Chapter 6

Concurrent Computation of Connected Pattern Spectra for Very Large Image Information Mining

Abstract

This chapter 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 Gigapixel input imagery are given. On images of 0.87 to 1.29 Gigapixel, 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.
6.1 Introduction

Connected pattern spectra (Urbach et al., 2007) have been used as effective feature vectors in many pattern recognition and image information mining tasks. In essence, they 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 (Salembier et al., 1998; Jones, 1999; Urbach et al., 2007) and the alpha-tree (Ouzounis and Soille, 2011). These tree structures allow interactive retrieval and management of the component attributes and have been used extensively in image information mining (Gueguen and Ouzounis, 2012; Ouzounis and Gueguen, 2011; Ouzounis et al., 2012), and interactive, explorative visualisation in 3D (Westenberg et al., 2007; Jalba and Westenberg, 2011).

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. 6.1 demonstrates this application on a very high resolution (VHR), panchromatic Quickbird acquisition of Sana’a, Yemen, courtesy of DigitalGlobe. The image in Fig. 6.1(a) is a sub-tile of a larger data set covering a test area of $10.6 \times 26.6$ km$^2$. Fig. 6.1(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 Urbach et al. (2007). Fig. 6.1(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. We are talking about very large images, with resolution in the order of Gigapixels. In this chapter, we present a new algorithm for the concurrent computation of pattern spectra from the max-tree structure, based the shared-memory parallel max-tree algorithm of Chapter 4. The current version is capable of handling images up to 4 Gigapixels, which is the limit of the Geo-TIFF format. The structure of this chapter is organised as follows. Section 6.2 gives a brief introduction on connected operators, the max-tree structure and connected pattern. Section 6.3 introduces the new algorithm and gives a pseudo-code listing. Section 6.4 presents timings and performance figures of the proposed algorithm on images of 870 Megapixels to 1.295 Gigapixels from both remote sensing and astronomy. Section 6.5 discusses the new method, gives an overview of its features and finishes off with a summary of conclusions.
6.1. Introduction

Figure 6.1: (a) A sample tile from the input VHR image of Sana’a, Yemen; (b) the connected pattern spectrum with a set of bins selected (in red); (c) the resulting segmentation.
6.2 Connected filters

In connected mathematical morphology (Serra, 1988), all image operations are carried out at the level of connected components, i.e. connected sets of foreground pixels of maximal extent. The pattern spectra discussed here are based on connected attribute filters (Breen and Jones, 1996). 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 grey-scale image. In practice, the max-tree (Salembier et al., 1998) can be used to speed up this process and derive more versatile operators.

6.2.1 The max-tree structure

Here we use the adaptation by Wilkinson et al. (2008) of Salembier et al’s algorithm (Salembier et al., 1998) to build the max-tree and compute the attributes. In the representation used by Wilkinson et al. (2008), 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 connected component $C^k_h$ indexed by $k$ at intensity $h$, as in Fig. 3.9 in Chapter 3, point directly or indirectly to a single member $r^k_h$ of $C^k_h$ as parent. This node $r^k_h$, called the level root of $C^k_h$, points to a member of the parent node $C^m_h$ of $C^k_h$. 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.

6.2.2 Connected Pattern Spectra

Pattern spectra (Maragos, 1989) 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 $i$. Connected openings can also be used in place of their structural counterparts.

A key feature of max-trees is that it is possible build the tree once, which is the costly part both in memory and computation time, and use it to filter an image many times, which is relatively cheap. Suppose we compute the area of each node,
6.3 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 Wilkinson et al. (2008). 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. 6.1. 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 Alg. 6.1 until the first barrier is essentially the same as in Wilkinson et al. (2008). The reader is referred to that paper for a more detailed discussion.

We adapted the code from Wilkinson et al. (2008) 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 (Hu, 1962) normalised to scale between 0 and 1. Once built, we replaced the filtering phase with a part performing the pattern spectrum computation.

Computation of the pattern spectra, shown in Alg. 6.2 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 $s\text{Area}$ and $s\text{Attrib}$ 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
Algorithm 6.1. Procedure ccaps (combined construction and pattern spectrum) - thread $p$.

```plaintext
procedure CCAPS(p)
    LocalMaxTreeBuild($V^p$)
    $i \leftarrow 1$, $q \leftarrow p$
    while $p + i < K \land q \text{ mod } 2 = 0$ do
        $\triangleright$ wait to glue with right-hand neighbour
        $P(sal[p + i])$
        FuseTrees(p, (p+i))
        $i \leftarrow 2 \ast i$, $q \leftarrow q/2$
    end while
    if $p \neq 0$ then
        $V(sal[p])$ \hspace{1cm} \triangleright$ signal left-hand neighbour
    end if
    Barrier(p) \hspace{1cm} \triangleright$ use Barrier synchronisation type
    MaxTreePatSpectrum2D(p, nodes, threadPatSpec[p])
    Barrier(p) \hspace{1cm} \triangleright$ use Barrier synchronisation type
    if $p = 0$ then
        for $i \leftarrow 1$ to $K - 1$ do
            SumPatSpects(0, i)
        end for
    end if
end procedure
```

Algorithm 6.2. MaxTreePatSpectrum2D - parameters are thread $p$, MaxTreeNodes $nodes$, and private pattern spectrum threadPatSpec. Variable $f$ is the grey-scale input image.

```plaintext
procedure MaxTreePatSpectrum2D(p, nodes, threadPatSpec)
    for all pixels $v \in V^p$ do
        if isLevelRoot(nodes, v) then
            $sArea \leftarrow \text{getScalesArea}(nodes[v])$
            $sAttr \leftarrow \text{getScalesAttr}(nodes[v])$
            if not isTreeRoot(nodes[v]) then
                $\text{threadPatSpec}[sArea][sAttr] \leftarrow \text{threadPatSpec}[sArea][sAttr] + (f[v] - f[nodes[v].parent]) \times \text{nodes[v].Area}$
            end if
        end if
    end for
end procedure
```

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 the computation down. The latter approach does incur an extra memory burden, but as the size of the pattern spectrum ($50 \times 50$...
doubles in this test) is tiny compared to the size of the images, this is hardly a problem. Thus each instance of \texttt{MaxTreePatSpectrum2D} is handed a private spectrum array \texttt{threadPatSpec}.

Once all private pattern spectra are computed in Alg. 6.1 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.

6.4 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 Megapixels panchromatic image of Sana’a, courtesy of Digital Globe, and a 1.2 Gigapixel 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 Gigapixel were created, to study the behaviour of the algorithm on images with very different statistics. The results are shown in Fig. 6.2.

To correct for differences in image size Fig. 6.2(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 and 447 s to between 8.13 and 15.17 s. The bulk of this is the 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 24.84 s for the smaller Sana’a image to 89.04 s for the large Haiti image.

As can be seen in Fig. 6.2(b)-(d), for 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 the speed-up did not improve. It is unclear what is preventing the algorithm from scaling-up properly. The timings shown are those of the latter version.

In terms of megapixels 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.

### 6.5 Discussion and Conclusions

Computing connected pattern spectra for very large images with resolution in the order of Gigapixels can be performed efficiently by extending the approach of Wilkinson et al. (2008). In particular, processing times for gigapixel 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 pa-
rameters 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 $45\times$, rather than $32\times$ as in this case. A key difference is the use of different, more costly attributes in this case, so some optimization there might help. Memory usage is quite high, a 1.2 Gigapixel 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 machines, to address the problem of computing max-trees on terapixel and petapixel images.