

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

**Spatial Constrained K-means Clustering Approach
to Image Segmentation**

Miłosz Chmura

Voditelj: *Marko Subašić*

Zagreb, May 2007

Content

1. Introduction to image segmentation1

2. K-means algorithm2

3. Spatial constrained image segmentation algorithm4

 3.1. Image downscaling4

 3.2. Features extraction4

 3.3. Spatial constrained K-means6

 3.3. Post-processing operations.8

4. Results10

5. Conclusion.....11

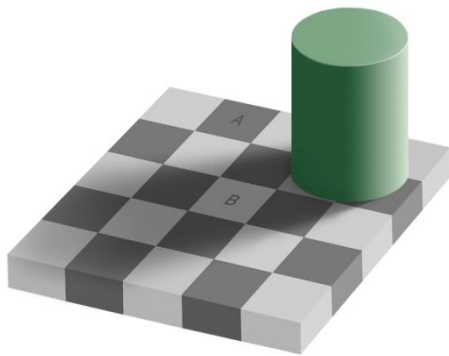
6. Bibliography.....12

7. Abstract13

1. Introduction to image segmentation

Image segmentation is a process of segmenting an image into distinct regions, which share the same defined features. This process is done in order to simplify description of the picture and to classify some of its features. Information retrieved from a segmented image can be used for object-based compression (used in MPEG-4 standard), in CBIR (content-based image retrieval), or in computer vision technology.

It is not an easy process to segment an image because, what is needed in defining specific algorithms, is understanding of how human being receives and transforms an image through the eyes and later the brain, i.e. how he/she recognizes and classifies objects.



Picture 1.1: Optical illusion

Difficulties come because some of the processes human being uses in classifying an image are: (i) using contrast in color recognition, (ii) fulfilling unfinished contours of objects, (iii) using past experience in items recognition. On the other hand a computer stores a digital picture only as a map of pixels, each having its own color information¹. This is why it is not an easy task to write algorithms, which are able to represent a picture as human being sees, stores, and recalls it. Image processing algorithms can then have a very high complexity level: polynomial $O(n^k)$ or even exponential $O(2^n)$.

For example of (i): on a dark background gray color will seem to be white, while on a light background the same color will be perceived with higher intensity. On *Picture 1.1* the color of square A is the same as the color of square B.

In my work I will present an approach of image segmentation using K-means algorithm with spatial constraints. In the following sections I will: (i) explain how K-means algorithm works, (ii) describe the algorithm, which segments pictures using spatial constrained K-means approach (SCK-means), (iii) show the segmentation results of some sample pictures, and (iv) conclude my work.

¹ There are several formats used for storing pixel information: RGB, CMYK, La*b*, HSV, etc. Most common is RGB in which separate intensity of three colors: red, green, and blue are considered.

2. K-means algorithm

There are several methods, which can be used to segment an image such as: region-growing, histogram-based, graph partitioning, split & merge, or cluster-based. In this section I am going to describe how K-means algorithm works, which is a clustering method.

K-means is an algorithm given by MacQueen in 1967, which clusters sample data into K groups, based on data's attributes. The sample data or feature space is a set of vectors any dimension. Vectors contain attributes, which are considered while grouping data into K clusters. Each of these vectors can belong only to one of K groups. K-means algorithm tries to minimize the distortion or variances among vectors and the group they belong to. In other words, K-means tries to find K centers of data, where the distances for each vector to its center are minimized. Formula (1) shows the variances among K centers and sample data vectors.

$$V(K) = \sum_{i=1}^K \sum_{x_j \in S_i} |x_j - \mu_i| \quad (1)$$

K is number of clusters, x_j is a data vector, S_i is a set of all vectors belonging to i-th cluster, and μ_i is the i-th center or centroid of S_i .

The algorithm's steps are as follows:

- (1) Place K vectors in the feature space. These K-vectors are initial centroids
- (2) Assign each data vector of the feature space to the closest centroid
- (3) Recalculate new centroids for each group of vectors
- (4) Repeat (2) and (3) until centroids no more move

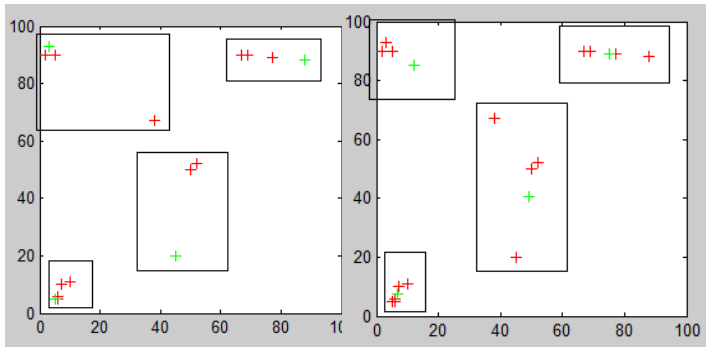
In the first step placing K-vectors (centroids) in the feature space can be done: randomly, by using some heuristic data, or by following the steps: (i) place the first K-vector anywhere in the feature space, (ii) place each next K-vector as far as possible from K-vectors, which are already set.

In the second step distances are calculated using Euclidian norm² between each data vector and each K-vector. Then each data vector is assigned one of centroids based on the minimum distance among them.

² There can be used several norms while calculating distance between two vectors like Manhattan, Euclidian, infinity, or any other p-metric.

After the assignment of all data vectors to their centroids is completed, the K-vectors are recalculated. The new centroids are the mean values of the data vectors they represent.

Steps (2) and (3) are repeated until no more recalculation of centroids is done (K centroids are at its best positions) or number of iterations exceeds the specified limit. As the result of K-means algorithm we get K groups represented by K-vectors, which in the most approximate way classify data vectors they represent.



Picture 2.1: K-means clustering of 2-D feature spaced sample data

vertical positions of points. The red pluses represent data vectors (sample data), the green pluses represent K-vectors (centroids), and the squares show the belongings of points to their centroids. On the left of *Picture 2.1* the initial conditions are presented, whereas on the right the final ones. The algorithm has finished clustering already at the first iteration.

K-means algorithm is commonly used because it converges quickly, i.e. centroids quickly find their right positions. It can be applied in several areas where estimation of data centers is needed, such as: physics, biology, chemistry, and image processing. As an example on *Picture 2.2* you can see the approximation of a RGB image using only 4 colors. In this example $K = 4$ and RGB is a 3-D data vector for each pixel; set of these RGB vectors create the feature space.

Although K-means is a quick algorithm it also has some withdraws. One of them is the problem of deciding where to put the initial K-vectors in the feature space because different positions of initial centroids produce different results. The other problem is determining K; in image segmentation if we chose too small or too big K the partitioned image may be over or under segmented. However one of solutions to this problem is to run the algorithm several times with different parameters.



Picture 2.2: 4-colored approximation

3. Spatial constrained image segmentation algorithm

The image segmentation algorithm, which I am about to present, is based on *A spatial constrained K-means image segmentation approach* [1] paper. The approach described in [1] shows how important it is to include spatial constraints in image segmentation and do not aim at regions, or pixels, while segmenting an image but rather focus on intermediate image segmentation.

The algorithm consists of four modules:

- (i) *downscaling,*
- (ii) *feature space extraction,*
- (iii) *spatial constrained K-means routine,*
- (iv) *post-processing morphological operations.*

Briefly, the algorithm reads a RGB image, downscales it to a reasonable size, divides the whole image over $N \times N$ blocks, and for each block assigns a 4-D data vector, which forms the feature space. Later on it alternatively performs region-growing connected-components and K-means algorithms on the spatial constrained feature space, stopping, when the specific criteria are met. In the end it erodes too thin parts, fulfills too small areas, smoothens the borders of the labeled picture, and merges over-segmented regions.

3.1. Image downscaling

In MATLAB each RGB image read from a file is stored as a $width \times height \times 3$ sized matrix. In this module a picture or the matrix is simply downscaled in order to speed up operations of clustering in (iii). In my experiment I downscale each image (preserving the ratio, width:height) in the way that the maximum number of pixels in width and height of the image does not exceed 320 pixels. This number is high enough to preserve the relevant information, which is used in the process of segmentation, because information about regions of the image is not lost during the process of downscaling. In my program I use function `im_dscaled=downscale(im_in)`.

3.2. Features extraction

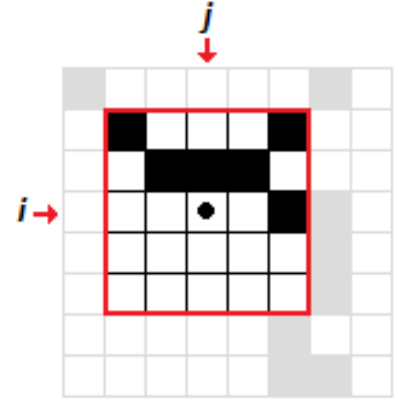
After the image is downscaled, we proceed to the next step – the feature space extraction. First, I block an image into $N \times N$ blocks (in my experiment $N = 2$). Each block contains mean approximation of RGB of the pixels it consists of and it is used for extracting features of the feature space. In other words in the blocked image each block is represented by a new pixel. In my experiment the feature space has

four features (dimensions); one feature is the texture intensity and the other three features contain color information. Each dimension of the feature space is normalized, i.e. the values range from zero to one. Normalization is done in order to treat each feature equally in (iii).

In order to calculate the texture intensity, I detect edges of the blocked image using SUSAN algorithm. I use SUSAN because it gives nice results and it is very quick. Each pixel of the resultant or edged image contains one if an edge is detected and zero elsewhere. The texture intensity feature is then extracted using the following formula:

$$I_{i,j} = \frac{N_e}{N_{int} \times N_{int}} \quad (2)$$

$I_{i,j}$ represents intensity of pixel in i -th row and j -th column of the image, and N_e is the number of edge pixels in $N_{int} \times N_{int}$ square area, which surrounds the (i,j) -th pixel. For example on *Picture 3.2.1*, which represents a part of an edged image, texture intensity for (i,j) -th pixel is $I_{i,j} = \frac{6}{25}$ because this pixel is surrounded by the red square of side $N_{int} = 5$ and number of edge pixels inside this square is $N_e = 6$. Notice that such defined texture intensity is already normalized. In my experiment I use $N_{int} = 9$.



Picture 3.2.1: Calculating texture intensity

In order to determine the other three dimensions of the feature space, color information standard RGB of the blocked image is transformed into HSV (*Hue Saturation Value*) standard for each pixel. This operation is done because HSV is more natural to human being's perception. Later, HSV is normalized using the following three transformations:

$$X = s \cdot v \cdot \cos h \cdot 0.5 + 0.5 \quad (3)$$

$$Y = s \cdot v \cdot \sin h \cdot 0.5 + 0.5 \quad (4)$$

$$Z = v \quad (5)$$

X , Y , Z are three dimensions in the color feature space, and $h \in [0, 2\pi]$, $s \in [0, 1]$, $v \in [0, 1]$ are the corresponding attributes of HSV. Notice that v is the intensity or gray level of the blocked image. The feature space is then a set of 4-D vectors, $[X Y Z I]^T$, each describing one pixel (block) of the blocked image. In my program I use function `im_feats=features4(im_downscaled)` for feature extraction.

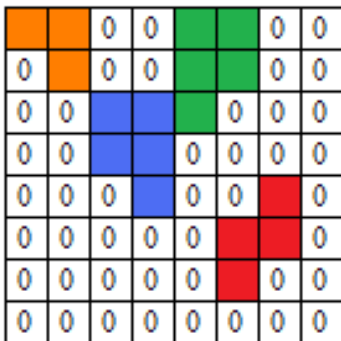
3.3. Spatial constrained K-means

After defining 4-dimensional color-texture feature space of the blocked image we can perform the top-down segmentation algorithm. The algorithm, which I am going to describe, is hierarchical, which means that it performs segmentation on level basis. The main algorithms it consists of are: K-means algorithm, which I have described in *Section 2*, and region growing connected-components algorithm, which I will describe now.

Connected-components algorithm performs the following actions:

- (i) in the feature space, it considers only one dimension – the gray level, Z , which I will call intensity of image.
- (ii) it computes a global threshold for the intensity image using Otsu's method³
- (iii) it converts the intensity image to black&white using the computed threshold level from (ii)
- (iv) it labels the separate regions, which are distinct from the background of the black&white image.

In a black&white image the background is marked as 0, whereas objects are marked as 1. In step (iv) the algorithm finds the separated regions by checking 8- or 4-connectivity among pixels. Each separate region gets its label number and as the total result we get a labeled image.



A region is 8-connected when there exists a vertical, horizontal, or diagonal path among all pixels in this region and it is 4-connected when there exists only a vertical or horizontal path. On *Picture 3.3.1* a labeled image has the orange and blue regions 8-connected because there exists only diagonal connection between pixels (2,2) and (3,3). The blue and green regions are 4- and 8-connected because there is a horizontal path between pixels (3,4) and (3,5). The red region is not connected with any other region. In my experiment I use 8-connectivity.

Picture 3.3.1: 4- and 8-connectivity

³ MATLAB function `level=graythresh(im_gray)` performs this operation. The function minimizes the intraclass variance of the thresholded black and white pixels.

The spatial constrained image segmentation algorithm performs region growing and K-means algorithms alternatively on each level, starting with region growing first. As the result of performing any of those algorithms we get a set of

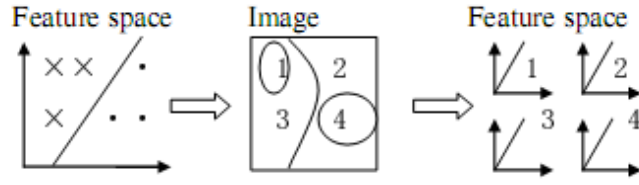


Diagram 3.3.2: Spatial Constrained K-means

new regions. Each of these regions will enter the next level separately if specific criteria are met. In other words after we perform region growing, level one is accomplished and as the result we get one or more regions. On level two on each of these regions (for which specific criteria are met) K-means is performed separately, i.e. with spatial constrains, resulting with a set of new regions. Level two is accomplished and we can enter level three performing region growing on new regions (if the specific criteria are met)... See *Diagram 3.3.2* for visual explanation.

A region will enter next level if all of these criteria are met:

- (i) the area of the region is large enough
- (ii) the level number of this region does not exceed the upper limit
- (iii) the region is divisible
- (iv) the fitness of the region is not good enough

In my experiment I set the minimum area to $A_{min} = 36$ pixels and the maximum level to $L_{max} = 4$. Criteria (iii) and (iv) are checked only on levels, on which K-means is performed. On these levels, I perform the algorithm for $K = 2$ and I calculate intra-compactness (9) of the region. If the resulting number is very close to zero the region is indivisible. Otherwise I perform K-means for $K = 2$ and $K = 3$ in order to find the fitness of the region. The fitness is defined as:

$$C = \min(C_2, C_3) \quad (6)$$

C_2 and C_3 are calculated using simplified formulas (7) and (8) respectively:

$$C_2 = \frac{d_1 + d_2}{d_{12}} \quad (7)$$

$$C_3 = \frac{1}{3} \left[\max\left(\frac{d_1 + d_2}{d_{12}}, \frac{d_1 + d_3}{d_{13}}\right) + \max\left(\frac{d_2 + d_1}{d_{21}}, \frac{d_2 + d_3}{d_{23}}\right) + \max\left(\frac{d_3 + d_1}{d_{31}}, \frac{d_3 + d_2}{d_{23}}\right) \right] \quad (8)$$

Distance d_{ij} is simply Euclidean distance between two K-vectors (centroids) c_i and c_j , whereas d_i evaluates inter-compactness of cluster c_i and is given by:

$$d_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d(v_j, c_i) \quad (9)$$

Distance $d(v_j, c_i)$ is Euclidean distance between each data vector v_j , which belongs to i -th cluster, and K-vector c_i , whereas n_i is number of vectors, which belong to i -th cluster.

Considering K-means and connected-components algorithms it can be noticed that K-means does not require information about position of each pixel – it only needs 4-D color-texture feature vectors. Connected-components, however, needs 1-D intensity feature vector and the position of each pixel. This is why, when performing the second algorithm, for each region I use a mask over the whole image, which covers all pixels not belonging to this region. The masked pixels are treated as the background and because of that, without losing the relevant information about positions of each pixel, I am able to perform the connected-components algorithm.

In my program I use function `im_labeled=sckmeans(features)` to perform spatial constrained image segmentation.

3.3. Post-processing operations.

When the process of spatial constrained segmentation is done some post-processing operations must be applied to the labeled image. Those operations are:

- (i) filtering too small areas
- (ii) eroding too thin labels
- (iii) smoothening borders of labels
- (iv) bottom-up operation – merging over-segmented regions.

When performing operation (i) there are two variables which should be taken into consideration: minimum area and maximum region number (in my experiment $A_{min} = 36$ and $R_{max} = 20$). Firstly, each label with area less than A_{min} is attached to its surrounding label. Secondly, until there exists more than R_{max} labels, the label of minimum area is attached to its surrounding label and so region number is decremented. In my algorithm I perform smoothening by applying mode filter for each pixel, which is surrounded by 2x2 area.

Decision of merging two adjacent regions as described in [1] is based on: (a) color histogram similarity and (b) edge information.

The color histogram similarity, h , is calculated using the following formula:

$$h = v_1^T S v_2 \quad (10)$$

where v_1 and v_2 are HSV color histogram vectors of two adjacent regions and S is the similarity matrix, the elements of which are given by:

$$S_{ij} = 1 - \frac{d_{ij}}{\max(d_{ij})} \quad (10)$$

where d_{ij} is the Euclidean distance between i -th and j -th bin, measured in HSV color space.

In (b) we have to determine whether there exists a real border between two adjacent labels. In order to do it detection of edges in the blocked image is done again by using SUSAN algorithm. Then we define the following sets:

- $E = \{\text{edge pixels of entire picture}\},$
- $B_{ij} = \{\text{border pixels of } i\text{-th and } j\text{-th region}\}$
- $S_i = \{\text{all pixels of } i\text{-th region}\}$

The likelihood of existence of a real border between two adjacent regions can be defined by following equations:

$$e = \frac{e_b}{\min(e_1, e_2)} \quad (11)$$

$$e_b = \frac{|E| \cap |B_{12}|}{|B_{12}|} \quad (12)$$

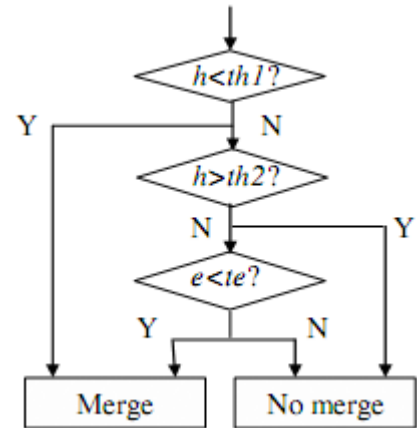
$$e_i = \frac{|E| \cap |S_i|}{|S_i|} \quad (13)$$

where $|\cdot|$ is number of elements in the set and e is the value, which determines if the border between two adjacent regions really exists.

Using *Flowchart 3.4.1* with predefined thresholds: th_1 , th_2 , t_e , and calculated values: h , e , we can decide whether two adjacent regions should be merged or not.

In my experiment I tried to merge over-segmented regions in the way described above, however I did not obtain the desired results. This is why in my experiment I only applied morphological operations (i), (ii), and (iii) to the segmented image.

In my program I perform these operations by using function `im_result=postprocessing(im_labeled)`.



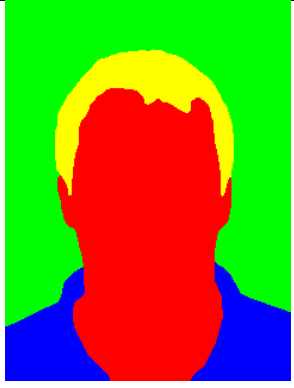
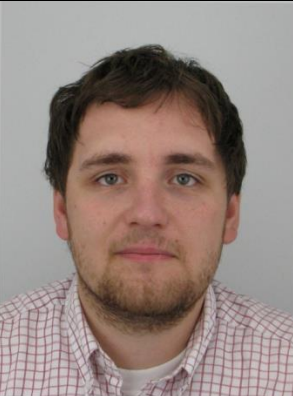

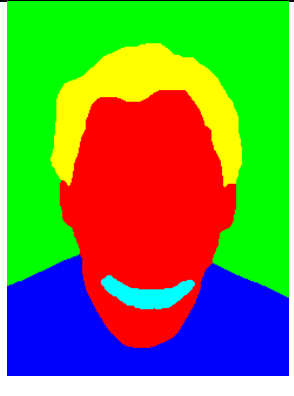
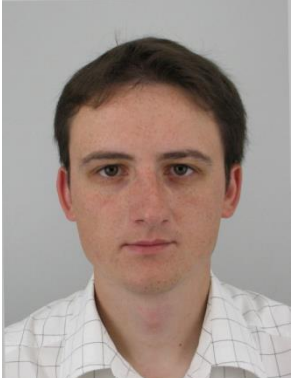

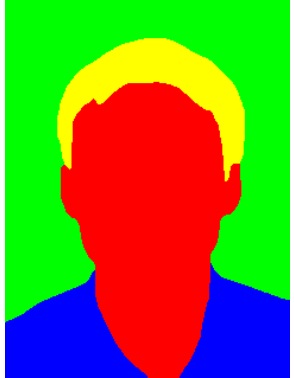


Flowchart 3.4.1: Decision tree for merging

4. Results

In this section I am going to present a set of pictures, which I have run over spatial constrained K-means algorithm with the following variables: size of block – $N = 2$, minimum area – $A_{min} = 36$, maximum level – $L_{max} = 4$, maximum number of regions – $R_{max} = 20$. These variables can be altered but they do not influence the output image much. The other variables such as fitness stay constant. In *Table 4.1* in the first column there are original images, in the second the outputs of the spatial constrained segmentation algorithm and in the third the hand-made desired segmentation images. Notice that if we merge specific regions of images from the second column, we will get images very close to the ones from the third column.

Table 4.1: Sample images and the corresponding results of segmentation

| | | |
|---|--|---|
|  |  |  |
|  |  |  |
|  |  |  |

5. Conclusion

Image segmentation is a powerful tool, which is applied almost in all image processing applications. Some of examples of usage of segmentation algorithms are: object-based compression, CBIR, or computer vision technology.

CBIR is used for searching large database of images based on some features such as shape, texture, or color, whereas computer vision technology is a science of machines that 'see'.

Although there are many segmentation algorithms, none of them is perfect. Some are not precise, while others take exponential times to resolve an image. Segmenting an image is then still an open problem of finding a quick and precise way to partition an image. In segmentation process speed is needed especially when considering real time video segmentation, which can be used, for example, when detecting the moving ball during a football match.

As you could see in the previous section, the algorithm, which I've presented, gives pretty fair results without a need to alter the input variables. If fewer regions are needed in characterizing an image, some merging algorithm can be used, which would join two similar regions together. Unfortunately the merging algorithm, which I have described, did not work in my case.

The important aspect of the spatial constrained algorithm, which I have presented, is usage of spatial constrains, which makes it possible to distinguish some regions, which would be ignored if the K-means or connected-components were applied on the whole image. Time used to partition a picture is also quite reasonable because it does not exceed half a minute (on Intel 1.74GHz Duo Core processor).

In my opinion in the very near future image segmentation algorithms will be used worldwide in search engines such as Google (CBIR technology). Soon it will be possible to upload a picture into a search engine, which will give in result all pictures containing the same objects, which can be found on the uploaded image. I think this will happen because of few reasons: internet technology progresses rapidly providing higher bandwidth connections and larger disk spaces, people's needs are never satisfied, and already now lots of images and videos are transferred over internet each day (e.g. on tulumarka.com, youtube.com).

Concluding my work, spatial constrained image segmentation is a good approach for processing an image, because it is no time consuming (due to usage of quick K-means algorithm) and it propagates through feature space using spatial constraints, which are relevant in recognition of smaller areas of the image, which would not be distinguished if spatial constraints had not been used.

6. Bibliography

[1] Luo, Yu-Fei Ma, Hong-Jiang Zhang. In *A Spatial Constrained K-Means Approach to Image Segmentation* [online].

Available from World Wide Web:

<URL: lampsrv01.umiacs.umd.edu/pubs/Papers/mingluo-03/mingluo-03.pdf>

[2] MATLAB Help Files

[3] The Mathworks, Inc. In *Color-Based Segmentation Using K-Means Clustering* [online].

Available from World Wide Web:

<URL: http://www.mathworks.com/products/demos/image/color_seg_k/ipexhistology.html>

[4] Wikipedia. In *K-means algorithm* [online].

Available from World Wide Web:

<URL: <http://en.wikipedia.org/wiki/K-means>>

[5] HIPR. In *Connected components Labeling* [online].

Available from World Wide Web:

<URL: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm>>

[6] ETFBL. In *Digitalna obrada slike* [online].

Available from World Wide Web:

<URL: <http://dsp.etfbl.net/dip/>>

[7] FER. In *Digitalna obradba slike* [online].

Available from World Wide Web:

<URL: <http://dosl.zesoi.fer.hr/vjezbe.html>>

7. Abstract

In my work I give description of image segmentation algorithm, which uses spatial constraints of feature space, i.e. on each next level it performs separate segmentation of segmented regions from the previous level. The main two modules of the program are: top-down segmentation process, and bottom-up merging process. In top-down segmentation process two main algorithms are used: K-means, which quickly clusters spatially constrained feature space vectors, and region growing connected-components, which labels distinct regions on spatially constrained intensity image. I use color-texture feature space for K-means algorithm, which not only contains information about color but also about edges of the image. In bottom-up process morphological operations are applied to the labeled image, as well as, merging over-segmented regions is performed based on color histogram similarity and edge information of the image.

Keywords: image segmentation, spatial constraints, K-means, region growing connected-components, merging