Color Pair Clustering for Texture Detection*

Lech Szumilas and Allan Hanbury

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9-11/1832, A-1040 Vienna, Austria
{lech, hanbury}@prip.tuwien.ac.at

Abstract. A novel approach to the extraction of image regions of uniform color and its application to automatic texture detection is discussed. The method searches for alternating color patterns, through hierarchical clustering of color pairs from adjacent image regions. The final result is a hierarchy of texture regions, described by their boundaries and a set of features, detected at multiple accuracy levels. The results are presented on some images of natural scenes from the Berkeley segmentation dataset and benchmark.

1 Introduction

Automatic texture detection can play an important role in many image analysis tasks, such as image segmentation or object recognition.

One of the most fundamental properties appearing in many textures is the presence of some regularity in a contiguous image region [1], which may manifest itself as a spatially repeating color pattern or shape. It is however not possible to find a strict regularity criterion which would discriminate between textures and non-textures. Several segmentation algorithms, either automatic or semi-automatic, aim at dividing an image into uniform color and/or texture regions [2,3,4,5,6,7]. The primary problem that these methods face is related to a fixed criterion for texture discrimination (usually texton based). In reality however, the texture similarity depends on the particular application or associated knowledge (see Figure 1).

The *Feature Co-occurrence Texture Detector* (FCTD) proposed in this paper detects potential texture regions at various "regularity levels", which can be later used for texture classification. 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.

A very common pattern found in textures is the alternation of two or more colors or luminance levels. Therefore the FCTD at present uses only color features to do a coarse texture detection. We assume that the texture consists of spatially alternating patches of uniform color and simple shape, which we call *texture elements*. The general idea is to segment the image into small regions with a relatively uniform color and then to find alternating color patterns among those segments. The pattern we are looking

* This work was supported by the European Union Network of Excellence MUSCLE (FP6-507752) and the Austrian Science Foundation (FWF) under grant SESAME (P17189-N04).

G. Bebis et al. (Eds.): ISVC 2006, LNCS 4292, pp. 255–264, 2006.

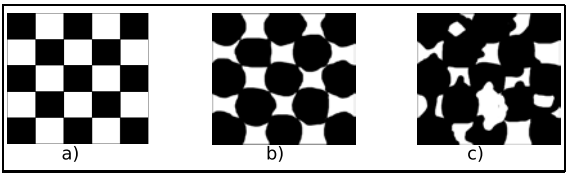

Fig. 1. All three images above may be classified as different or identical textures depending on the texture discrimination criteria. If only the color pattern is considered then all textures are identical, however when the geometrical pattern is also considered then these textures become different.

for is a group of similar color segments neighboring with another group of segments (or potentially more groups) with different colors. The result of the FCTD is a set of textures for which boundaries and color patterns are known.

We also present a novel image segmentation method, the Dominant Color Image Segmentator (DCIS), which aims at extracting uniform color regions in the image. The FCTD method uses the DCIS segmentation to extract color features.

The following sections describe all the steps leading to detection of texture regions: image segmentation (Section 2), feature extraction (Section 3), feature clustering (Section 4) and texture detection (Section 5).

2 Image Segmentation

The primary goal of the image segmentation step is to generate uniform color regions approximating the texture elements, which will be used for extracting color pairs from adjacent regions. We do not aim at obtaining a perfect representation of the texture elements, which would not be possible. We nevertheless want to avoid too much over-segmentation, which could lead to a region of almost uniform color being internally divided into more than one segment. Such an over-segmentation would destroy the texture element adjacency information that we use.

Several well known methods, which perform a general color segmentation of the image are available:

- Watershed [8] – uses the image gradient as a topographical representation of the image. It is however sensitive to noise and weak gradients, which often lead to a significant over-segmentation.
- Waterfall [9] – allows the reduction of the number of segments generated by the Watershed, however the final result depends on the number of iterations executed, which is rather difficult to estimate for particular image.
- Mean-shift segmentation [10] – is capable of delivering segments which are a good representation of uniform color regions in the image, but the final result is very sensitive to a set of input parameters. There is no easy way to adjust them to suit each particular image.
- Normalized Cuts - in [11] it is used to create a “superpixel” or segments which aim at being local, coherent and representing the image structure at higher than pixel

level. There are two ways to control the number of regions produced: specifying the number of regions in advance [12] or choosing a threshold on the normalized cut value [12]. Neither of these methods is optimal for finding the texture elements.

It is not possible to list all image segmentation methods here, but so far we did not find any method which provide a good representation of uniform color regions for an arbitrary image using stable set of parameters.

We propose to use a novel image segmentation method called "Dominant Color Image Segmentator" (DCIS), which is less sensitive to noise and adaptively discards weak gradients in the image using locally dominant colors for segment boundary detection.

DCIS can be divided into four steps:

1. Selecting uniform color region central points (also called markers).
2. Obtaining adjacency between region central points
3. Detecting locally dominant colors for each point and its neighborhood
4. Generating region labels

The final segmentation depends in a large degree on the first step, the selection of region central points. For this purpose a radial symmetry measure is calculated over the square area surrounding each pixel and region central points are aligned with local symmetry maxima (see Figure 2). The radial symmetry measure is defined as:

$$S(x,y) = -\sum_{i=-R}^{R}\sum_{j=0}^{R}\left|\boldsymbol{I}(x+i,y+j) - \boldsymbol{I}(x-i,y-j)\right| \tag{1}$$

where $\boldsymbol{I}(x+i,y+j)$ is an image pixel at coordinates $(x+i,y+j)$ and R defines the image window size used for symmetry measure calculation to be a $(2R+1)\times(2R+1)$ rectangle.

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

- The symmetry measure s reaches a 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 the 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 [13], intended primarily as interest point detectors. They therefore do not detect as many interest points as is usually needed for representing texture elements.

The second step of DCIS is obtaining adjacency between markers, which is needed for detecting locally dominant colors. Because of dense marker distribution, adjacency can be captured by using a Voronoi tessellation of markers and analysis of the boundaries between tessellation regions. As a result, each marker is assigned a list of adjacent markers with which it shares a boundary between corresponding tessellation regions.

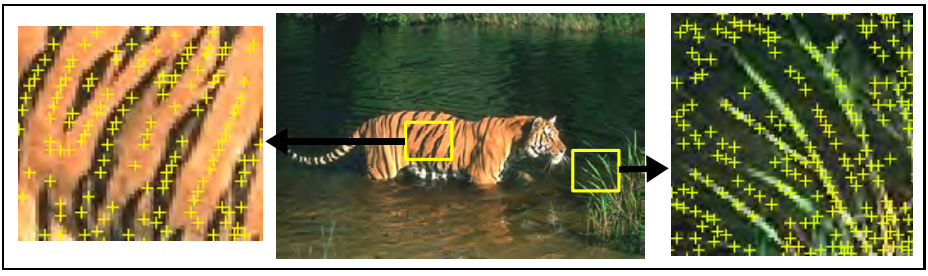

Fig. 2. Example of symmetry maxima location

Extraction of locally dominant colors is performed for each marker separately by agglomerative hierarchical clustering [14] of the colors within each tessellation region and its immediate neighborhood. The clustering uses an average linkage and the Euclidean distances to calculate distance between clusters. The inputs to the clustering algorithm are the colors of all pixels belonging to the tessellation region containing the marker and all adjacent regions. We assume that in the majority of images of natural scenes, the number of dominant colors around each marker is very low, between two and four colors. Therefore for each marker only three or four clusters are used to represent locally dominant colors. The clusters are extracted from the clustering tree level below the one containing two clusters, which can contain three or four clusters depending on the input data - the number of clusters is dictated by the automatically built clustering tree structure.

The process of region boundary selection for a single marker can be best illustrated on the example in Figure 3. The marker m_i has seven adjacent markers $m_a(i, 1..7)$, obtained through marker tessellation. Markers m_i, $m_{a(i,1)}$ and $m_{a(i,2)}$ are located within the uniform color 1 area, while the rest of the markers are located within the uniform color 2 area. Let us also assume that there is a transition area between color 1 and color 2, as is frequently the case in images of natural scenes. After color clustering of pixels belonging to all tessellation regions we obtain three clusters, corresponding to color 1, color 2 and the color transition areas. Cluster 1 is assigned to marker m_i. Since markers $m_{a(i,1)}$ and $m_{a(i,2)}$ also belong to the cluster 1 the tessellation boundaries between region m_i and these markers are preserved. Thus any weak gradients between these regions are ignored. However, the other markers belong to a different cluster, which means that they represent areas with significantly different color. The final boundary of the m_i region is therefore a combination of the cluster 1 boundary with tessellation boundaries inside cluster 1. This boundary selection is performed for each marker in turn.

The boundary selection strategy may leave some pixels unclaimed by both segments, which is usually the case for pixels belonging to the color transition area or small areas with significantly different color not represented by the marker. It is an advantage as it allows for more precise average segment color estimation, since outliers and boundary values are automatically rejected. It does not however allow one to obtain the adjacency of the segment. Fortunately it is relatively easy to assign unclaimed pixels to the segments adjacent to unclaimed pixel regions. In such cases each pixel in the unclaimed

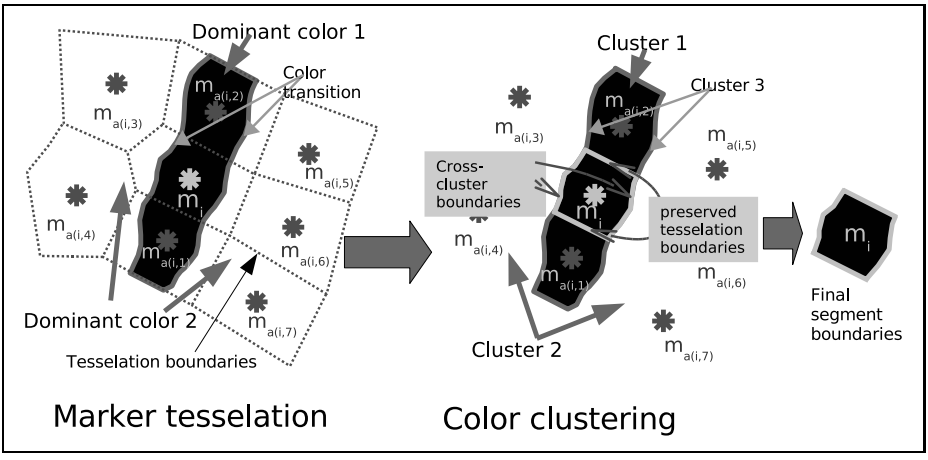

Fig. 3. Example of DCIS segment boundaries selection

region is assigned to the most similar neighboring segment producing a refined segmentation. The refined segmentation is not used for segment feature extraction.

Figure 4 shows differences between the Watershed and DCIS segmentations. One can see that the Watershed produces many more segments and that their boundaries react to weak gradients/noise. DCIS segments are much better suited to the extraction of color pairs related to texture elements. Even though DCIS also over-segments images, e.g. stripes contain multiple similar color segments, each segment of a dark stripe neighbors with a segment of a bright stripe, which is sufficient for extracting texture element related color pairs. It is also worth noticing that the DCIS refined segmentation is a de-noised version of the original image, while the Watershed preserves the noise in the form of small segments.

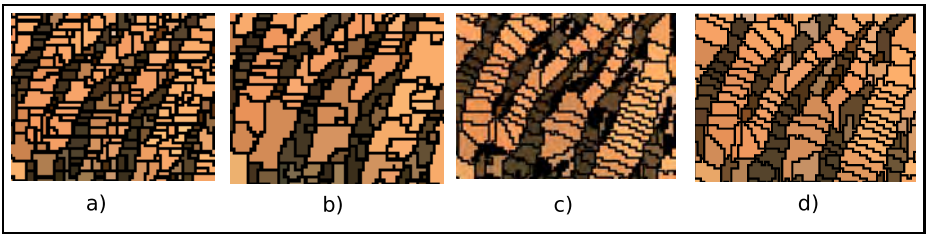

Fig. 4. Examples of various segmentation methods (crop of the tiger body from Figure 2): a) Watershed, b) symmetry marker based Watershed, c) DCIS, d) refined DCIS

Let us summarize the advantages of the DCIS algorithm:

– insensitive to noise (can be used for image de-noising) and locally weak gradients
– segments and extracts local dominant colors in a single pass

– Requires a single parameter R, but in practice it is adjusted to the image resolution. In all our tests we use $R = 5$ for an image resolution of 480×320 pixels.

Hierarchical clustering is computationally expensive. The DCIS execution time for 480x320 pixel images reached approximately 20 seconds on 2.4GHz AMD processor which is significantly longer than the Watershed.

3 Feature Extraction

A feature vector is assigned to each segment resulting from the DCIS segmentation. The feature vector contains the color of the segment to which it is assigned 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, but in our case we use dominant colors generated by the DCIS segmentation. DCIS extracts 3 or 4 clusters for selecting segment boundaries, but 2 clusters can easily be extracted from the next level in the clustering tree. Those two clusters represent two colors, one related to the current segment and adjacent segments containing similar colors, and the second color representing all adjacent segments containing color which is not similar to the current segment. Although it is possible to use more than two colors, all our experiments were carried out with two color feature vectors.

4 Feature Clustering

Color clustering has been applied in the past for color reduction and segmentation [15] as well as other problems, but it was always performed in three dimensional space (single color clustering). Since our goal is the 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 [14] based on the average linkage to build a cluster hierarchy of color pairs (and corresponding DCIS 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 5 is an example of a color-pair cluster hierarchy. At clustering level 2 the whole image is divided into two large regions. More significant clusters appear at level 4. Also a number of segments on the boundary between the tiger and water form a separate regions. 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 have 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. 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 region, but at clustering level 4 it is divided into two regions and at level 5 into 5 regions. At the same time the tiger body is divided into smaller regions at higher clustering levels, which makes them less usable in this case.

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.

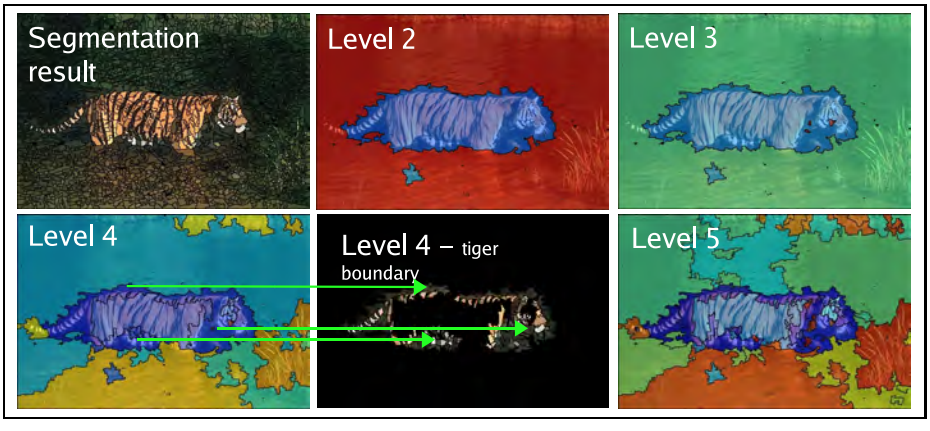

Fig. 5. Example of a cluster hierarchy ($R = 5$)

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. To distinguish between the two types of clustering done, we refer to the clustering level obtained by the algorithm of Section 4 as *image clustering level* and the clustering level obtained by the clustering described in this section as the *texture clustering level*.

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 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):

$$\frac{B_{kl}}{B_k} \geq \tau \quad \wedge \quad \frac{B_{lk}}{B_l} \geq \tau \tag{2}$$

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 that 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 2 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 (two adjacent clusters) into 2 textures (4 clusters altogether). It means that up to 3 textures can be detected from a single image cluster. 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. Figure 6 shows examples of the most representative textures detected at various clustering levels on images from the Berkeley segmentation dataset and benchmark [12]. Further results are presented in [16].

The final result of the texture detection depends on two parameters: R and τ. All results shown were obtained using $R = 5$ and $\tau = 0.7$. These parameters primarily affect the boundary of the detected textures. Figure 6 shows only the most representative textures detected.

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.

FCTD in many cases provides similar texture boundaries to the segments produced by methods described in [3,7] on the same image.

6 Conclusion

We have presented two novel algorithms performing color image segmentation (DCIS) and texture region detection (FCTD).

DCIS is a general method for finding uniform dominant color regions. It has a single stable parameter, is noise insensitive and adapts to a local contrast.

FCTD is a detector of textures made up of alternating colors in still images. It is based on a novel idea of color pair hierarchical clustering, where color pairs are extracted from neighboring uniform color segments. The method proved capable of detecting most textures consisting of alternating color patterns, at different regularities, from natural scenes. FCTD is based on an idea of detecting textures at different regularity levels, which is the primary difference with respect to existing methods already cited in this paper. This flexibility allows for detection of less and more regular textures within the same image. For example many existing methods cannot detect both the tiger and the water in Figure 2 at the same time. Though currently FCTD uses only color features,

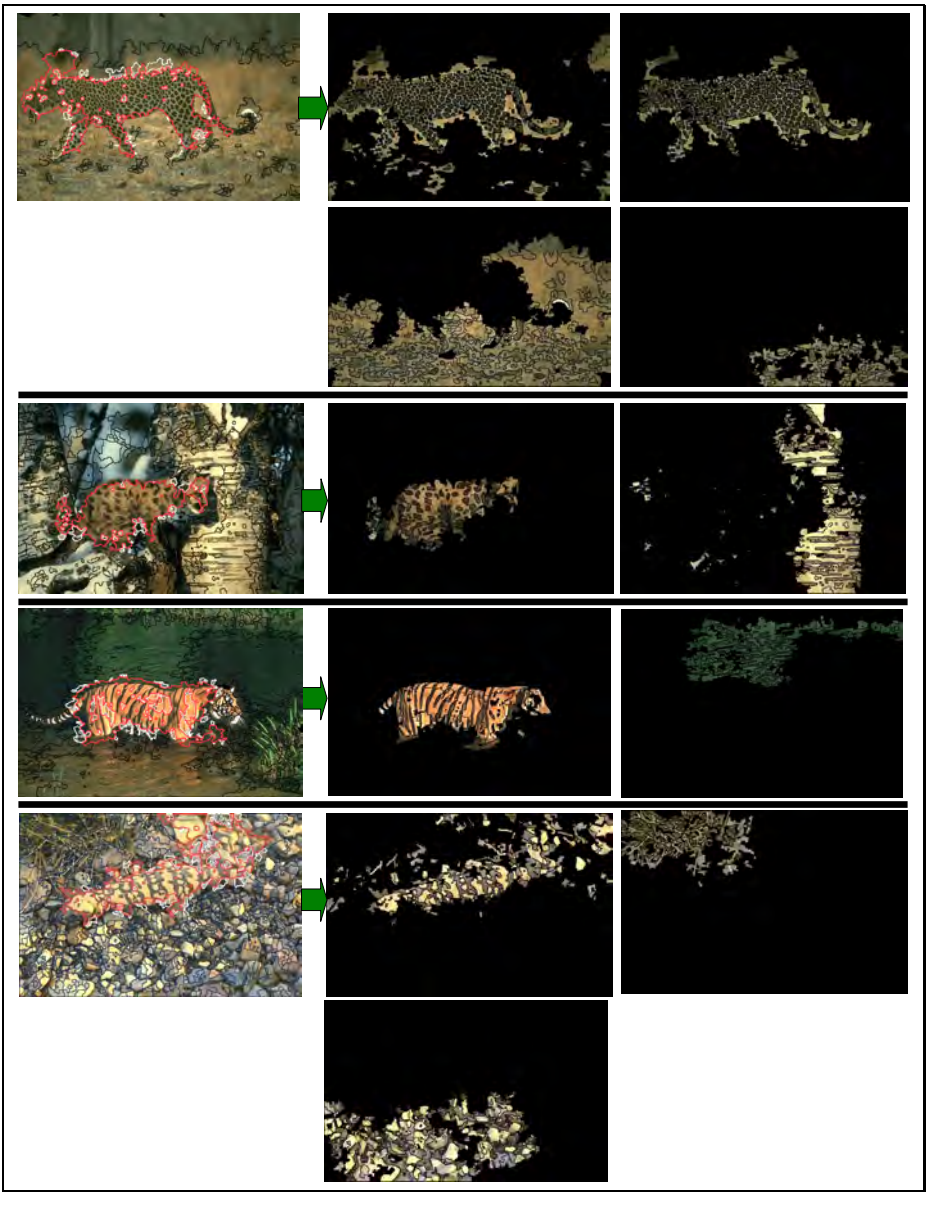

Fig. 6. An example of texture detection from images of size 481x321 pixels (in the first column). The regions corresponding to the detected textures are shown in the second and third column.

which limits its application as a texture detector, the method can be extended to use geometrical features.

Future work includes the application of geometrical features in addition to color, to make FCTD more insensitive to luminance changes and particularly shadows. We are

also going to use detected texture regions for texture classification and use the classification results for selecting regions best representing textures.

References

1. Chetverikov, D.: Pattern regularity as a visual key. Image and Vision Computing **18** (2000) 975–985
2. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. International Journal of Computer Vision **43** (2001) 7–27
3. Mičušík, B., Hanbury, A.: Steerable semi-automatic segmentation of textured images. In: Proc. Scandinavian Conference on Image Analysis (SCIA). (2005) 35–44
4. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI **22**(8) (2000) 888–905
5. Fowlkes, C., Martin, D., Malik, J.: Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In: Proc. CVPR. (2003) II: 54–61
6. Paragios, N., Deriche, R.: Geodesic active regions and level set methods for supervised texture segmentation. IJCV **46**(3) (2002) 223–247
7. Sumengen, B., Manjunath, B.S., Kenney, C.: Image segmentation using curve evolution and region stability. In: Proc. International Conference on Pattern Recognition. Volume 2. (2002)
8. Soille, P.: Morphological Image Analysis. Springer (2002)
9. Marcotegui, B., Beucher, S.: Fast implementation of waterfall based on graphs. In: Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'05. (2005) 177–186
10. Comanicu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. In: IEEE Trans. Pattern Anal. Machine Intell. Volume 24. (2002) 603–619
11. Ren, X., Malik, J.: Learning a classification model for segmentation. In: International Conference on Computer Vision. (2003) 10–17
12. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision. Volume 2. (2001) 416–423
13. Loy, G., Zelinsky, A.: Fast radial symmetry for detecting points of interest. Pattern Analysis and Machine Intelligence, IEEE Transactions on **25**(8) (2003) 959–973
14. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience (2000)
15. Li, G., An, C., Pang, J., Tan, M., Tu, X.: Color image adaptive clustering segmentation. In: Proc. Image and Graphics Conference. (2004) 104–107
16. Szumilas, L.: Feature co-occurrence texture detector. Technical report, PRIP-TR-099. Pattern Recognition and Image Processing, Vienna University of Technology (2005)