations in the computation of the attribute. As there are only 32 FPUs, this could cause a bottleneck. We modified our code to avoid floating point math during the parallel phase almost completely, but this did not solve the problem. The timings shown are those of the latter version.

In terms of Mpixels per second there are no systematic differences between the astronomical and remote sensing images, though in general images with more nested features and higher dynamic range in grey scale are more costly to compute, simply because these contain more Max-Tree nodes, needing more computation.

5. DISCUSSION AND CONCLUSIONS

Computing connected pattern spectra for very large images can be performed efficiently by extending the approach of [9]. In particular, processing times for Gpixel images drop from several minutes to between 8 and 15 seconds. Equally important, we could compute the Max-Tree once, and allow the user to interactively set the desired parameters for the spectrum, such as number of bins, range, log or linear scaling, and recalculate the pattern spectrum in less than 3s on our 64 core

machine. When interacting with the pattern spectrum to select features to filter out of the image, a similar response time is obtained. This performance was achieved on a single 2U rack server, not a supercomputer.

One curious issue is the quite sudden change in behaviour beyond 32 threads. In related, as yet unpublished results on a similar Max-Tree based method, we achieve a speed up of up to 45×, rather than 32× as in this case. A key difference is the use of different, more costly attributes in this case (based on 2D moment of inertia, rather than simply area), so some optimization there might help. The current algorithm accepts any combination of scale and shape attributes that can be computed whilst building the tree [1–3].

Memory usage is quite high, a 1.2 Gpixel image typically requiring 90 GB memory in our current implementation. We are working on more memory efficient versions. Memory does scale linearly with the image size.

As stated we have limited our code to the existing 4GB image size dictated by Geo-TIFF. Moving beyond that barrier requires use of 64 bit integers to index the arrays and to compute node areas. This increases memory use, but does not require any structural change to the code. A further limitation is that we can only run efficiently on shared memory machine. Ultimately, we aim to develop code for distributed memory machine, to address the problem of computing Max-Trees on terapixel and petapixel images.

ACKNOWLEDGEMENTS

The position of U. Moschini and the Dell R815 compute server were funded by the Netherlands Organisation for Scientific Research (NWO) under project number 612.001.110

6. REFERENCES

[1] E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 29, no. 2, pp. 272–285, 2007.

[2] P. Salembier, A. Oliveras, and L. Garrido, "Anti-extensive connected operators for image and sequence processing," *IEEE Trans. Image Processing*, vol. 7, no. 4, pp. 555–570, 1998.

[3] R. Jones, "Connected filtering and segmentation using component trees," *Comp. Vis. Image Understand.*, vol. 75, no. 3, pp. 215–228, 1999.

[4] G. K. Ouzounis and P. Soille, "Pattern spectra from partition pyramids and hierarchies," in *Proc. Int. Symp. Math. Morphology (ISMM) 2011*, P. Soille, M. Pesaresi, and G.K. Ouzounis, Eds., vol. 6671 of *Lecture Notes in Computer Science*, pp. 108–119. Springer Berlin/Heidelberg, 2011.

[5] L. Gueguen and G.K. Ouzounis, "Hierarchical data representation structures for interactive image information mining," *International Journal of Image and Data Fusion*, vol. in press, 2012.

[6] G.K. Ouzounis and L. Gueguen, "Interactive collection of training samples from the max-tree structure," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, Sept. 2011, pp. 1449–1452.

[7] M. A. Westenberg, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Volumetric attribute filtering and interactive visualization using the max-tree representation," *IEEE Trans. Image Processing*, vol. 16, pp. 2943–2952, 2007.

[8] A. C. Jalba and M. A. Westenberg, "A comparison of two tree representations for data-driven volumetric image filtering," in *Proc. Int. Symp. Math. Morphology (ISMM) 2011*, 2011, vol. 6671 of *Lecture Notes in Computer Science*, pp. 405–416.

[9] M. H. F. Wilkinson, H. Gao, W. H. Hesselink, J. E. Jonker, and A. Meijster, "Concurrent computation of attribute filters using shared memory parallel machines," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 30, no. 10, pp. 1800–1813, 2008.

[10] J. Serra, Ed., *Image Analysis and Mathematical Morphology. II: Theoretical Advances*, Academic Press, London, 1988.

[11] E. J. Breen and R. Jones, "Attribute openings, thinnings and granulometries," *Comp. Vis. Image Understand.*, vol. 64, no. 3, pp. 377–389, 1996.

[12] F. Cheng and A. N. Venetsanopoulos, "An adaptive morphological filter for image processing," *IEEE Trans. Image Processing*, vol. 1, pp. 533–539, 1992.

[13] L. Vincent, "Morphological area openings and closings for grey-scale images," in *Shape in Picture: Mathematical Description of Shape in Grey-level Images*, Y.-L. O, A. Toet, D. Foster, H. J. A. M. Heijmans, and P. Meer, Eds., pp. 197–208. NATO, 1993.

[14] P. Maragos, "Pattern spectrum and multiscale shape representation," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 11, pp. 701–715, 1989.

[15] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. IT-8, pp. 179–187, 1962.