Computer Vision Winter Workshop 2006, Ondřej Chum, Vojtěch Franc (eds.)
Telč, Czech Republic, February 6–8
Czech Pattern Recognition Society

Texture segmentation through salient texture patches

Lech Szumilas[1], Branislav Mičušík[1], and Allan Hanbury[1]

[1]Vienna University of Technology
Pattern Recognition and Image Processing Group
Favoritenstr. 9/1832, A-1040 Vienna AUSTRIA
lech@prip.tuwien.ac.at, micusik@prip.tuwien.ac.at, hanbury@prip.tuwien.ac.at

Abstract

A novel approach to automatic, hierarchical texture detection in single images as a step towards image understanding is presented. First, the proposed method searches for alternating color patterns through hierarchical clustering of color pairs from adjacent, symmetrical image segments to localize salient regions in terms of color and texture. Second, the salient regions are fed as seeds to an image segmentation method based on min-cut/max-flow in the graph to localize the texture boundaries more accurately. The final result is a hierarchy of potential textured regions in the image useful for further object/texture recognition step. This work gives a proof of concept that the stable salient texture regions supported by a semi-automatic segmentation algorithm may provide fully automatic image segmentation into uniform color and/or texture regions. The results are presented on some images of natural scenes from the Berkeley database.

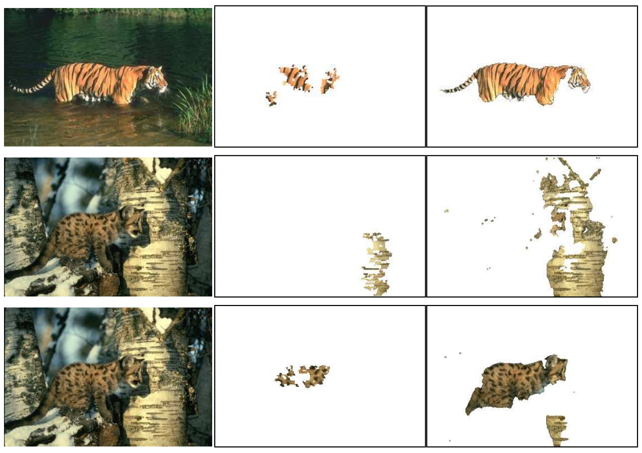

Figure 1: Automatic texture detection from single images. (a) An input image. (b) A mask provided by the proposed salient texture patches detection method. (c) Final result after the seed segmentation method using the mask from (b).

1 Introduction

Texture detection and classification play an important role in many image analysis tasks. Detection of texture boundaries is crucial for general image segmentation algorithms, while texture classification can provide extremely useful information for object recognition methods.

The term "texture" typically describes the presence of some regularity in a continuous image region, which may manifest itself as a spatially repeating color pattern or shape, but it is not defined how regular it must be. Several segmentation algorithms, either automatic or semi-automatic, exist which are capable of dividing an image into a uniform color and/or texture regions [4, 11, 8, 9, 7, 10, 12]. The primary problem these methods face is related to texture discrimination. The texture similarity cannot be precisely defined as it depends on the particular application – if we want to segment out the whole tiger from an image, then our texture similarity criteria will have to be different than if we want to segment out its legs, tail and head separately. This leads us to conclusion that the definition of a texture is task-dependent, which means that detection of texture should be knowledge based. Current segmentation methods do not allow the selection of a knowledge database for texture detection. In this work we attempt to combine a separate texture detection method with the seed segmentation method [8] to achieve a fully automated image segmentation, see Figure 1, further useful for knowledge based texture classification.

The method *Feature Co-occurrence Texture Detector* (FCTD) proposed in this paper detects textures at various "regularity levels" and produces a hierarchy of textures. The advantage of such an approach is the ability to detect less or more regular patterns automatically and then to make a knowledge based selection. Since the texture classification is not yet implemented, we devised a simple method to select the most regular textures from the hierarchy of detected ones, and use it as a texture marker for an image segmentation algorithm. This work is therefore a proof of concept of whether the combination of both methods works well.

A very common pattern found in textures is the alternation of two or more colors or luminance levels like for example the patterns covering tiger and zebra skins or the ripples on water. Therefore FCTD at the moment uses only color features to do a coarse texture detection. We assume that the texture consists of spatially mixed patches of uniform color, called *texture elements*. The general idea is to segment the image into small segments with a relatively uniform color and then find alternating color patterns among these segments. The pattern we are looking for is a group of similar color segments neighboring with another group of segments (or potentially more groups) but with different color.

The FCTD method can be divided into several steps: image segmentation, feature extraction, feature clustering and texture detection. After the FCTD a semi-automatic image segmentation method [8], based on maximal cut/minimal flow is used. This method requires user interaction to provide a representative template patch for the texture of interest. Here we avoid the problem of specifying the patch by using salient texture patches found by the FCTD. It turns out that the salient texture regions are large enough to capture the basic structure and color information of the texture. The result of the FCTD is a set of textures for which boundaries and color patterns are known.

The novelty of the whole approach described in this paper lies in showing how salient texture patches play a significant role in further seed image segmentation and possible (not discussed here) image understanding.

The structure of the paper is the following. First, the four steps of FCTD are described in Sections 2-5. These are image pre-segmentation, feature extraction, feature clustering, and texture detection. Second, the use of a the segmentation method is discussed in Section 6. Finally, the results with discussion and conclusion complete the paper.

2 Image Pre-segmentation

An image is pre-segmented into small segments, where each segment is a continuous, preferably convex shaped patch of similar color pixels. These constraints cause that segments do not exceed a texture elements size in the majority of cases and allow the use of average segment color as a primary feature.

FCTD uses a marker based version of the Watershed segmentation method, in which locations of markers are aligned with local symmetry maxima (see Figure 2). A radial symmetry $s(x_c, y_c)$ is calculated over the square area surrounding each pixel (x_c, y_c) as

$$s(x_c, y_c) = -\sum_{i=-R}^{R}\sum_{j=0}^{R}\Big|\boldsymbol{I}(x_c+i, y_c+j)-\boldsymbol{I}(x_c-i, y_c-j)\Big| \quad (1)$$

where $\boldsymbol{I}(x_c + i, y_c + j)$ is an image pixel at coordinates $(x_c + i, y_c + j)$.

This symmetry measure has several useful properties which help to achieve convex shaped and low color gradient segments:

- The symmetry measure s reaches maximum (equal to 0) if all corresponding pixel pairs $(x_c + i, y_c + j)$ and $(x_c - i, y_c - j)$ are identical.
- Symmetry is maximized at the center of radially symmetric shapes (like a filled circle, star, etc.) or along their symmetry axis (for elongated shapes).
- An edge irregularity produces more symmetry maxima and more segments, which in turn prevents generation of segments with complex boundaries.

The proposed symmetry measure tends to generate an excess of local maxima points, but it does not miss any symmetrical regions in the image (assuming sufficiently large R). Other symmetry transforms exist [6], intended primarily as interest point detectors. The comparison of texture detection using other symmetry measures is part of the future work.

The final shape of segments depends also on the image gradient used for the Watershed, therefore even the use of symmetry does not fully guarantee shape convexity. However relatively small irregularities in the segment boundary have no great effect on the final result. All the results in this paper are produced using a color gradient, which is an average of gradients $\Big|\nabla\boldsymbol{I}_c\Big|$ calculated over each color channel c in the CIELAB color space:

$$\Big|\nabla\boldsymbol{I}_c\Big| = \Big|\frac{\partial\boldsymbol{I}_c}{\partial x} + \frac{\partial\boldsymbol{I}_c}{\partial y}i\Big| \quad (2)$$

We are able to decrease the overall number of segments by applying symmetry maxima as Watershed seeds (see Figure 2), however in the majority of cases images are still over-segmented, i.e. texture elements contain multiple segments. If we can avoid over-segmentation we could use segment (and so texture element) geometry as a feature for clustering. Section 8 discusses possible improvements allowing the use of additional features.

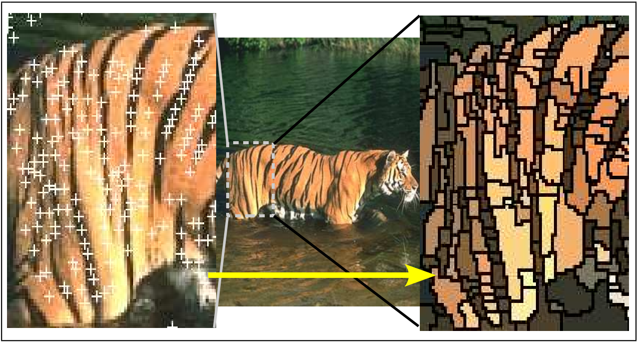

Figure 2: Example of symmetry maxima (left) and the resulting Watershed segmentation (right).

3 Feature Extraction

A feature vector is calculated for each segment resulting from the Watershed segmentation. The feature vector contains the color of the segment it is assigned to and the color or colors of segments surrounding it. Such features allow one to analyze alternating color patterns. There are several possibilities as to how colors are calculated. The most basic option would be a feature vector consisting of two most different colors (six real values) (option 1 in Figure 3). The color difference is measured as the Euclidean distance in three dimensional color space (e.g. CIELAB).

Another option is to average the color of the surrounding segments for which the color distance to the current segment is larger or equal to $0.5d_{max}$, where d_{max} is a distance between two most different colors (option 2 in Figure 3). It is also possible to use more than one color for the descrip-

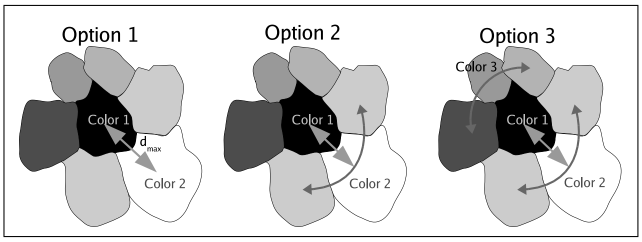

Figure 3: Extraction of color features from neighboring Watershed segments

tion of the surrounding segments (option 3 in Figure 3). The results in this work were generated using "option 2".

4 Feature Clustering

Color clustering has been applied in the past for color reduction and segmentation [5] as well as other problems, but it was always performed in three dimensional space (single color). Since our goal is detection of color patterns we cluster color pairs, which means using a six dimensional space.

Before the clustering starts, color pairs are sorted by their luminance. The color with the lower luminance value always occupies the first three elements. This prevents two clusters per pattern forming due to different color orders.

FCTD uses agglomerative hierarchical clustering [3] based on the average linkage to build a cluster hierarchy of color pairs (and corresponding Watershed segments). Hierarchical clustering performs a cluster reduction by pairing nearest clusters when progressing from level N to level $N-1$. This process is repeated until only two clusters remain at level 2.

Figure 4 is an example of a color-pair cluster hierarchy. At clustering level 2 the whole image is divided into two relatively large clusters. More significant clusters appear at level 4. Also a number of segments on the boundary between the tiger and water form a separate cluster. At the tiger boundary bright tiger segments and dark water segments create the most different color pairs. It is possible to re-attach separated boundary clusters to the tiger body based on statistical analysis of the segment features and their position. We know that a relatively small number of separated boundary segments has a very similar color to the segments belonging to the tiger body. Also most of the boundary segments are adjacent to the segments belonging to the tiger body, representing both dark and bright stripes. This allows us to assume that boundary segments belong to the tiger body as well. This example shows that different textures may be better detected at different clustering levels, as the feature spread varies between textures.

The example from Figure 4 shows that different textures may be better detected at different clustering levels. This is caused by the fact that feature spread varies between textures and that feature spread inside clusters decreases at higher clustering levels. For example at clustering level 3, the water is represented by only a single segment, but at clustering level 4 it is divided into two clusters and at level 5 it is divided into 5 relatively big clusters. At the same time the tiger body is divided into more smaller clusters at higher clustering levels, which makes them less usable in this case.

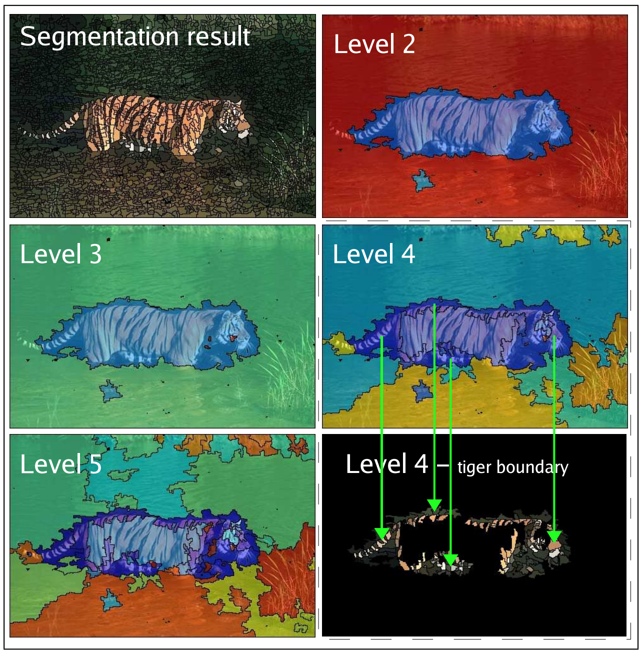

Figure 4: Example of a cluster hierarchy ($R = 5$, gradient calculated using luminance). It is recommended that the color version of this image s viewed.

The main advantage of hierarchical clustering is that it enables texture detection at different clustering levels. We can also observe that some image regions remain stable in a number of different clustering levels, i.e belong to a single cluster across multiple clustering levels, like most of the tiger body or a number of water regions, which may be a useful tool for texture detection and classification.

5 Texture Detection

The texture detection step attempts to discover color patterns in each cluster separately. It means that only features belonging to a single cluster are clustered again, but this time color pairs in feature vectors are not sorted. This approach divides a texture into two clusters whose segments neighbor each other. Following this path we can say that if two or more clusters have a high percentage of segments neighboring each other, then we should treat them as a texture. The texture is detected then if the length of the boundary B_{kl} in the image between groups of pixels belonging to two clusters k and l, relative to their total boundary length B_k and B_l exceeds a cluster co-occurrence ratio threshold τ (in range 0 to 1):

$$\frac{B_{kl}}{B_k} \geq \tau \quad \wedge \quad \frac{B_{lk}}{B_l} \geq \tau \tag{3}$$

Typical values for τ vary in the range from 0.7 to 0.9 due to the fact that clusters are usually of different size and we also allow for a small number of segments in both clusters to not be neighbors of the segments in the other cluster.

Re-clustering is performed until the adjacency condition in Equation 3 is fulfilled or the maximum clustering level is reached, which means that no textures were detected. If the adjacency condition is fulfilled then a texture consisting of two adjacent clusters is detected. The texture detection is continued further in an attempt to divide the detected texture

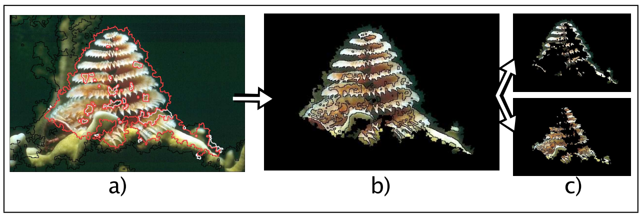

Figure 5: Textures detected in image (a) at image clustering levels 3 (b) and 4 (c).

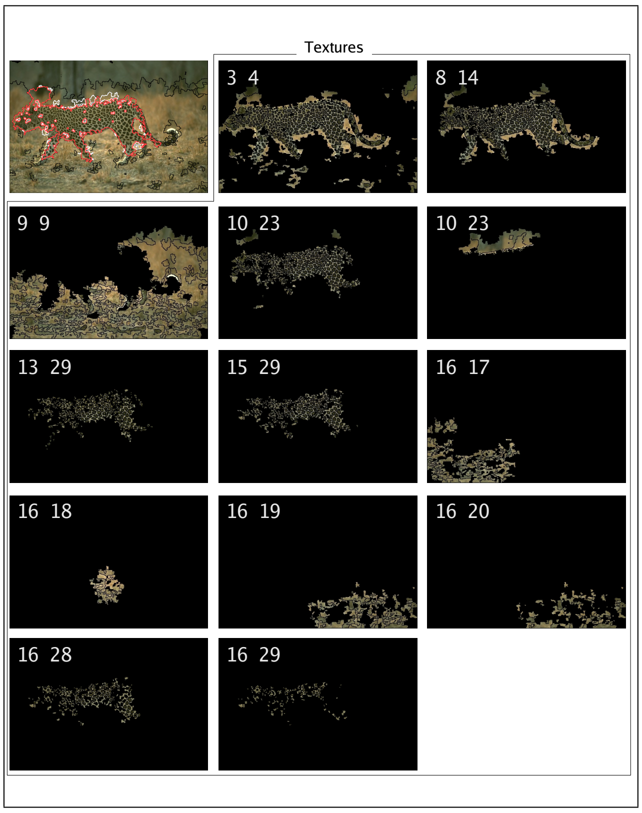

Figure 6: An example of texture detection from an image of size 481x321 pixels. Textures are detected at various clustering levels shown on the texture images. A total of 13 textures were detected. Note that not all detected textures become seeds for the segmentation method.

(two adjacent clusters) into 2 textures (4 clusters altogether). It means that up to three textures can be detected from a single cluster, as in the example in Figure 5. This way we provide a choice of less and more generalized results for further processing.

The texture detection step produces multiple textures from all clustering levels. Figures 6, 7 and 8 show examples of textures detected at various clustering levels. Each texture is accompanied by two numbers – the first number indicates the image clustering level, while the second number indicates the texture clustering level. Redundant (identical) textures are automatically removed. The rest is used to build a texture hierarchy tree. The texture which is not a subset of another texture forms a tree root node. Then the textures which are subsets of the root textures are attached to the corresponding root nodes. It may happen that texture 1 is a subset of texture 2, which is a subset of texture 3. In that case only texture 2 is attached to texture 3 and texture 1 is

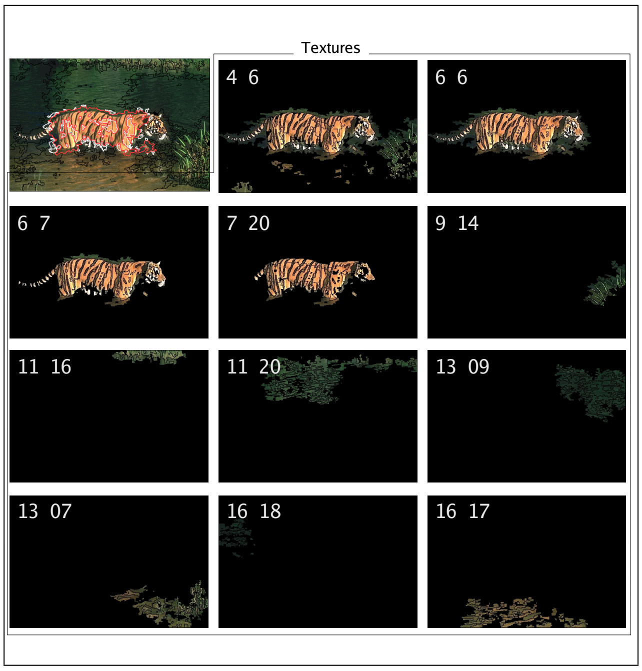

Figure 7: An example of texture detection from an image of size 481x321 pixels. Total of 34 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

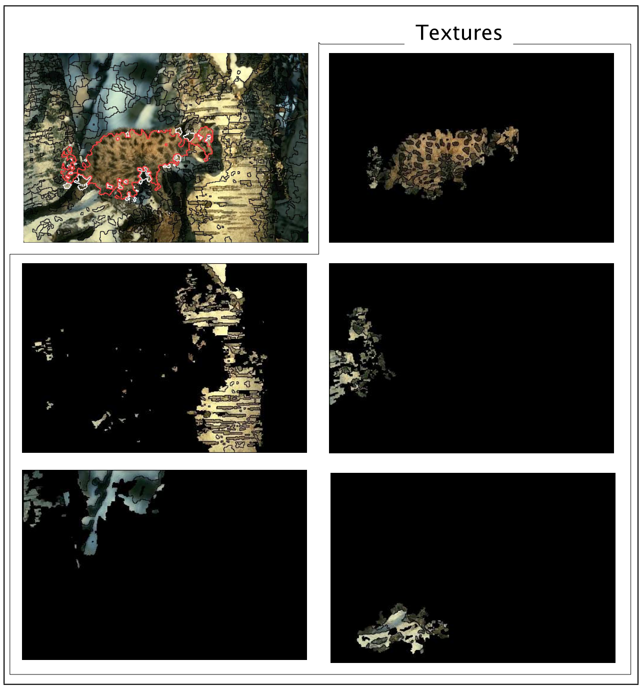

Figure 8: An example of texture detection from an image of size 481x321 pixels. Total of 24 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

attached only to texture 2. An example of a texture hierarchy tree is shown in Figure 9.

6 Seed Segmentation

We use a seed segmentation technique [8] based on the interactive graph cut method [1] improved to colour and tex-

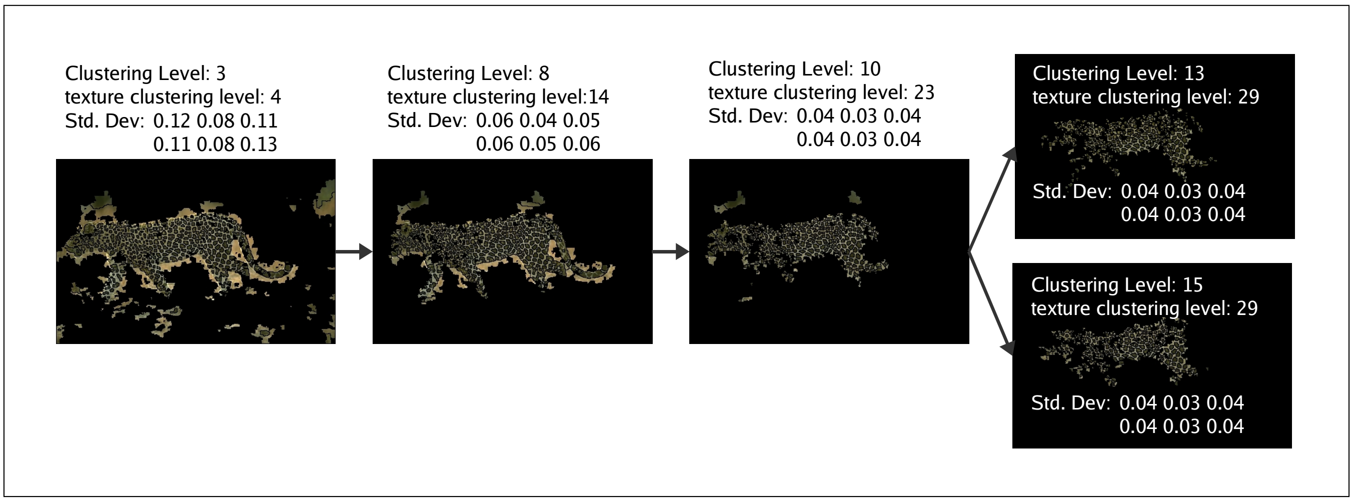

Figure 9: Example of texture hierarchy tree. The label 'Std. Dev.' describes standard deviation of the color pairs in each texture – the top row describes standard deviation for three color components of the first color, while the second raw provides same values for the second color. All color values were normalised to range 0–1.

tured images. The core of the segmentation method is based on minimizing "Gibbs" energy through an efficient algorithm [2] for finding the min-cut/max-flow in a graph.

The segmentation method as it uses a color and texture gradient in the process of building the graph respresenting an image, can specify the boundaries of textures more accurately than the FCTD. On the other hand, the drawback of the method lies in the need of a priori information about the region of interest. Therefore user interaction through establishing seeds in the image is required. In this paper we avoid need of the user interaction and feed salient texture patches found by the FCTD to the segmentation method as seeds. It turns out that the FCTD gives good representative samples allowing the texture to be enlarged and the texture boundaries to be localised more accurately, as shown Figures 1 and 10.

The seed segmentation method is run on all salient texture patches found by FCTD yielding a hierarchy of regions in the image. The regions can overlap and not all image pixels are assigned to any region. To further handle the regions, e.g. to merge them, texture recognition is needed which is out of scope of this paper.

7 Discussion of the Results

The final result of the texture detection depends on a few parameters: R, τ and the type of image gradient used for segmentation. All results shown in this section were obtained using $R = 5$, normalized CIELAB color space and averaged color gradient. These parameters primarily affect the boundary of the detected textures. Parameter R has the highest influence – it varied from 3 to 16 pixels for the image size used in the experiments. Despite small differences in texture boundaries, the same color patterns were discovered in each case. Other results are presented in [13].

The use of color-pair based features sometimes leads to inaccuracies in the detected texture boundaries, when segments at the texture boundary have other neighbors with more different colors, not belonging to the texture. This problem however can be solved as discussed in Section 4.

Figure 10 shows final results after applying the seed segmentation method on the salient texture patches detected by the FCTD. Not all textured region extracted from the input image are shown. We chose manually the most significant textures only to show how they are captured by the proposed technique. As it was stated above the proposed automatic procedure produces many potential textured regions in hierarchical manner. To prune the hierarchy and keep only meaningful textures or textures of interest a texture recognition algorithm has to follow.

The whole texture salient patches detection time did not exceed 60 seconds on P4@2.8GHz. The seed segmentation method applied afterwards takes another 60 seconds on reduced size 250x375 pixels image. We reduced the size for the seed segmentation method due to the large memory and computational demand on the original resolution.

8 Conclusion

FCTD is a method for fully automatic detection of textures made up of alternating colors in two dimensional still images. It is based on a novel idea of color pair hierarchical clustering, where color pairs are extracted from neighboring Watershed segments. The method proved capable of detecting a variety of color patterns, at different accuracy levels, from natural scenes. The results obtained lead to the conclusion that the analysis of color patterns alone allows the detection of the majority of the significant textures in natural scenes, although texture boundaries are not very precise. This imprecision, as was shown can be solved by feeding the salient texture patches found to the seed segmentation method. The results show reasonable performance on the Berkeley dataset.

Future work includes several improvements, including the use of improved segmentation and adaptive segment merging, which allow for geometrical features to be used. Currently, image segmentation produces rather small and dense segments, which often divide texture elements into several parts. Using current results, the segments within detected color patterns will be merged if their color difference is smaller than the color spread in the pattern. Texture detection will be repeated after segment merging, but this time with geometry features (such as segment elongation) added. Images will also be analyzed at different scales to allow detection of very small texture elements (less than 3 pixels in diameter).

Acknowledgement

This research was done thanks to the following projects: the Austrian Science Foundation (FWF) under grant SESAME (P17189-N04), and the European Union Network of Excellence MUSCLE (FP6-507752).

References

[1] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proc. ICCV*, pages 105–112, 2001.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[3] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[4] C.C. Fowlkes, D.R. Martin, and J. Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *Proc. CVPR*, pages II: 54–61, 2003.

[5] G. Li, Ch. An, J. Pang, M. Tan, and X. Tu. Color image adaptive clustering segmentation. In *Proc. Image and Graphics Conference*, pages 104–107, 2004.

[6] G. Loy and A. Zelinsky. Fast radial symmetry for detecting points of interest. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):959–973, 2003.

[7] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43:7–27, 2001.

[8] B. Mičušík and A. Hanbury. Supervised texture detection in images. In *Proc. Conference on Computer Analysis of Images and Patterns (CAIP)*, 2005.

[9] B. Mičušík and A. Hanbury. Automatic image segmentation by positioning a seed. In *Proc. European Conference on Computer Vision (ECCV)*, to appear 2006.

[10] N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 46(3):223–247, 2002.

[11] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[12] B. Sumengen, B.S. Manjunath, and C. Kenney. Image segmentation using curve evolution. In *Conference on Pattern Recognition*, volume 2, 2002.

[13] L. Szumilas and A. Hanbury. Segment feature co-ocurrence based texture detection. Technical report, PRIP, Vienna University of Technology, 2005.

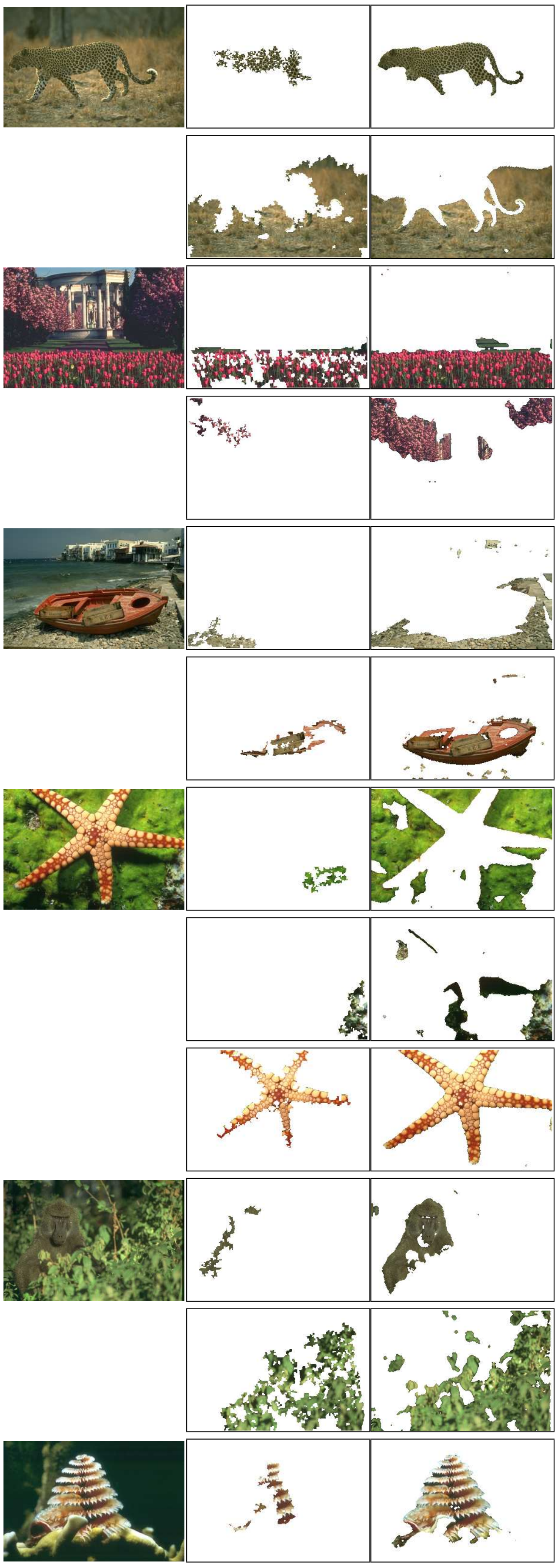

Figure 10: Some results for automatic texture detection from single images. (a) An input image. (b) A mask provided by proposed texture detection method. (c) Final result after the seed segmentation method using the mask from (b).