Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: lech@prip.tuwien.ac.at, hanbury@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-99 December 13, 2005

Segment Feature Co-ocurrence Based Texture Detection

Lech Szumilas, Allan Hanbury

Abstract

This work on texture detection was inspired by the general problem of object recognition in two dimensional still images. One of the crucial challenges associated with the object recognition is selecting and obtaining discriminative features. In analysis of real scenes, like nature or urban places, many objects contain textures, which can be considered as one of the object features. Texture detection may also significantly improve image segmentation, which is one of the tools used for object recognition. This report presents a novel method named Feature Co-ocurence Texture Detector (FCTD) which allows for fully automatic detection of textures common in real scenes, like water in lakes, tiger skin, fields of flowers or tree crowns. The method searches for an alternating color pattern, like for example black and orange stripes covering tiger skin, which is very often present in those textures. The final result is a hierarchy of textures (described by their boundaries and a set of features) detected at multiple precision levels, which can be used for further analysis or texture classification. It is achieved through hierarchical clustering of color pairs related to adjacent image segments, where each segment represents a low color gradient, simple shaped patch of pixels in the image. The results are presented on some images from the Berkeley database.

Keywords: texture, hierarchical clustering, symmetry points, watershed.

Contents

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 [9, 8], while texture classification can provide extremely useful information for object recognition methods.

The term "texture" typically describes a 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. In many industrial applications a texture is considered as a number of identical elements uniformly distributed in 2 dimensional space, which allows for example for detection of faults in analyzed materials [1]. This definition however is too strict to detect less regular patterns frequently appearing in natural scenes. The ripples on the water create an easily noticeable pattern from a human perspective though they differ in size, direction and are not uniformly distributed. The same is true for the skin of a tiger which consists of bright and dark stripes, but varied in width, length and direction. These textures usually contain spatially mixed elements with some common features like color, shape, size, etc. but varied to some degree. Very often we can distinguish two or more types of elements, where each type has common features so that the final pattern consists of alternating features in 2D space like dark and bright patches in Figure 1. Discrimination of such patterns is a common problem in image segmentation algorithms, which attempt to divide images into regions of uniform color or texture. Here the question arises of how regular a pattern must be in order to be classified as a texture. It is impossible to define any hard limits, as they may differ between textures. This in turn implies that any texture detection should be based on prior knowledge.

The method "Feature Co-ocurence Texture Detector" (FCTD) does not use a priori knowledge to detect textures, instead it detects textures at various "regularity levels" and produces a hierarchy of textures, which can be later used for texture classification. The advantage of such an approach is the ability to capture less and more regular patterns automatically and then to make a knowledge based selection.

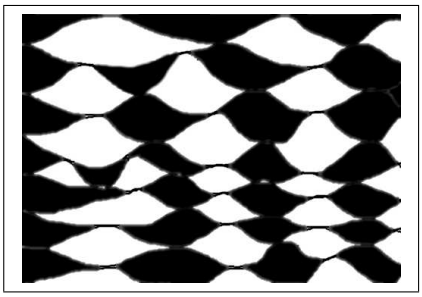

Figure 1: Example of a texture, where texture elements are not perfectly regular.

Let us assume that we start texture detection with no prior knowledge and that our task is to find all possible patterns in the image. The primary problem associated with this task is the lack of any information about texture elements. This seems to be a "chicken

and egg" problem as it is impossible to find a pattern not knowing features related to texture elements and it is impossible to find texture element boundaries without knowing the texture characteristics. Fortunately a very common pattern found in textures is the alternation of two or more colors or just luminance levels (like in Figure 1). Therefore FCTD initially uses only color features to do a coarse texture detection. The general idea is to segment the image into small segments with a relatively uniform color and then find alternating color patterns among those segments. Since a single segment is a patch of similar color pixels, 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. For example in Figure 1 we can distinguish two groups of bright and dark elements. Furthermore each dark element neighbors with at least one bright element. The word *element* is intentionally used instead of segment as in practice it can be difficult to obtain segments covering a whole texture element, nevertheless FCTD uses segments to find color patterns.

In Section 2 we describe the FCTD algorithm in detail. Section 3 presents results along with a discussion. Finally Section 4 concludes and discusses future work. For example, the coarse texture detection performed by FCTD provides information, which could be later used to obtain texture element boundaries and then use more features (like shape) to perform fine texture detection as well as texture classification.

2 Overview of the Method

The FCTD searches for simple color patterns and at this stage all other potential texture invariants are ignored. For example even if texture elements vary significantly in size, shape, direction or are not uniformly distributed a texture is detected assuming there is a clear alternating color pattern present. This may also lead to detection of image areas which would not be classified as texture by most humans. However any false positives can be eliminated by further analysis and therefore the best strategy is to allow some false positives rather than miss important textures.

The FCTD method can be divided into several steps outlined below:

- **Image Segmentation** – The primary goal of this step is to produce segments with uniform color and low shape complexity. The second requirement is necessary to avoid segments which may span over several texture elements.

- **Feature Extraction** – The only feature type used is color, however the feature itself consists of color pairs or triplets from neighboring segments. This allows the grouping together of segments with similar color and with neighboring segments having other specific color(s).

- **Feature Clustering** – All segment color-pairs are hierarchically clustered producing a cluster tree with 2 clusters at the top and many at the bottom.

- **Texture Detection** – Relatively large clusters from all clustering levels are separated from the rest and clustered separately until texture is detected or maximum clustering level is reached. This process also produces duplicated textures, which are removed at the end.

The result of the FCTD is a set of textures, for which boundaries and color patterns are known. We now describe each of theses steps in detail.

2.1 Image Segmentation

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

A typical segmentation algorithm attempts to create segments covering continuous patches of pixels of uniform color or uniform color and texture. This approach does not guarantee that the segment shape is convex. The algorithm closest to our requirements is the well known Watershed [11], however it may also generate complex shaped segments. FCTD uses a marker based version of the Watershed in which locations of markers (typically local gradient minima) are aligned with local symmetry maxima. A radial symmetry is calculated over the area surrounding each pixel using the following equation:

$$s(x_c, y_c) = -\sum_{\alpha=0}^{\pi}\sum_{r=1}^{R}\left|\boldsymbol{I}_{(x_c,y_c)}(r,\alpha) - \boldsymbol{I}_{(x_c,y_c)}(r,\alpha+\pi)\right| \tag{1}$$

where $\boldsymbol{I}_{(x_c,y_c)}(r, alpha)$ is an image pixel at polar coordinates (r, α) with the center at image Euclidean coordinates (x_c, y_c). Positions of local symmetry maxima are then used as Watershed seeds (see Figure 2).

Direct implementation of Equation 1 would require conversion of the coordinate system from Euclidean to Polar coordinates, which is computationally costly, especially if each pixel in polar coordinates is interpolated for better accuracy. A simpler method exists, which calculates a difference for every pixel pair at coordinates $(x_c + i, y_c + j)$ and $(x_c - i, y_c - j)$ inside a square region of the image, centered at (x_c, y_c). The sum of all pixel pair differences is the symmetry measure:

$$s(x_c, y_c) = -\sum_{i=-R}^{R}\sum_{j=0}^{R}\left|\boldsymbol{I}(x_c+i, y_c+j) - \boldsymbol{I}(x_c-i, y_c-j)\right| \tag{2}$$

The symmetry measure resulting from equations 1 and 2 are similar but not identical. In both cases an exact result cannot be achieved. The first method suffers from inaccuracies associated with coordinate system transformation, while the second method uses square pixels, which is an approximation only of pixel shape in polar coordinates. The inaccuracies however are relatively small in both cases, therefore the second and most efficient method is used.

The 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 (at angles α and $\alpha + \pi$) are identical. Therefore a local symmetry maximum corresponds to the image location with the most similar pixel pairs.
- 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 (see Figure 2).

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 [7], 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 the primary goal of the segmentation is to avoid creation of segments larger than texture elements. In that case relatively small irregularities in the segment boundary do not matter. All the results in this report are produced using a color gradient, which is an average of gradients $\left|\nabla \boldsymbol{I}_c\right|$ calculated over each color channel c:

$$\left|\nabla \boldsymbol{I}_c\right| = \left|\frac{\partial \boldsymbol{I}_c}{\partial x} + \frac{\partial \boldsymbol{I}_c}{\partial y} i\right| \tag{3}$$

We are able to decrease the overall number of segments by applying symmetry maxima as Watershed seeds (see Figure 3), however in the majority of cases images are still over-segmented. The notion of over-segmentation refers to the fact that 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, but it is impossible to achieve perfect segmentation without some prior knowledge. This forces us to limit segment clustering to only basic color features (as discussed in 2.2). Section 4 discusses possible improvements allowing the use of additional features.

2.2 Feature Extraction

In this step a feature vector is assigned to 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 the current one. Such features allow one to analyze alternating color patterns, however there are several possibilities as to how colors are calculated and how many colors the feature vector contains. The most basic option would be a feature vector consisting of two colors (six real values) – one belonging to the segment the feature vector is assigned to and another belonging to the most different

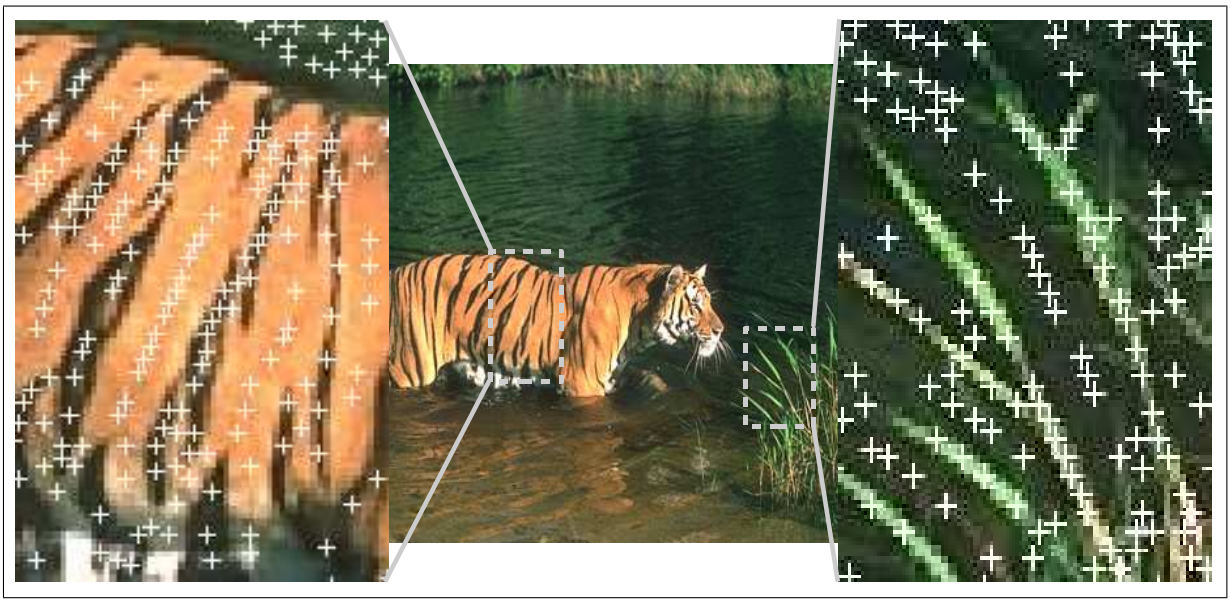

Figure 2: Example of symmetry maxima. Note that slight irregularities in the tiger stripe or grass stalk boundaries produce higher numbers of symmetry maxima along them and thus the number of Watershed segments is increased. Refer to Figure 14 to see the full image.

Figure 3: Example of Watershed segmentation based on symmetry maxima seeds (right) and without it, using only gradient minima (left). Note the number of segments on the right side is significantly lower than on the left side, but the segments on the right have still relatively low complex shapes in majority of cases.

color among segments surrounding the current one (see option 1 in Figure 4). The color difference is measured as the Euclidean distance in three dimensional color space (typically CIE-Lab [4]):.

$$d_{ij} = (\mathbf{C}_i - \mathbf{C}_j)(\mathbf{C}_i - \mathbf{C}_j)^T \tag{4}$$

where $\mathbf{C}_i$ and $\mathbf{C}_j$ are three element vectors containing color descriptions for segments i and j respectively.

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 color distance between the current segment and the one with the most different color among surrounding segments (see option 2 in Figure 4).

It is also possible to use more than one color for the description of the surrounding segments like in option 3 in Figure 4.

The particular choice of features depends on the expected pattern characteristics. Nothing prevents us from performing texture detection multiple times using different features and then selecting the best results as explained in Section 3, however all results presented there use "option 2".

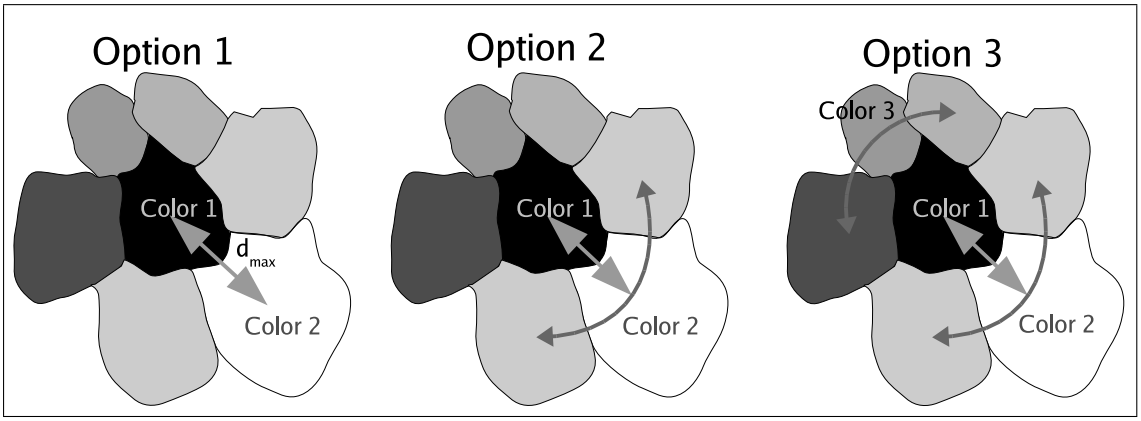

Figure 4: Extraction of color features from neighboring Watershed segments

The feature extraction step also produces a region adjacency graph stored as an array, used for obtaining color pairs and cluster co-occurrence measures as described in Section 2.4. A list of neighboring segments is generated for each segment and stored in the corresponding row of the array.

Depending on the color space used we may have to adjust different color channels, to make the value range equal for all channels. For example CIE-Lab color space uses different value ranges for channels L, a and b. If used directly all three channels would have uneven influence on the distance measure between feature vectors and thus on the clustering. All results presented in following sections were generated using the CIE-lab color space with channels normalized to value range between 0 and 1.

2.3 Feature Clustering

Our primary goal is to detect alternating color patterns, which can be achieved by grouping segments with similar feature vectors (similar color pairs). Here, the problem of a similarity measure arises, as we do not know how similar objects should be in order to be assigned to the same group. Unfortunately if the similarity measure is simply an Euclidean distance between two feature vectors, then we may have to use different thresholds for different textures on the same image, which are not known a priori. We propose a different approach based on multi-level hierarchical clustering. Color clustering has been applied in the past for color reduction and segmentation [2, 6] as well as other problems, but it was always performed in three dimensional space, where each feature vector contained a description of a 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 of the feature vector and the color with higher luminance value occupies another three elements. This way adjacent texture elements containing different colors are represented by similar feature vectors and will fall into the same cluster. For example tiger body is covered with the bright and dark stripes. Without color sorting the bright and dark stripes would be splitted into two different clusters.

FCTD uses aglomerative hierarchical clustering [3] using the average linkage to build a cluster hierarchy of color pairs (and corresponding Watershed segments). The first step of the clustering is building a binary tree, where each node of the tree corresponds to a single cluster. The bottom of the tree contains nodes corresponding to each segment feature vector. The closest nodes in terms of the Euclidean distance between two feature vectors are paired together and form the next level of tree nodes. Each tree node is assigned the average value of paired feature vectors. The operation of node pairing is repeated until only two nodes remain at the top of the tree. Two nodes are paired only if both are the closest nodes to each other. If a node happens to have a closest neighbor, which has another even closer neighbor, then the node is not paired at this tree level and progresses to the next tree level without change. The paired node is assigned with the average value of all feature vectors at the tree bottom, which are linked to it through tree branches and nodes at previous tree levels. The two nodes at the top level of the tree are referred as a tree level 2 and all levels below are indexed with the increasing numbers. Only a limited number of tree levels (typically 10) are considered for further processing, however this number is application specific.

The clustering process itself does not take into account any geometrical dependencies, except for the fact that color pairs correspond to the neighboring segments. This in turn may produce non-continuous clusters - several unconnected patches of the image may be selected as a single cluster. If this happens FCTD splits such a cluster into as many clusters as the number of unconnected patches the initial cluster contains. As a result each cluster may be divided into more than two clusters at next clustering level.

Figure 5 is an example of a color-pair cluster hierarchy. At clustering level 2 the whole

image is divided into two relatively big clusters and several smaller clusters (containing 1-4 segments), separated from the big ones as unconnected patches. Level 3 seems to be nearly identical to level 2, except that a few outlier color-pairs are separated from the bigger 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 (see Figure 5). This is a side-effect of the method used for color-pair extraction, based on finding the most different color in the neighborhood of each segment. The majority of color pairs extracted from the tiger body represent bright (orange) and dark segments corresponding to bright and dark stripes. However at the tiger boundary, bright (orange) segments belonging to the tiger body and dark segments belonging to the water create the most different color pairs. At clustering level 4 this difference is large enough to cause separation of most boundary segments from the tiger body. Although this problem is not addressed in this report, 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.

It is important to note that the segment boundaries and the final clustering result will differ if a different gradient is used or radius R used for the symmetry measure calculation is changed. Nevertheless those differences are typically small and in each case allows us to find similar clusters. It is impossible to define a single rule for choosing the optimal radius R or gradient type, but nothing stops us from repeating the clustering many times with different parameters and then performing a selection among all results. This is practical as image segmentation and clustering time on a modern computer usually takes less than one minute [1].

The example from Figure 5 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 as it was already explained feature spread inside clusters decrease with clustering level growth. For example at clustering level 3 the water is represented by only single segment, but at clustering level 4 it is divided into two clusters and at level 5 it is divided inti 5 relatively big clusters. Top left and right water clusters are less illuminated than the one in the center, while the cluster at the bottom contains reflections of the tiger and therefore its color is changed. All those differences become significant only at cluster levels 4 and higher. 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.

The main advantage of hierarchical clustering is that it enables texture detection at different levels, thus allowing us to choose between less and more precise results. 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

[1] a multi-scale analysis is another possibility

classification.

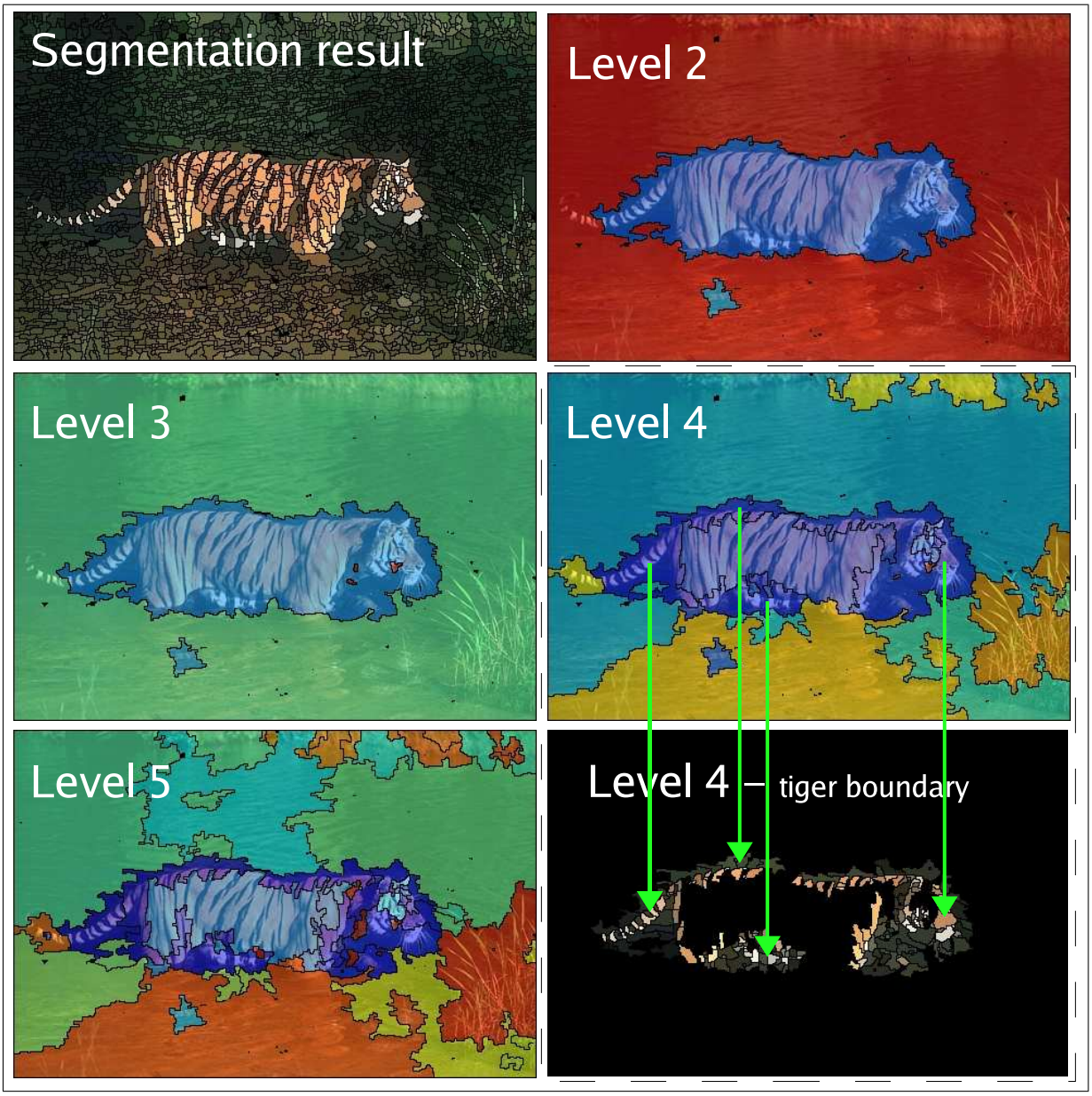

Figure 5: Example of a cluster hierarchy ($R = 5$, gradient calculated using luminance). It is best viewed in color.

2.4 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. Let us reconsider the color pattern from

Figure 1. If divided between two clusters (at level 2), then one cluster would contain dark color segments, which have bright color neighbors and another would contain bright color segments, which have dark color neighbors (without color sorting). In this case it is clear that almost all segments in both clusters have at least one neighboring segment belonging to another cluster and that is the color pattern we are looking for at this stage. 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 potential texture. Since segments vary in size and they are counted as neighbors even if they only share a one pixel long boundary it is actually better to measure the length of the boundary between two clusters, instead of counting neighboring segments. 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):

$$\frac{B_{kl}}{B_k} \geq \tau \quad \wedge \quad \frac{B_{lk}}{B_l} \geq \tau \tag{5}$$

Here, B_k and B_l describe the total boundary length of clusters k and l respectively (excluding boundary with background), B_{kl} specifies the boundary length between cluster k and l and τ is a cluster co-occurrence ratio in the range 0 to 1. If τ is set to 0 then every pair of clusters would fulfill this condition (even unconnected), however if τ is set to 1, then only clusters fully connected would be considered as potential textures. Typical values for τ vary in the range from 0.5 to 0.7 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.

Calculation of B_{kl} and B_{lk} is based on the segment adjacency graph, described in Section 2.2.

Each cluster from each clustering level obtained in the previous step is re-clustered using non-sorted color pairs. Re-clustering is performed until the adjacency condition in Equation 5 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. Instead of ending texture detection here clustering is continued in an attempt to divide the detected texture (two adjacent clusters) into 2 textures (4 clusters altogether). It means that up to three textures can be detected from a single cluster. This way we provide a choice of less and more generalized results for further processing.

The final result is a list of potential textures, each consisting of at least two clusters obtained from a single clustering level (see an example in Figures 6 and 7). Comparing results from Figures 6 and 7 we can clearly see that higher clustering levels are better suited to the detection of textures with less color spread among texture elements, while lower clustering levels do the opposite, which explains why the texture detection is performed across all clustering levels.

The list of detected textures describes regions of the image containing some color pattern. However, depending on the value of the threshold τ in Equation 5, clusters may have very few neighboring segments or even if τ is close to 1, clusters may be connected through a single boundary (for example two elongated clusters as in Figure 8), which is far from the alternating color pattern we are looking for. The other problem relates to the fact that many potential textures overlap each other, because the co-occurrence analysis is performed for each clustering level, producing many very similar potential textures. It is relatively easy to remove duplicates from the list, by leaving only the single biggest texture out of a set of highly overlapping textures. Removal of textures which do not contain alternating color patterns requires the usage of features other than color for clustering (like segment geometry) and will be addressed in future work.

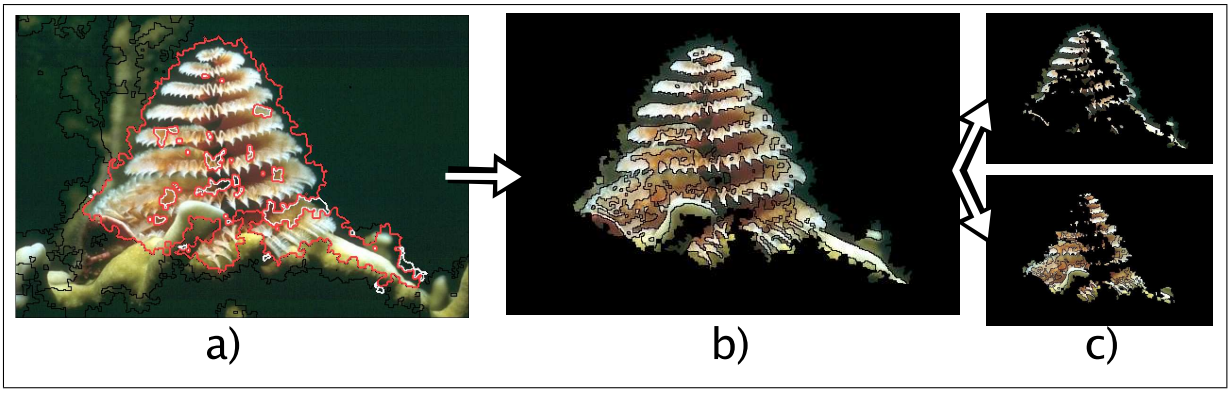

Figure 6: An example of textures detected in image (a) at image clustering level 3. Image (b) shows a texture detected from cluster marked with the red boundary line. Black lines show boundaries of the two adjacent clusters constituting the texture. Images (c) shows textures detected at the next texture clustering level.

3 Discussion of the Results

The current version of FCTD method is designed to automatically detect color patterns only. It uses segment color pairs for clustering, which does not allow for detection of purely geometrical regularities (when only the shape of texture elements is regular, but color varies significantly among those). We assume however that in the majority of cases, especially in natural scenes, textures contain both color and geometrical patterns, and color pattern analysis is a good first approximation to the texture detection problem. The method will be later improved with the addition of geometrical pattern analysis (Discussed in Section 4).

The final result of the texture detection depends on a few parameters listed below:

- R – the maximum radius over which the symmetry measure is calculated for each pixel neighborhood. Small values tend to generate more and denser distributed symmetry

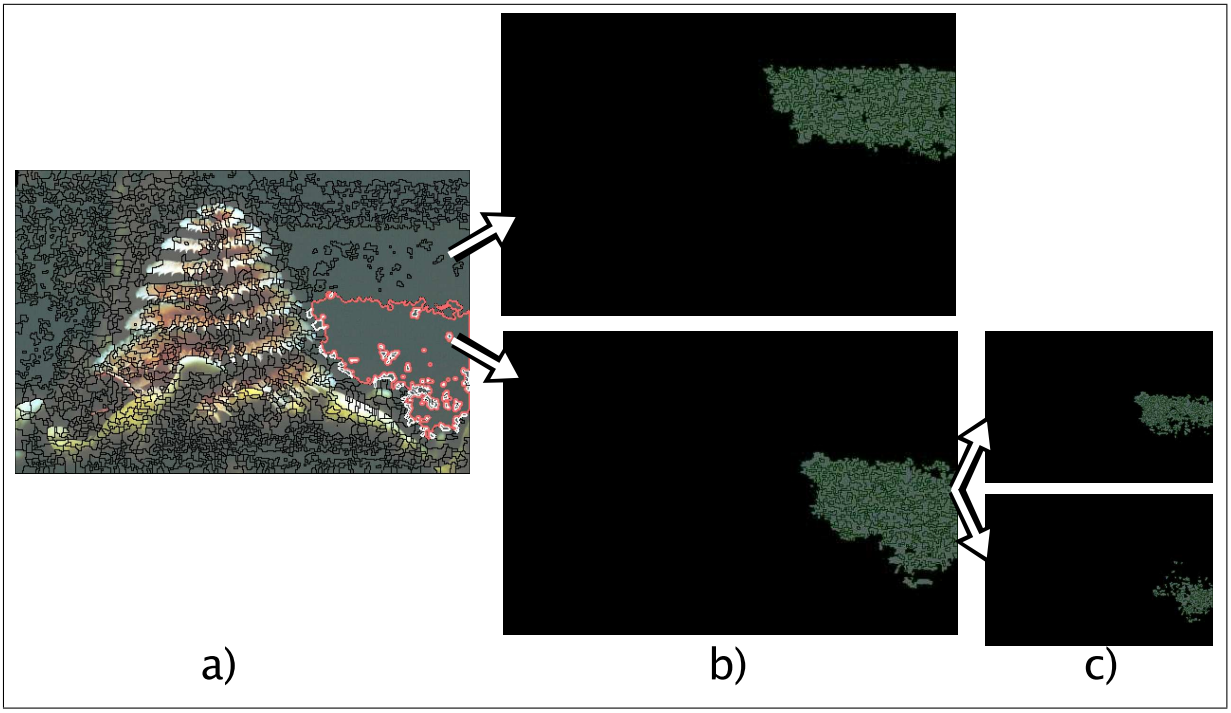

Figure 7: An example of textures detected in image (a) at image clustering level 10. Images (b) and (c) show textures detected at various texture clustering levels. Images were manually enhanced for better visibility.

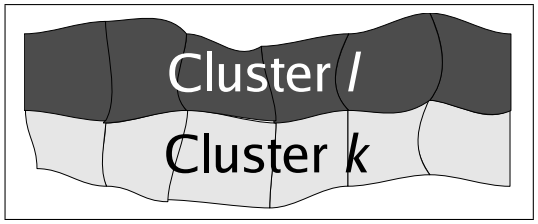

Figure 8: Example of two clusters with 100% segment neighborhood ratio which does not represent an alternating color pattern.

maxima, while bigger values do the opposite. Since each segment is associated with a single symmetry maximum, the number and the size of segments is strongly correlated to the value of R (see Figure 9), which also has implications on final clustering and texture detection. In practice a texture detection should be repeated for multiple values of R.

- τ – the cluster co-occurrence ratio is typically set to 0.8 to avoid analysis of clusters with very few neighboring segments, but also to allow analysis of cluster pairs which do not have all segments neighboring each other and/or are different in size.

- maximum clustering level – generally at higher clustering levels clusters become smaller and the feature spread between segments belonging to the same cluster narrows down. Different textures may be better detected at different levels, therefore multilevel analysis is essential. The maximum clustering level is typically set to 12, which allows the detection of all presented textures in this section. The higher the limit for the clustering level, the more processing time is needed to complete the texture detection. Figures 6 and 7 show an example of texture detection at different clustering levels.

While usage of color-pair based features allows us to capture alternating color patterns it also sometimes leads to inaccuracies in the detected texture boundaries. It may happen that some segments at the texture boundary have neighbors with colors which are more different than the colors of neighbors belonging to the texture. In this case the color pair for these boundary segments will be different than inside the texture and in effect those segments will belong to a different cluster. This is well visible in Figure 12, where the boundary of starfish (detected as a single texture) includes also some of the green background. Another similar example is shown in Figure 14. This problem however can be easily solved as discussed in Section 2.3.

All results presented below were obtained using a two color feature vector (see Section 2.2). This means that color patterns containing more than two significantly different alternating colors would not be detected or some colors would be missing in the detected texture. Such patterns can be detected using more colors in the feature vector.

Since the textures are detected from multiple clustering levels and in addition the texture detection phase can extract up to three textures out of a single cluster (see 2.4), the FCTD result is a texture hierarchy typically consisting of multiple bigger textures and a number of smaller textures containing a subset of segments belonging to the bigger ones, as in Figure 10.

Most of the results were obtained using $R = 5$, normalized CIE-Lab color space and averaged color gradient. These parameters influence the size, number and boundaries of watershed segments, thus producing different features for clustering. These parameters primarily affect the boundary of the detected textures. The highest influence has parameter R, which for the image size used for experiments was varied from 3 to 16 pixels. For example the results of clustering at level 2 in Figure 5 divide the whole image into two large

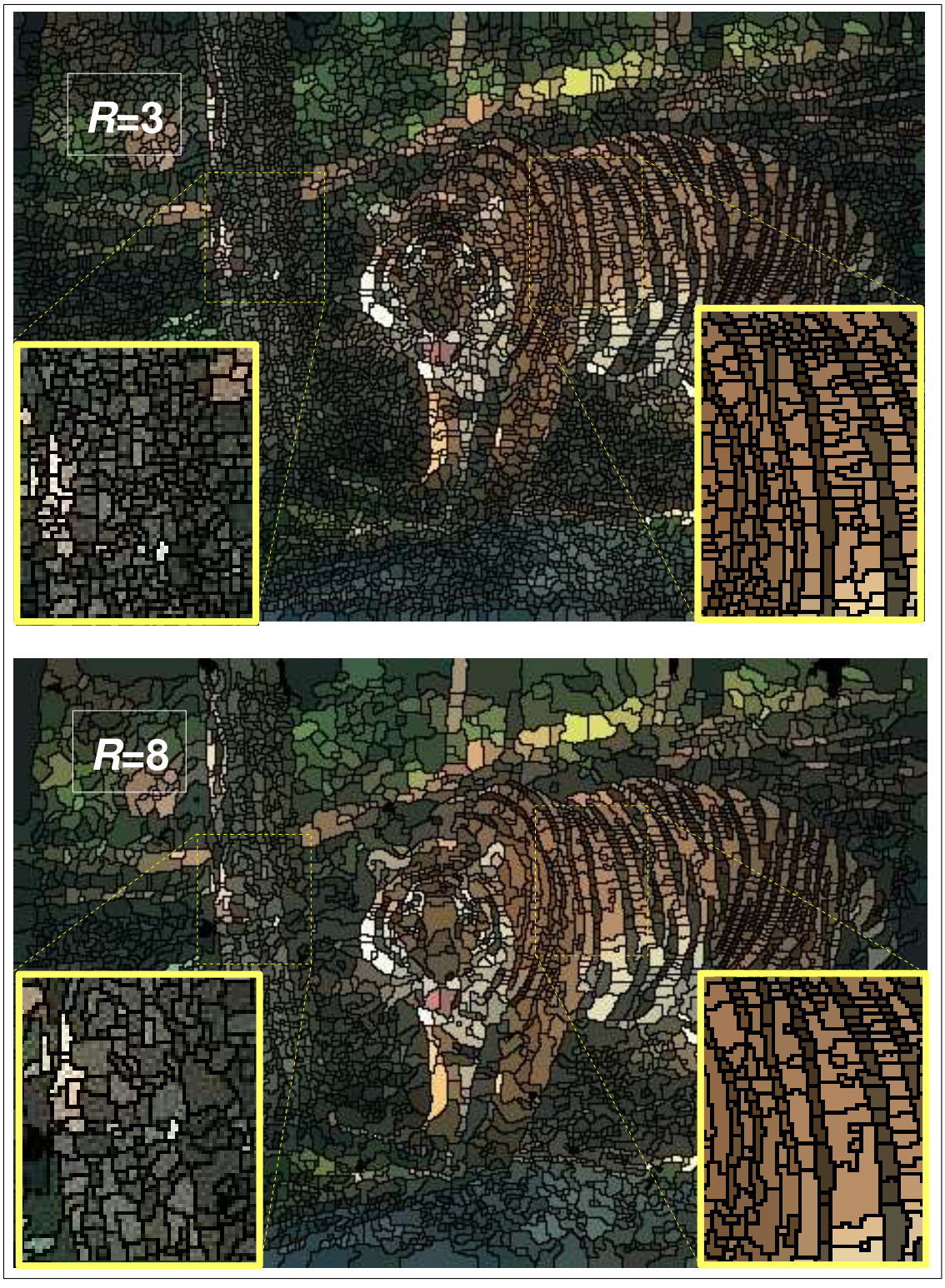

Figure 9: The influence of R parameter on segmentation result (R equal to 3 and 8 respectively).

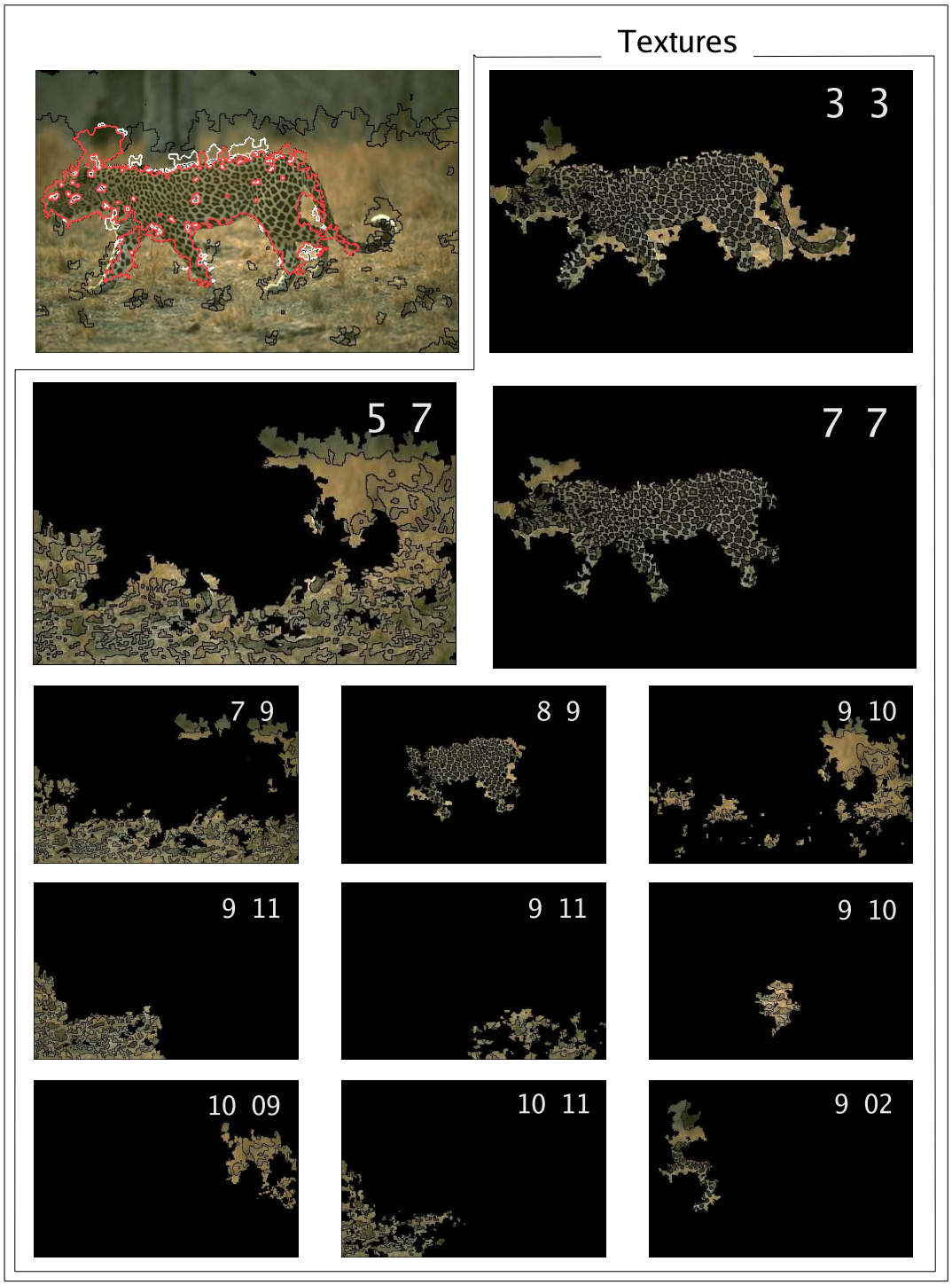

Figure 10: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalized CIELAB color space, averaged color gradient, 10 clustering levels). Textures are detected at various clustering levels shown on the texture images – the first number indicates the image clustering level, while the second number indicates the texture clustering level. A total of 12 textures were detected.

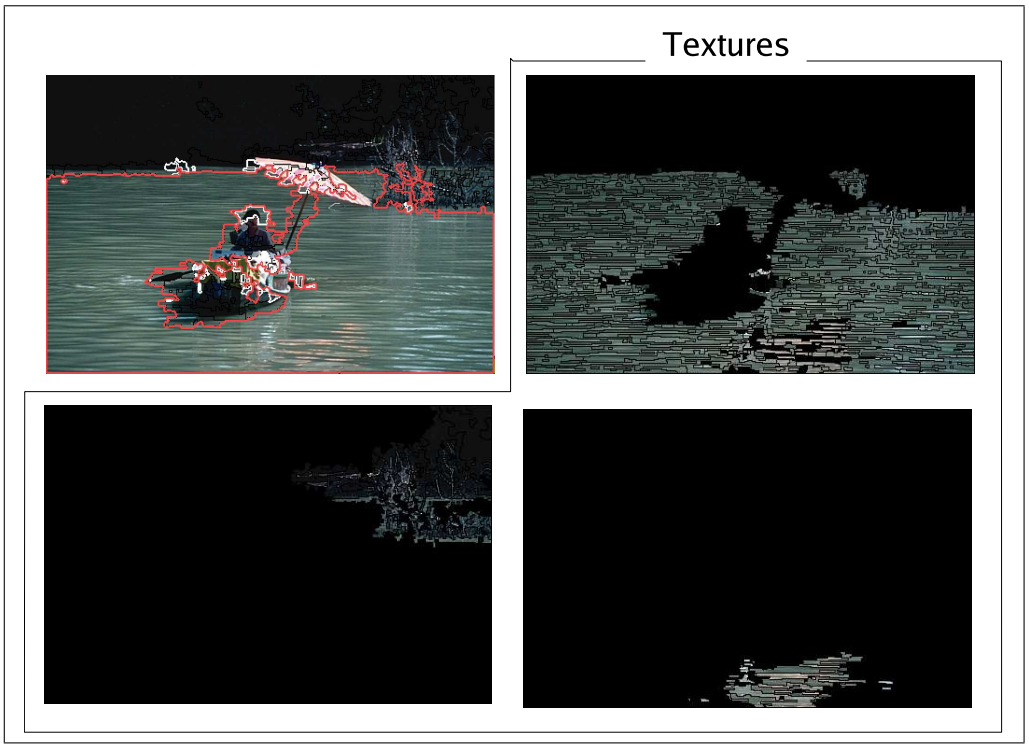

Figure 11: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalized CIE-Lab color space, averaged color gradient, 10 clustering levels). Total of 14 textures discovered with three of the most distinctive shown, the rest of the textures are subset of these three and are not shown to conserve space.

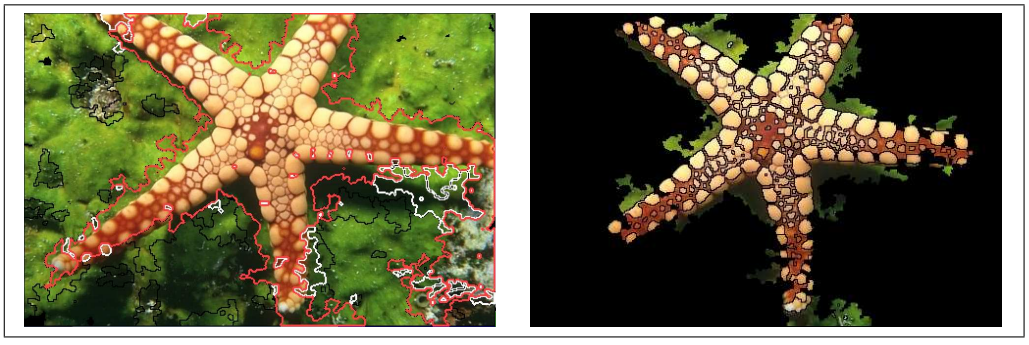

Figure 12: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalized CIE-Lab color space, averaged color gradient, 10 clustering levels). Total of 18 textures were detected, but only the most representative one is shown. Other textures contain parts of the starfish body and background.

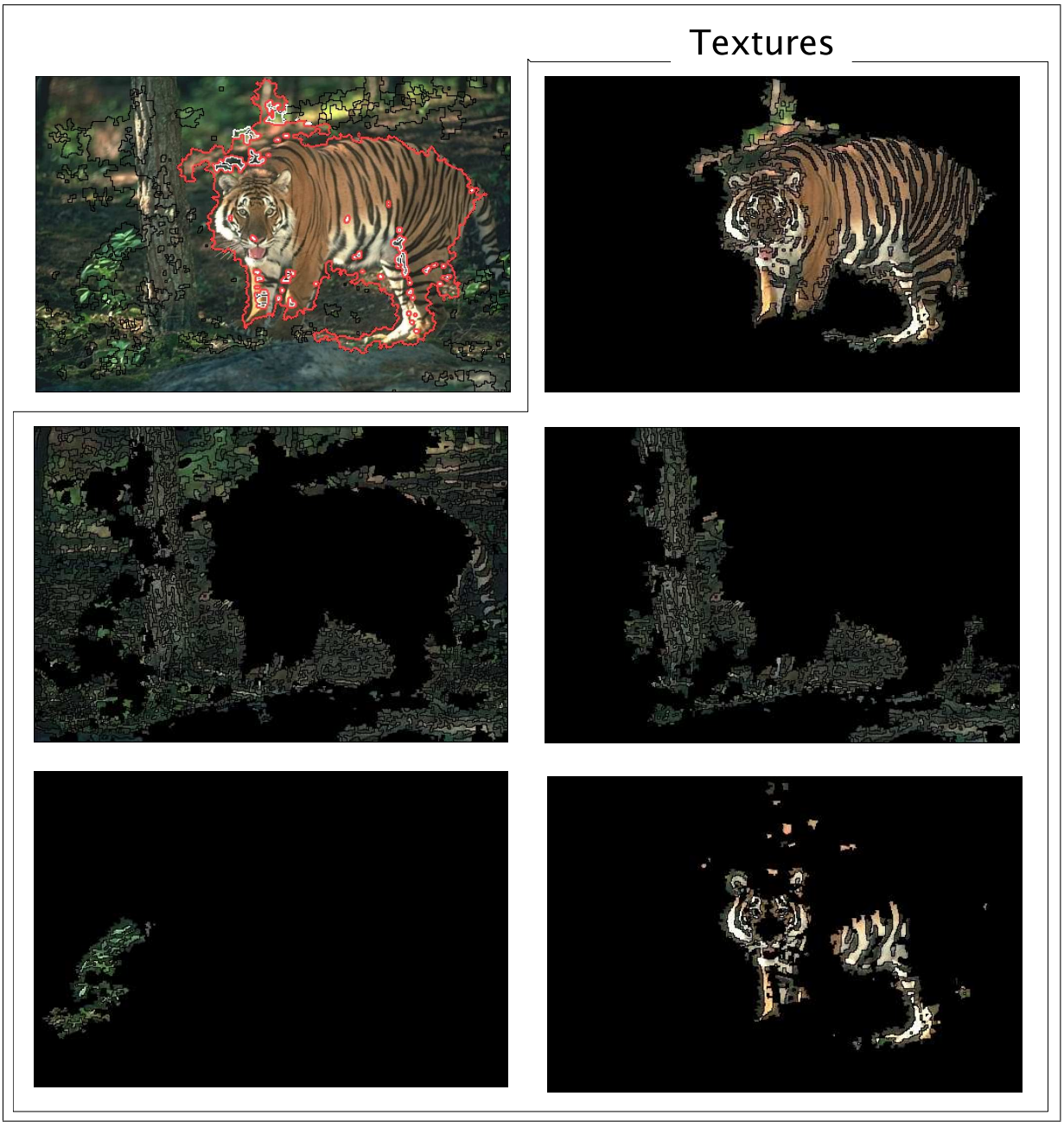

Figure 13: An example of texture detection from an image of size 481x321 pixels ($R = 3$, not normalized CIE-Lab color space, averaged color gradient, 10 clustering levels). Total of 29 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

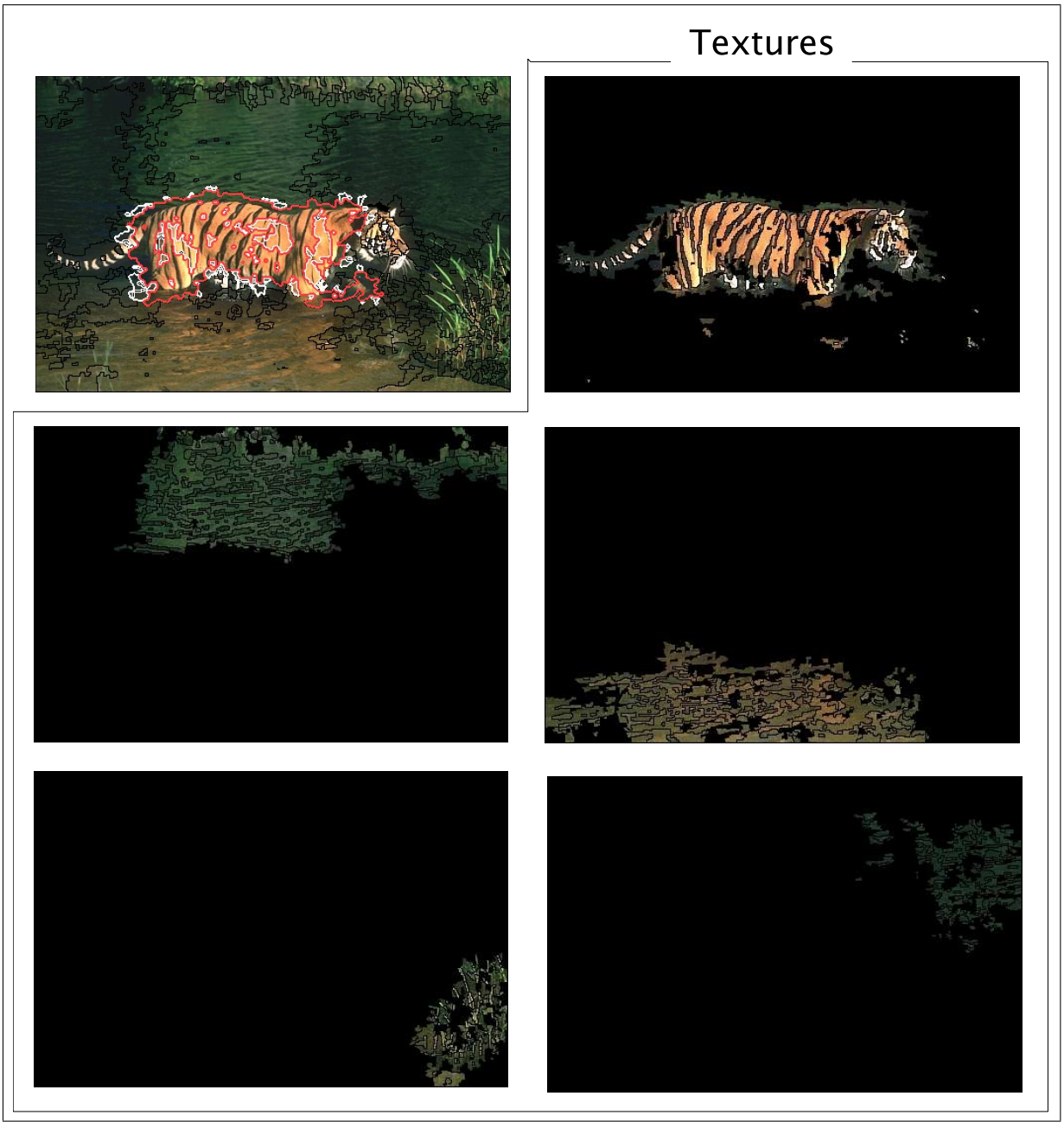

Figure 14: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalized CIE-Lab color space, luminance gradient, 10 clustering levels). Total of 27 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

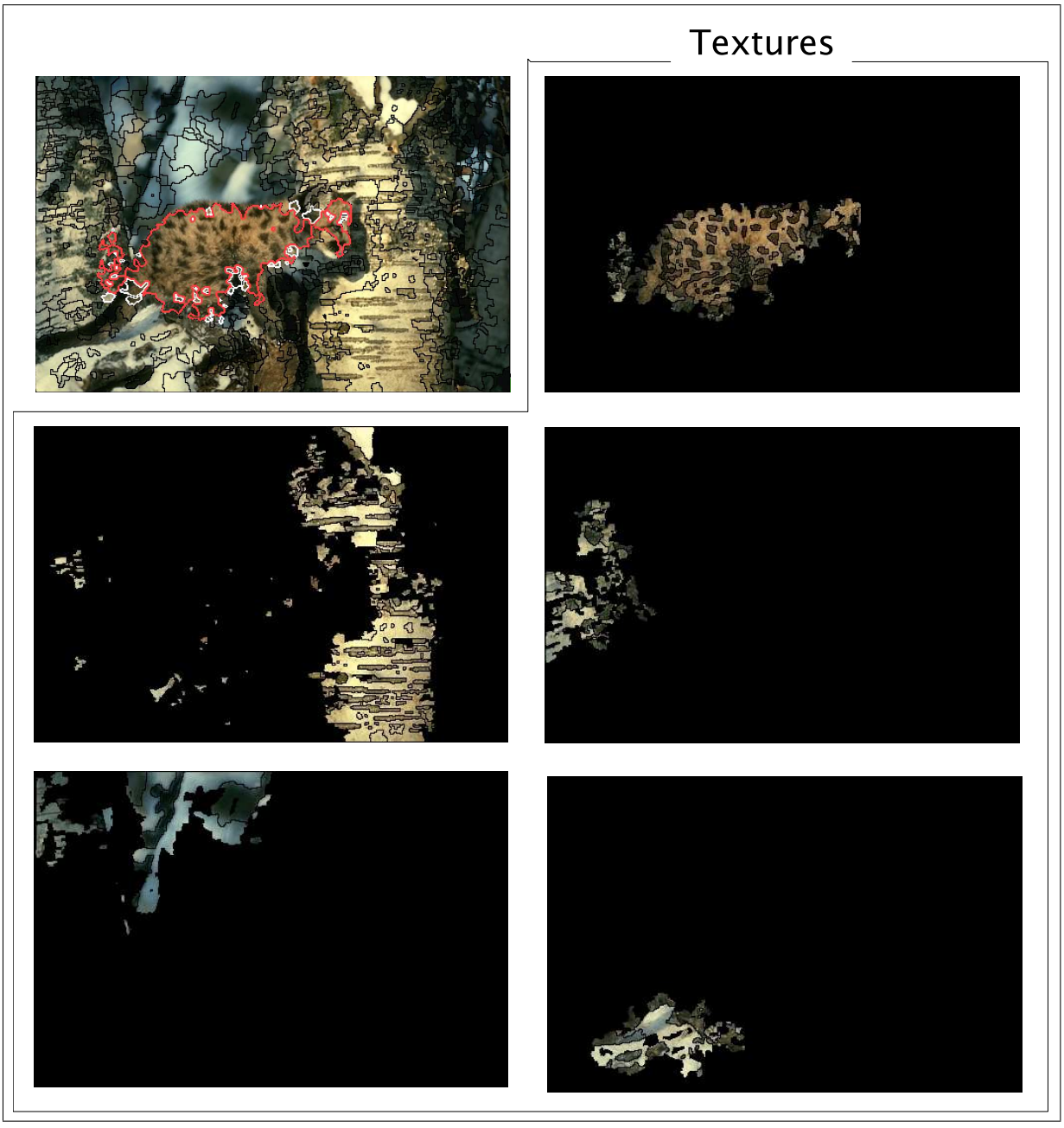

Figure 15: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalized CIE-Lab color space, averaged color gradient, 10 clustering levels). Total of 24 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

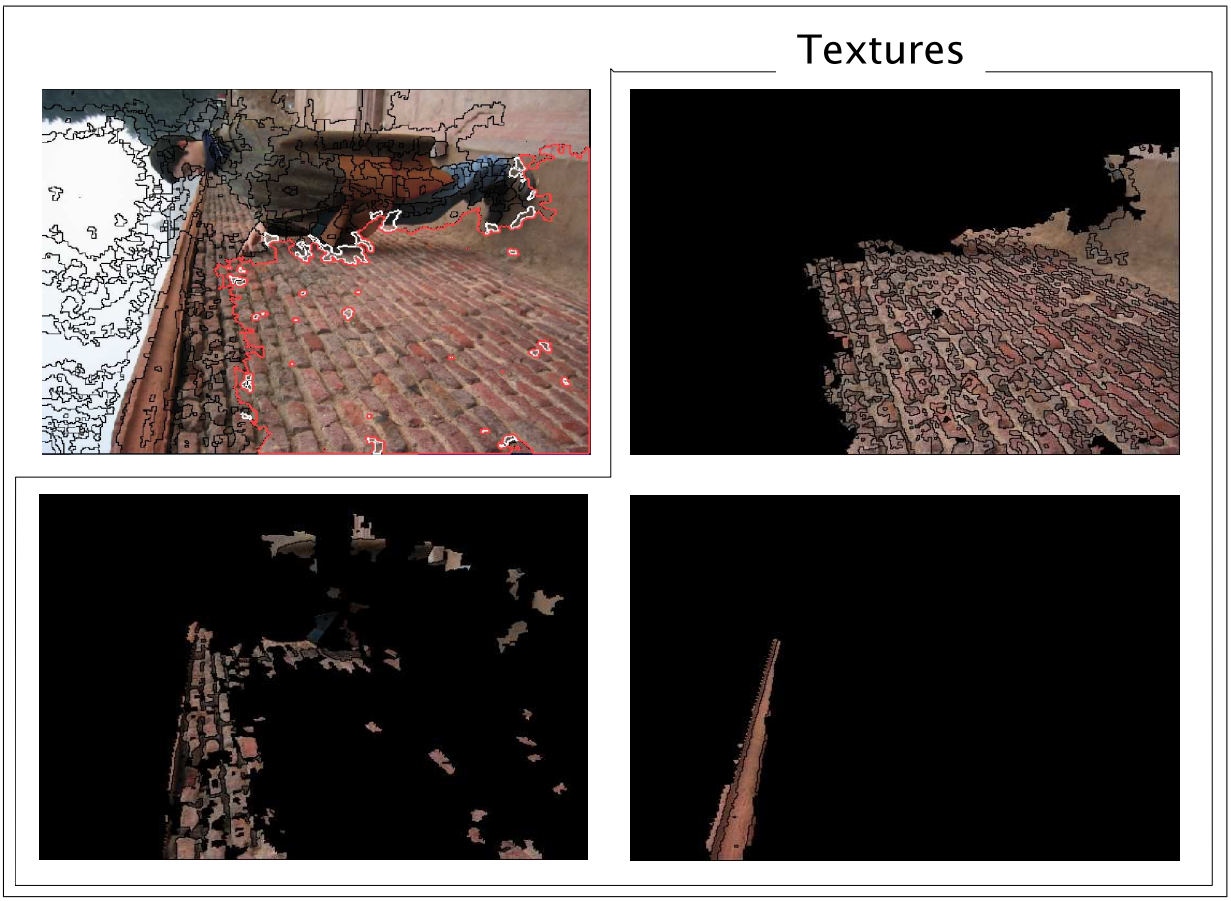

Figure 16: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalized CIE-Lab color space, averaged color gradient, 10 clustering levels). Total of 8 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

regions, one associated with the tiger and another with mostly the water. If the clustering is repeated with R set to 8 instead of 5, then the tiger and part of the water reflecting the tiger become a single cluster. The tiger body will be separated from the water at higher levels and make possible detection of the characteristic color pattern associated with the tiger. This example shows that though the boundaries of clusters will differ, the same color patterns will be detected when using various R parameters, but it may happen at different clustering levels. Nothing stops us from repeating the detection process using different parameters. This may generate a higher number of similar textures detected, but these results may be further selected, for example using texture classification. The whole texture detection time did not exceed 60 seconds on an average modern PC computer. If repeated for a combination of a different parameter values it could consume approximately 10 minutes (for example for R=3,5,8 and 10, two gradient types and with color normalization switched off and on). Figure 17 represents a difficult case from the color pattern detection point of view, because the color pattern of the lizard and background are quite similar. It was possible to detect a part of the lizard using typical parameter values, but the texture detection in this case is more sensitive to parameter values.

4 Conclusion

FCTD is a method for fully automatic detection of textures 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, which allows detecting color patterns in the images. The method proved capable of detecting a variety of color patterns, at different accuracy levels, from natural scenes as discussed in Section 3. 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. FCTD has several important features, which can be used for several image analysis tasks including image segmentation, texture classification and object recognition:

- FCTD generates a texture hierarchy, which captures most of the textures in the image (with higher and lower feature spread). This allows FCTD to be used in a variety of applications, which may require more or less accurate patterns to be detected or classified.
- FCTD is fully automatic and therefore can be used as a black box by other methods.
- FCTD provides not only texture boundaries, but also important texture features, like dominant colors and color spread. This information can be easily extracted, because detected texture always consist of two clusters (in the case of two color patterns) of specific color and specific color neighbor (see Section 2.4).
- The clustering step of FCTD can be considered as an image segmentation. Together with the texture detection it can provide segment boundaries as well as information on which regions contain textures.

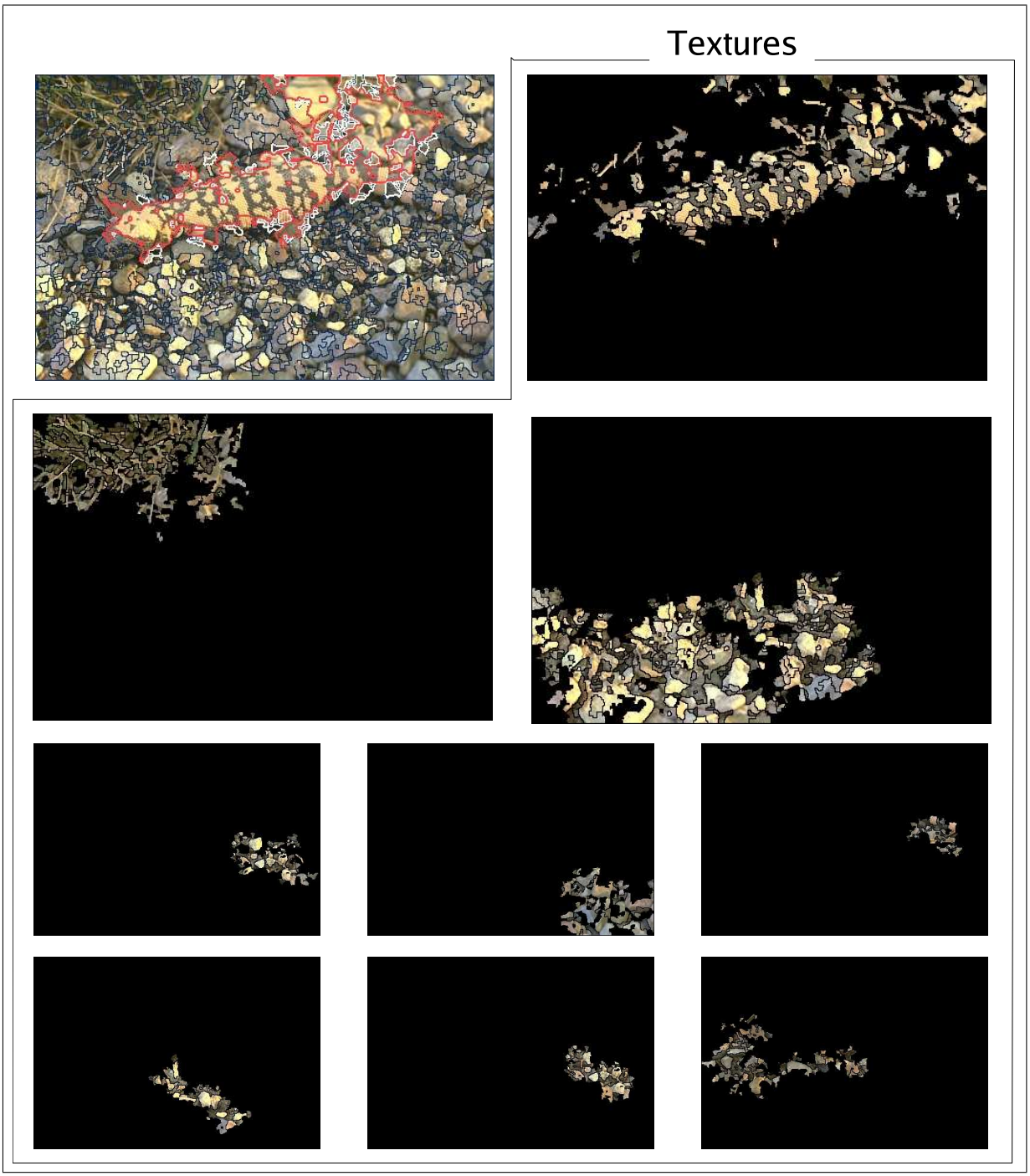

Figure 17: An example of texture detection from an image of size 481x321 pixels ($R = 5$, normalised CIE-Lab color space, averaged color gradient, 10 clustering levels). Total of 27 textures were detected, but only the most representative subset is shown. Other textures contain parts of the shown ones.

The FCTD is different from other texture detection methods in many ways. There are few other algorithms which attempt general texture detection [5] or try to detect textures and find their boundaries [10], but neither uses the same texture detection approach nor produces a texture hierarchy. It is difficult to do a performance comparison, as we do not have results of other methods on identical images and other methods do not provide a hierarchy of textures. Segmentation algorithms are another group of methods utilizing texture detection, which attempt to find boundaries of uniform color and/or uniform texture regions. Segmentation algorithms also use a non-hierarchical approach for texture detection. This means that they typically contain fixed parameters governing texture detection and discrimination [8] or require user to specify the most characteristic textures on the image [9]. FCTD generates a hierarchy of textures, because different textures may be better/only captured at different accuracy levels (which directly correspond to clustering levels). In addition FCTD extracts at the same time high level texture features, which can be used for texture classification. Nevertheless FCTD in many cases provides similar texture boundaries as segments in methods described in [9, 12] on the same image (compare tiger and leopard textures in Figures 14 and 10 with segmentation of identical images in [9]).

The current version of FCTD is the first step towards a general texture detector and classifier, capable of analyzing color and geometrical patterns. It is therefore necessary to address several issues discussed in Sections 2.4 and 3. Several improvements are planned:

- Over-segmentation - The current version of the image segmentation produces rather small and dense segments, which often divide texture elements (like tiger stripes in Figure 9) into several parts. This in turn has two negative effects:
  - we do not have texture element boundaries so geometrical analysis of those is impossible.
  - because of the image over-segmentation the interior and boundary of texture elements may be inserted into different clusters. Segments inside the element have neighbors with similar colors, while boundary segments neighbor with significantly different colors of another texture element (observe tiny segments in the middle of bright tiger stripes in Figure 9).

  Both issues will be resolved using adaptive segment merging, based on current color pattern detection. Currently detected textures are the source of information about color spread inside the cluster, which can be used to select which adjacent segments should be merged. Special care must be taken to avoid the creation of complex shaped segments. For example stripes on the tiger body often join. If the method creates a single segment covering multiple stripes, then geometrical analysis of those would be very difficult.

- Boundary segments - As discussed in Section 3 sometimes segments belonging to the texture boundary are separated from the texture itself. This will be solved using

statistics of the segment features belonging to the texture and those adjacent to the texture with similar features.

- Features - The color pattern is usually not the only one occurring in textures, therefore other features, especially geometry related, must be applied in order to obtain more precise texture boundaries and improve texture classification. The current plan assumes that other features will be obtained after the detection of color patterns, which can be compared to a two pass texture detection. Detection of color patterns allows for segment merging, which in turn will enable the extraction of basic geometrical properties of the segments - for example segment elongation, size, direction, etc.

- Scale - At present a textures with very small elements, i.e less than 3 pixels in diameter, are not detected due to the Watershed implementation limitations. This problem can be solved either by inflating the image 3 times across vertical and horizontal axes or using an improved Watershed implementation.

The present version of FCTD can be treated as a first approximation to the texture detection and despite several problems discussed in Section 3, provides useful texture detection.

Future plans include using the FCTD results for texture classification.

References

[1] D. Chetverikov. Pattern regularity as a visual key. *Image and Vision Computing*, 18:975–985, 2000.

[2] G. Dong and M. Xie. Color clustering and learning for image segmentation based on neural networks. *IEEE Transactions on Neural Networks*, 16:925–936, 2005.

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

[4] G. Hoffmann. *CIELab Color Space. http://www.fho-emden.de/~hoffmann/cielab03022003.pdf.*

[5] K. Karu, A. K. Jain, and R. M. Bolle. Is there any texture in the image? *Pattern Recognition*, 29:1437–1586, 1996.

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

[7] 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.

[8] 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.

[9] B. Mičušík and A. Hanbury. Steerable semi-automatic segmentation of textured images. In *Proc. Scandinavian Conference on Image Analysis (SCIA)*, pages 35–44, 2005.

[10] R. E. Sanchez-Yanez, A. Moral-Perea, and V. Ayala-Ramirez. Fuzzy texture detection using homogeneity cues. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 7, pages 6429–6433, 2004.

[11] P. Soille. *Morphological Image Analysis*. Springer, 2002.

[12] B. Sumengen, B. S. Manjunath, and C. Kenney. Image segmentation using curve evolution. In *16th International Conference on Pattern Recognition*, volume 2, 2002.