Fast Pattern Spectra using Tree Representation of the Image for Patch Retrieval

Behzad Mirmahboub, Jérôme Moré, David Youssefi, Alain Giros, François Merciol, Sébastien Lefèvre

Fast Pattern Spectra using Tree Representation of the Image for Patch Retrieval

Behzad Mirmahboub[1], Jérôme Moré[1], David Youssefi[2], Alain Giros[2], François Merciol[1], and Sébastien Lefèvre[1]

[1] IRISA - Université Bretagne Sud, UMR 6074, F-56000, Vannes, France
firstname.lastname@irisa.fr
http://www.irisa.fr/obelix
[2] CNES - Centre National d'Etudes Spatiales, Toulouse, France
firstname.lastname@cnes.fr

Abstract. We extend the notion of content based image retrieval to patch retrieval where the goal is to find the similar patches to a query patch in a large image. Naive searching for similar patches by sequentially computing and comparing descriptors of sliding windows takes a lot of time in a large image. We propose a novel method to compute descriptors for all sliding windows independent from number of patches. We rely on tree representation of the image and exploit the histogram nature of pattern spectra to compute all the required descriptors in parallel. Computation time of the proposed method depends only on the number of tree nodes and is free from query selection. Experimental results show the effectiveness of the proposed method to reduce the computation time and its potential for object detection in large images.

Keywords: content based image retrieval · patch retrieval · tree representation · pattern spectra · large satellite images.

1 Introduction

Content based image retrieval (CBIR) is the problem of finding images in a database that are similar to a query image [10]. This is generally done by two main components. The first step is feature extraction that computes discriminating descriptors for the images. Therefore, each image is represented by a descriptor. The second step is to find the similarities between the images by calculating the distances between their descriptors. Then, the dataset images are ranked based on their distances to the query image and the most similar images are retrieved.

Current datasets for image retrieval consist of the cropped images with specific content for each image and the task of CBIR is to find the similar images. However, in some applications such as remote sensing we have a patch in a large satellite image and we need to find similar patches in the same image or other images (Figure 1). CBIR can be trivially adapted to this problem by dividing the large image into small patches and extracting a descriptor for each image

patch. Then, distances between patches can be computed to find the most similar patches to the query patch. This approach is time consuming and is not applicable in real time retrieval system for very large images.

In order to address this problem, we propose in this paper a patch retrieval system to quickly compute descriptors for all the patches in an image. As far as we know, it is the first work that tries to find the similar patches in a large image. We rely on Pattern Spectra (PS) based on tree representation of the image [12]. It is a histogram-like morphological descriptor for images. We introduce fast-PS descriptor as an approximation of PS that is computed for all image patches in parallel. Therefore, the computation time will be independent from the number of patches which is important for processing large images.

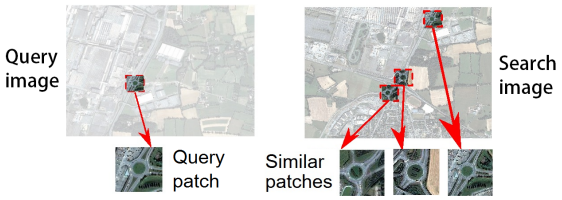


Fig. 1: Patch retrieval: User selects a query patch from an image and the goal is to find similar patches in the same or other images.

This paper is organized as follows. Section 2 reviews previous works on image retrieval, tree representation of the image and pattern spectra. We explain our proposed method in Section 4 and present experimental results in Section 5. Section 6 concludes the paper.

2 Related works

2.1 Image Retrieval

CBIR is a challenging task to find the images in a database that are similar to a query image [10]. The general framework consists of two main steps as it is shown in Figure 2. First, a discriminant feature vector is extracted from each image. Second, these descriptors are used to compute the distance of each dataset image to the query image. The dataset images are sorted according to their distances to the query image and we expect to see the similar images at the beginning of the ranked list.

Various types of feature vectors are proposed in literature based on color or texture of the image such as Gabor Filter [9], LBP (Local Binary Pattern) [8], SIFT (Scale-Invariant Feature Transform) [13], MSER (Maximally Stable Extremal Regions) [16] and recently deep CNN features [4]. An ideal feature vector must be rotation/translation/scale-invariant and robust to changes in illumination. Descriptor computation is the bottleneck in image retrieval. Therefore,

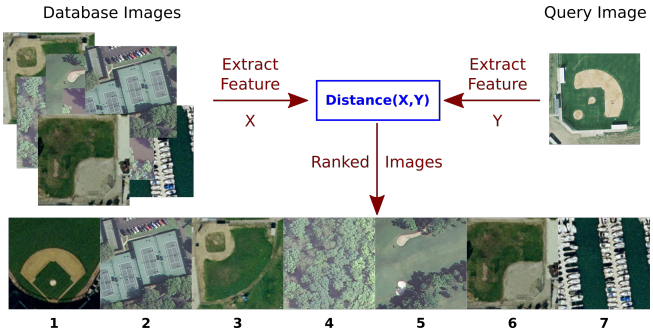


Fig. 2: General framework of content based image retrieval. The images in the database are sorted based on their distances to a query image. (The images are selected from Merced dataset [15] only for illustration.)

practical image retrieval systems pre-compute image descriptors and store them along the original images.

Distance between two feature vectors can be simply computed using their Euclidean distance. However, in many cases the feature vectors are high dimensional and different dimensions have different importance. An efficient solution is to learn a weighting matrix to emphasize the important dimensions [7]. On the other hand, some distance metrics are specially designed to compare histograms. Since pattern spectra is a histogram-like feature vector, we use the well-known χ^2 distance to compare them. The χ^2 distance of two histograms x and y tries to reduce the effect of large bins:

$$\mathcal{X}^2(x,y) = \sum_i \frac{(x_i - y_i)^2}{(x_i + y_i)} \tag{1}$$

Previous works on image retrieval mainly try to compare two whole images [5]. To the best of our knowledge, it is the first time that the concept of CBIR is extended to find similar patches in a large image as depicted in Figure 1. We assume that the user selects a query patch with arbitrary size. Therefore, the query patch size is not known and the descriptors cannot be computed in advance. A closely related field of study exists in literature for patch matching [1]. However, the solutions to this problem cannot meet our requirements since patch matching is generally applied between two similar scenes such as stereo images. Therefore, those methods exploit similarity between images to restrict the possible locations of matching patches and expedite the search, while we do not impose that the images have similar content and therefore we cannot use such methods.

2.2 Evaluation metric

When we calculate and sort the distances between the image patches and the query patch, ideally we expect to see all the relevant images (true positives or tp)

at the beginning of the retrieved list. However, since the descriptors and distance metric are not perfect, we also see irrelevant images (false positives or fp) in the list and even some of the relevant images are not retrieved (false negatives or fn). Mean Average Precision (mAP) is a measure to evaluate the performance of a retrieval system [11]. It shows how well the relevant images are ranked in the retrieved list and it is calculated based on precision and recall.

Precision is defined as the ratio of relevant retrieved images to all retrieved images or $tp/(tp+fp)$ and it is computed for each rank position in the retrieved list. In the example of Figure 2 if baseball field is the relevant class, the precisions for first seven ranks will be $\{\frac{1}{1}, \frac{1}{2}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \frac{3}{6}, \frac{3}{7}\}$. Recall is defined as the ratio of retrieved relevant images to all relevant images in the dataset or $tp/(tp+fn)$. Assume that there are four relevant baseball fields in Figure 2 and that three of them are retrieved in the first seven ranks. Then, the recalls for these ranks will be $\{\frac{1}{4}, \frac{1}{4}, \frac{2}{4}, \frac{2}{4}, \frac{2}{4}, \frac{3}{4}, \frac{3}{4}\}$. Average Precision (AP) is the area under precision-recall curve. Practically, it is the average of precisions at the positions of relevant retrieved images (where the recall is changed). In the above example AP is calculated as $\frac{1}{3}(\frac{1}{1} + \frac{2}{3} + \frac{3}{6}) = \frac{13}{18}$. Finally, mAP is simply the mean value of all APs for different queries.

3 Theoretical background

3.1 Tree Representation

Usual image representation by a matrix of pixels gray levels does not include spatial relations between image components. In contrast, hierarchical image representation shows an image with a tree structure that considers spatial relations [3]. Each node in this tree structure represents a connected component in the original image. Each pair of components in the image are either disjoint or one of them is a subset of the other one making a child-parent relationship. The leaves of the tree consist of the smallest components while the tree root represents the entire image. Such a hierarchical structure lets us quickly process the image in different scales. Depending on how to define the image components, various tree types can be built including max-tree, min-tree, Tree-of-Shapes, α-tree, ω-tree and Binary Partition Tree [3].

Figure 3 shows a simple gray scale image on the left with its max-tree representation in middle. Max-tree is built using the connected components of upper level sets of the image [6] that is defined as:

$$\mathcal{H}_{max} = \{\mathcal{C}(\mathcal{L}^k) | v_{min} \leqslant k \leqslant v_{max}\} \tag{2}$$

where k is a threshold that is selected between minimum gray level v_{min} and maximum gray level v_{max}. Level set $\mathcal{L}^k$ is an image that is obtained by thresholding the original image with k. It only contains all the components with intensity values equal or greater than k and $\mathcal{C}(\mathcal{L}^k)$ is the set of those components. Therefore, $\mathcal{H}_{max}$ is the set of all connected components in different levels.

Each circle in Figure 3 represents a node in the tree structure and the corresponding connected component in the image is shown beside it. Also the gray level k is written next to each tree node. The leaves of the max-tree consists of the brightest image components or local maxima while the tree root represents the whole image.

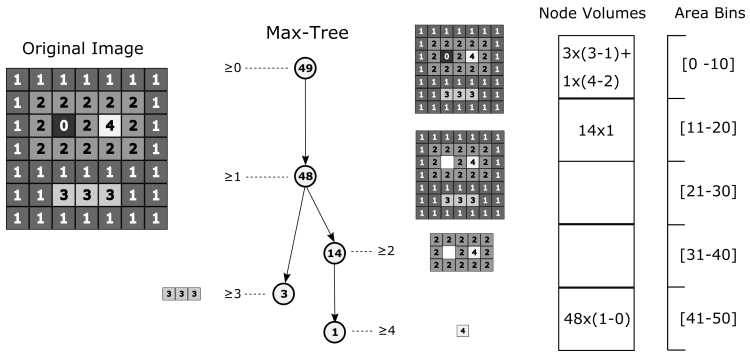


Fig. 3: Computation of pattern spectra using area attribute of a max-tree nodes. Tree nodes are shown by circles in the tree structure. The number inside each circle shows the node area and the number next to it represents the node altitude. Each tree node contributes to a histogram bin by its area multiplied by the gray level difference with its parent.

3.2 Pattern Spectra

Tree representation is an efficient structure to process the image at multliple scales. Pattern Spectra is a morphological image descriptor that can be computed using the tree representation [12]. It is a histogram-like feature vector that shows the distribution of components in an image. After building the tree structure, various geometrical or statistical attributes can be computed for each tree node and PS shows the frequencies of those attributes in the image. In the example of Figure 3, area attribute for each node is written in the relevant circle in the tree structure and PS is shown on the right of the figure. In this example, PS consists of five bins for the area attribute and each tree node contributes to one relevant bin. Each node contributes by its volume which is defined as the node area multiplied by the difference between node gray level and its parent gray level. Contributions of all nodes to the relevant PS bin are summed and the final descriptor will be $[8, 14, 0, 0, 48]$. Generally, bin j of pattern spectra PS is computed based on tree representation of the image as:

$$PS_j = \sum_{\forall i, A(c_i) \in [b_{j-1}, b_j[} a(c_i) \big| k(c_i) - k\big(p(c_i)\big) \big| \tag{3}$$

where $a(c_i)$, $k(c_i)$ and $p(c_i)$ represent the area, the gray level and the parent of the component c_i respectively. Therefore, the area of each component is weighted by the grey level difference with its parent. $A(c_i)$ denotes an attribute of the component c_i. All the components that their $A(c_i)$ attributes are located in an interval between b_{j-1} and b_j contribute to bin j of the PS. The histogram bin edges $\{b_0, ..., b_{max}\}$ of the desired attribute can be divided linearly or logarithmic. In the above example, the attribute $A(c_i)$ is defined as the area of the component c_i and it is distributed linearly between 0 and 50 producing a size pattern spectra. However, it can be any other type of attribute such as compactness value that produces shape pattern spectra.

Pattern spectra describes the probability of presence of a component with a certain size or shape in the image. It gives an estimation for the amount of details that is removed from the image after filtering its components. PS is a translation and scale invariant image descriptor that can be used for image classification and retrieval. PS is also rotation invariant if it is computed based on rotation invariant attributes. Several node attributes may be used to compute a multi-dimensional PS. The number of bins for each attribute is a parameter that should be selected for each dimension. Two dimensional PS based on elongation and entropy was used in [2] for satellite image retrieval. The authors also proposed local pattern spectra that computes PS in overlapped patches in different scales of the image. The final image descriptor is the aggregation of all computed PS.

4 Proposed Method

In conventional image retrieval, similar images to a query image are selected according to the distances between their descriptors. In this paper, we aim to extend that concept to patch retrieval where we look for some patches in the dataset that are similar to a query patch. The similar patches can be found in the same image of the query patch or in other images as shown in the example of Figure 1.

The naive method is to divide each image to a regular grid of patches with the same size of the query patch. Then, a sliding window is used and the similarity of each window with the query patch is computed. The computation time of this approach is proportional to the number of sliding windows that is too much for a large image. Notice that the size (and so the number) of the sliding windows depends on the user selected query patch. Therefore, it is not possible to compute and store the descriptors of the sliding windows in advance. In this section, we exploit the tree representation of the image and introduce the notion of fast pattern spectra descriptor to reduce the searching time. Finally the computation time is related to the number of tree nodes instead of number of the sliding windows.

4.1 Sub-Pattern Spectra

As explained in Section 3.1, pattern spectra is a histogram-like image descriptor that represents the distribution of connected components in the image. Each

image component contributes to one of the histogram bins and all image components build the whole histogram. Since a small patch in the image contains a subset of image components, we expect that it contributes to only a few number of histogram bins. This intuition is shown in Figure 4 where all the image components make the full histogram and the components in the small patch make a sparse histogram called here "sub-pattern spectra". We also notice that each image patch consists of a subset of nodes in the tree structure of the whole image and this subset of nodes does not necessarily form a single integrated sub-tree structure.

Fig. 4: (Left) Pattern spectra is a histogram-like distribution of image components. (Right) The components of each image patch contribute to a few numbers of histogram bins creating a sub-pattern spectra.

Having all related sub-pattern spectra computed, we can calculate the distance between the query patch and the sliding windows in the image to find the similar windows as depicted in Figure 5. The straightforward approach to compute sub-pattern spectra for each image patch is to directly compute pattern spectra on each window that are called here "local pattern spectra". Two issues arise from this approach. First, the components in the patch are not the same components in the whole image, because some of the image components are partially located in the sliding window and create new local components (Figure 6 Top). Second, there are many sliding windows in a large image and directly computing a pattern spectra for each of them will be time consuming. We address these problems by introducing "fast pattern spectra" in next section.

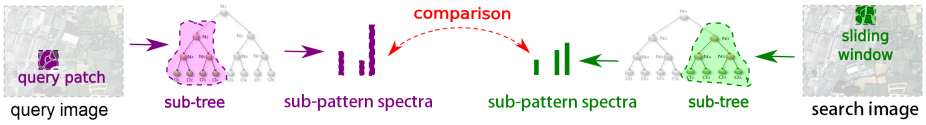


Fig. 5: Similarity of a query patch with a sliding window is computed based on the distance between their sub-pattern spectra.

4.2 Fast Pattern Spectra

Our motivation in this paper is to avoid directly computing a descriptor for each image patch. For this purpose we rely on the histogram nature of pattern spectra

and propose an indirect approach to compute the histogram in each patch. In this case, sub-pattern spectra consists of a subset of tree nodes in the whole image. Each node in the tree structure comes with some information such as center and bounding box of the corresponding component in the image. Therefore, when we select a patch from the image, we know which tree nodes are located inside it. We use these nodes to create our sub-pattern spectra and refer to it as fast pattern spectra (Figure 6 Bottom).

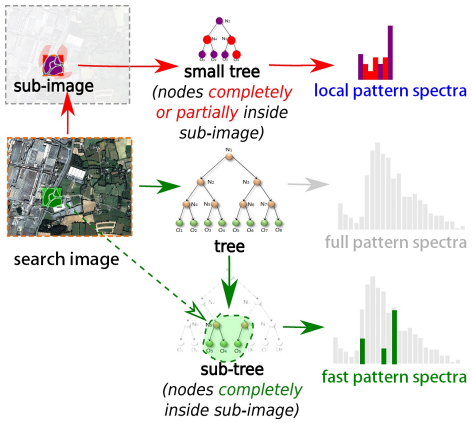


Fig. 6: Sub-pattern spectra of a given patch can be computed in two different ways. (Top) local pattern spectra: A small tree is built directly on the patch and its pattern spectra is computed. The nodes of this small tree may be different from the big tree of the whole image. (Bottom) fast pattern spectra: Tree representation of the whole image is constructed and only the nodes that are located inside the patch are selected to compute the histogram.

The proposed fast pattern spectra is an estimation of local pattern spectra. Instead of computing PS on the patch separately, we use the tree representation of the whole image to compute descriptors for all image patches. The computation time of the proposed descriptor is independent from the number of the patches that is determined by user selection. Instead, the processing time depends on the number of tree nodes that are fixed for each image.

Figure 7 shows the process of computing fast pattern spectra. We build a tree representation of a given image and compute the desired attribute for its nodes. According to Equation 3, pattern spectra is the sum of components volumes which are computed by multiplying the nodes areas by their gray level differences with their parents $v(c_i) = a(c_i)\big|k(c_i) - k\big(p(c_i)\big)\big|$. We calculate the volumes of all tree nodes and store them in a "tree table" along with nodes locations in the original image. Then, we compute a full pattern spectra and use it to assign a bin number to each tree node. The bin number is also stored in the tree table. When a user selects a query patch, we divide the image into sliding windows

based on that selection and create an empty histogram for each sliding window. Therefore, we have a three dimensional matrix $PS[x, y, b]$ spanned by image locations and histogram bins. Each tree node c_i contributes by its volume $v(c_i)$ to the histogram bin $b(c_i)$ in a specific location $(x(c_i), y(c_i))$.

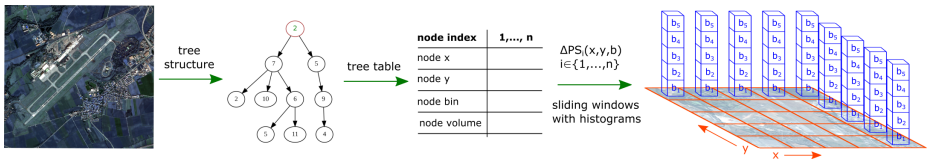


Fig. 7: Computation of fast pattern spectra consists of populating the local histograms for sliding windows. The algorithm traverses the tree structure and adds the contribution of each node to the corresponding histogram bin.

In order to compute fast pattern spectra, we traverse the tree table and simply add the volume of each node to the corresponding histogram bin. It is formulated in Equation 4 that is computed with complexity $O(n)$ where n is number of the tree nodes. In this way, we get the sub-pattern spectra for all sliding windows of the image in one pass of the tree structure. We point out that the tree table is pre-computed and fixed. Every time that the user selects a new query patch, only Equation 4 needs to be evaluated again.

$$PS[(x(c_i), y(c_i), b(c_i)] \mathrel{+}= v(c_i) \quad , \quad i = 1, ..., n \tag{4}$$

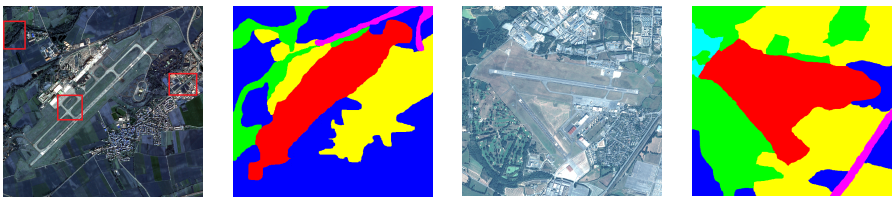
5 Experimental Results

In this section, we experimented our proposed fast pattern spectra for the application of patch retrieval on selected images and compare it with baseline local pattern spectra. All the experiments were done using Python with Higra package [14] in Windows 7 64-bit Operating System running on a laptop with Intel® Core™ i7 CPU @ 2 GHz and 8 GB RAM.

5.1 Dataset

The ideal dataset for patch retrieval, as we addressed in this paper, consists of a large aerial image with annotations for various landmarks on it. The available datasets for content based image retrieval such as "UC Merced Land Use Dataset" [15] are not suitable for this purpose since they consist of separate image patches with different classes. As far as we know, there is no available dataset with the desired annotations. Therefore, in order to conduct the reported experiments, we selected two satellite RGB images and manually annotated them with 6 classes including farm, forest, lake, airport, road and building as shown in Figure 8. We used Figure 8(a) as query image where the user selects its query

patches. Three user selected patches with different sizes and contents are marked in this image. Figure 8(c) was used to search for the similar patches. Each image patch may contain various pixel labels. We assigned the majority of pixel labels inside each patch as its class.



(a) Query image (b) Query labels (c) Search image (d) Search labels

Fig. 8: Selected satellite images for experiments. (a) 617×647 RGB image that user selects query patches from it. Three selected query patches are marked with red rectangles. (b) Ground truth labels for query image pixels showing 6 different classes. (c) 633×668 RGB image to search for similar patches (d) ground truth labels for search image pixels.

5.2 Experiment Settings

We compare the proposed fast pattern (fast-PS) spectra with the baseline local pattern spectra (local-PS). In both cases, the user selects an arbitrary patch from the query image. Then, the search image is divided to regular overlapped windows with the same size of the query patch and the stride of half size of the query patch horizontally and vertically. In the case of baseline local-PS, pattern spectra is computed independently on each search window. However, in the proposed fast-PS approach, a big tree is built on the whole search image and the pattern spectra is computed for all the search windows in parallel as detailed in Section 4.2.

In these experiments, pattern spectra is computed based on the max-tree structure that is built on the gray scale image. We use one dimensional pattern spectra based on $A(c_i) = \textit{compactness}$ attribute. The compactness of a tree node is defined as its area divided by the square of its perimeter. We divided nodes compactness linearly into 30 bins resulting to 30-dimensional descriptors for image patches. The similarity between descriptors are computed using χ^2 distance.

5.3 Results and Discussions

We selected three query patches with different sizes and contents as marked in Figure 8(a) and retrieved similar patches from Figure 8(c). For each case, we computed the average precision as explained in Section 2.2 using baseline and

proposed methods. The average precision and computation time are reported in Table 1. The second column of table shows the number of sliding windows in the search image that depends on the size of the selected query patch. As we expected, local-PS needs more computation time which depends on number of sliding windows. However, the computation time of fast-PS is almost constant since it depends only on the number of tree nodes. As we explained in Section 4.2, in proposed fast-PS we need to store a table of tree nodes attributes for search image. Last column of Table 1 reports the required time to compute this tree table. Notice that the tree table is independent from query patch. It is computed only one time for each search image and is stored along the image. The stored tree table can be used later for retrieval. Therefore, its computation time does not affect the retrieval time. However, it adds an extra cost for the storage. In our experiment the search image is 1.09 MB and the stored tree table is 4.42 MB.

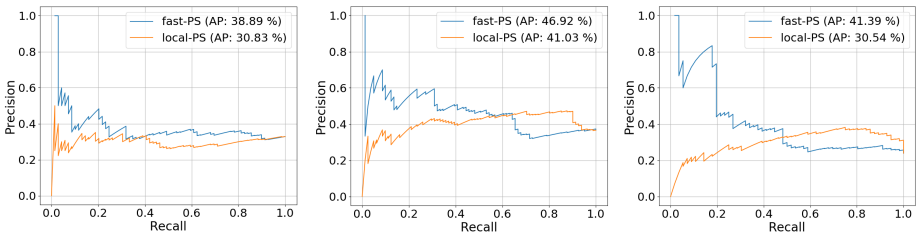
Table 1: Average precision and computation time for retrieval of three selected query patches using local pattern spectra and fast pattern spectra.

		Average Precision (%)		Computation Time (seconds)		
query size (pixels)	total patches	local-PS	fast-PS	local-PS	fast-PS	tree table
(a) 80×80	210	30.83	**38.89**	3.91	**1.97**	
(b) 70×90	221	41.03	**46.92**	4.12	**1.98**	2.86
(c) 90×70	234	30.54	**41.39**	4.38	**2.01**	

We basically proposed fast-PS (as its name suggests) as an approximation of local pattern spectra that is computed quickly over sliding windows in a large image. Table 1 shows that this novel descriptor actually works better than the original local-PS for the application of patch retrieval. A possible reason can be explained referring to Figure 6. When we crop a patch from a large image, it contains additional components that do not exist in the original image and may be irrelevant to its contents. In this case, fast-PS helps to rely on relevant components from original image and achieves better results. Figure 9 shows the precision-recall curves for three selected query patches. We observe that fast-PS performs better than local-PS especially at first ranks.

6 Conclusion and Future Lines

We extended the problem of content-based image retrieval and introduced a new application that we call patch retrieval where a user selects a query patch with arbitrary size from a large image and we look for similar patches in the same or other image. Available solutions from conventional image retrieval need to divide the image into many sliding windows and compute an image descriptor on each window separately. This approach takes a lot of time especially for



(a) 80 × 80 pixels (b) 70 × 90 pixels (c) 90 × 70 pixels

Fig. 9: Precision-recall curve and average precision (AP) for three selected query patches with different sizes using baseline local-PS and proposed fast-PS.

large images. We relied on pattern spectra that is an image descriptor based on tree representation of the image. We exploited the histogram nature of pattern spectra and proposed fast pattern spectra. It is computed for all sliding windows of the image in parallel and its processing time is independent from the number of the sliding windows. Interestingly, it is not only faster than local pattern spectra, but also achieves a higher average precision. However, the proposed method comes with a cost that needs to store an extra structure along with the original image. We showed that the fast pattern spectra is very promising on a simple dataset. More experiments on larger datasets are necessary to explore its full potential.

Our main goal in this work was to show the capability of fast-PS to offer interactive search in a large image when the size of the query patch is not known in advance. Although we found out that the average precision of fast-PS is higher than local-PS, we recall that our objective is not to surpass previous descriptors in term of accuracy. Rather, we relied on the histogram nature of pattern spectra to propose a novel descriptor that is computed quickly on a large image while achieving acceptable accuracy. We did not compare our proposed descriptor with state-of-the-art descriptors such as deep features and left it for future work. While other descriptors possibly lead to a higher accuracy, they cannot adapt to our framework for parallel computation since they have to be computed locally in each patch. We assume that the retrieval time is more important in a large-scale dataset where our proposed method is more efficient.

There is a lot of rooms for improvement of the proposed method. We used a max-tree structure and computed one dimensional pattern spectra based on the compactness attribute. However, our method is not restricted to the tree types and their attributes. We designed an efficient framework that can be realised with various types of pattern spectra to find the best combination of tree structures and attributes. Also, we simply calculated the distance between descriptors using χ^2. So, we did not rely on a learnt distance metric. We expect that learning a weighting matrix to compute the distances will lead to a higher accuracy.

Acknowledgment

This work was funded by DAJ-AR-NO-2018.0010814 project from CNES.

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. In: ACM Transactions on Graphics (ToG). vol. 28, p. 24. ACM (2009)
2. Bosilj, P., Aptoula, E., Lefèvre, S., Kijak, E.: Retrieval of remote sensing images with pattern spectra descriptors. ISPRS International Journal of Geo-Information **5**(12), 228 (2016)
3. Bosilj, P., Kijak, E., Lefèvre, S.: Partition and inclusion hierarchies of images: A comprehensive survey. Journal of Imaging **4**(2), 33 (2018)
4. Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P.: Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Transactions on Geoscience and Remote Sensing **54**(10), 6232–6251 (2016)
5. Cheng, G., Xie, X., Han, J., Guo, L., Xia, G.S.: Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **13**, 3735–3756 (2020)
6. Jones, R.: Component trees for image filtering and segmentation. In: IEEE Workshop on Nonlinear Signal and Image Processing, E. Coyle, Ed., Mackinac Island (1997)
7. Koestinger, M., Hirzer, M., Wohlhart, P., Roth, P.M., Bischof, H.: Large scale metric learning from equivalence constraints. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 2288–2295. IEEE (2012)
8. Li, W., Chen, C., Su, H., Du, Q.: Local binary patterns and extreme learning machine for hyperspectral imagery classification. IEEE Transactions on Geoscience and Remote Sensing **53**(7), 3681–3693 (2015)
9. Li, W., Du, Q.: Gabor-filtering-based nearest regularized subspace for hyperspectral image classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **7**(4), 1012–1022 (2014)
10. Liu, Y., Zhang, D., Lu, G., Ma, W.Y.: A survey of content-based image retrieval with high-level semantics. Pattern recognition **40**(1), 262–282 (2007)
11. Manning, C.D., Schütze, H., Raghavan, P.: Introduction to information retrieval. Cambridge university press (2008)
12. Maragos, P.: Pattern spectrum and multiscale shape representation. IEEE Transactions on pattern analysis and machine intelligence **11**(7), 701–716 (1989)
13. Paul, S., Pati, U.C.: Remote sensing optical image registration using modified uniform robust sift. IEEE Geoscience and Remote Sensing Letters **13**(9), 1300–1304 (2016)
14. Perret, B., Chierchia, G., Cousty, J., Guimarães, S., Kenmochi, Y., Najman, L.: Higra: Hierarchical graph analysis. SoftwareX **10**, 100335 (2019)
15. Yang, Y., Newsam, S.: Bag-of-visual-words and spatial extensions for land-use classification. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems. pp. 270–279 (2010)
16. Yu, H.Y., Sun, J.G., Liu, L.N., Wang, Y.H., Wang, Y.D.: Mser based shadow detection in high resolution remote sensing image. In: 2010 International Conference on Machine Learning and Cybernetics. vol. 2, pp. 780–783. IEEE (2010)