UDC 621.391

TEXTURE IMAGE SEGMENTATION BASED ON CLASSIFICATION, SKELETONIZATION AND CROSSING OF CONTOUR ELEMENTS

H.M. ALZAKI

Belarusian State University of Informatics and Radio Electronics
P. Brovki, 6, Minsk, 220013, Belarus

Submitted 23 May 2016

A method for texture image segmentation based on classification, skeletonization and crossing of contour elements is presented. The essence of the method consists in the contouring of the image, determining the position of contour elements in the image of different types (points, lines, and shapes) converting closely spaced similar contour elements into binary regions objects, binary coding mutual position obtained areal objects within the boundaries of the image segmentation are resulting in code matrix.

Keywords: texture image segmentation, classification, contour elements.

Introduction

Texture segmentation is among the most challenging problems in image segmentation. The problem already begins with the definition of textures. The human eye can easily recognize different textures. Fig. 1 shows different images. Image segmentation techniques can be classified into two broad groups – (1) region-based, and (2) contour-based approaches [1]. First group approaches try to find partitions of the image pixels into sets according to coherent image properties such as brightness, color and texture. Second group approaches start with a first stage of edge detection, followed by a linking process that seeks to exploit curvilinear continuity. These two approaches need not be different from each other. Boundaries of regions can be defined to be contours. If one enforces closure in a contour-based framework [2, 3] then one can get regions from a contour-based approach. In contour-based approaches, often the first step of edge detection is done locally. Subsequently efforts are made to improve results by a global linking process that seeks to exploit curvilinear continuity. A criticism of this approach is that the edge or no edge decision is made prematurely. To detect extended contours of very low contrast, a very low threshold has to be set for the edge detector (Fig. 1). Different methods used for texture analysis and texture segmentation, one of them – the energy map, which are widely used in texture analysis and can be used for texture segmentation. Their disadvantage is the high computational complexity of the simple texture and separation with a high degree of homogeneity.

In this paper, it is proposed to use contour elements for texture analysis. However these approaches are based on a small set of contour primitives and sufficiently rough statistical evaluation, which leads to significant errors separation of complex textures.

Contour analysis (e.g. edge detection) may be adequate for untextured images, but in a textured region it results in a meaningless tangled web of contours. Think for instance of what an edge detector would return on the bean region in (Fig. 2). The traditional “solution” for this problem in edge detection is to use a high threshold so as to minimize the number of edges found in the texture area. This is obviously a non-solution such an approach means that low-contrast extended contours will be missed as well.

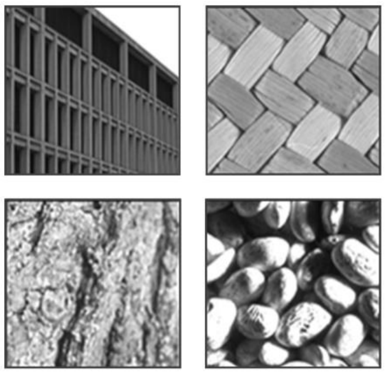
Fig. 1. Some challenging images for a segmentation algorithm. The goal is to develop a single grouping procedure which can deal with all these types of images

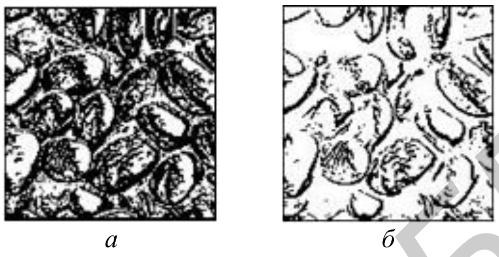
Fig. 2. Demonstration of texture as a problem for the contour process. Each image shows the edges found with a Prewitt filter and Roberts filters for the penguin image using different thresholds:
a – low threshold; *b* – high threshold

Texture image segmentation based on energy map

Widely spread for texture analysis is the method of energy maps [4, 5], based on the image filtering, weighting and summing the results. The essence of the methods to use filters with smooth changing of brightness, smooth contour lines, broken contour lines, contour points and small spots with constant or slowly changing brightness; resulting in the formation of binary matrices, single elements that indicate the presence of corresponding positions of certain objects; weighting obtained matrices and their summing element-wise, resulting in an energy map (Fig. 3). Image texture has a number of perceived qualities which play an important role in describing texture. Laws [5] identified the following properties as playing an important role in describing texture: uniformity, density, coarseness, roughness, regularity, linearity, directionality, direction, frequency, and phase. Laws texture energy measures determine texture properties by assessing Average Gray Level, Edges, Spots, Ripples and Waves in texture. The measures are derived from three simple vectors. L3 = (1, 2, 3) which represents averaging; E3 = (–1, 0, 1) calculating first difference (edges); and S3 = (–1, 2, –1) corresponding to the second difference (spots). After convolution of these vectors with themselves and each other, five vectors result:

Level L5 = [1, 4, 6, 4, 1]
Edge E5 = [–1, –2, 0, 2, 1]
Spots S5 = [–1, 0, 2, 0, –1]
Ripples R5 = [1, –4, 6, –4, 1]
Waves W5 = [–1, 2, 0, –2, –1]

Energy map contains mainly smoothly changing in the brightness, which can be separated by using known image segmentation methods, such as region growing [6, 7]. To improve the texture segmentation (to eliminate the influence on the result of high-frequency noise and joining smaller closely objects to the brightness of large objects) before applying the method of region growing the energy map is processing with low-pass filtered using a Gaussian filter.

The method of energy map is not strictly defines the types of filters that can be used to select image elements, which in theory makes the method suitable for the determination of arbitrary texture. However, combining the results of filtering in the energy map can lead to a segmentation fault, especially for complex textures, due to the lack of ranking results by relevance filtering. Eliminating this disadvantage is possible due to: a) the use of contour elements for image texture analysis as the most homogeneous and therefore equivalent to merge the results; b) a clearly classification of contour elements of the structure to form a universal device for describing both simple and complex textures; c) logically combining for the results of filtering.

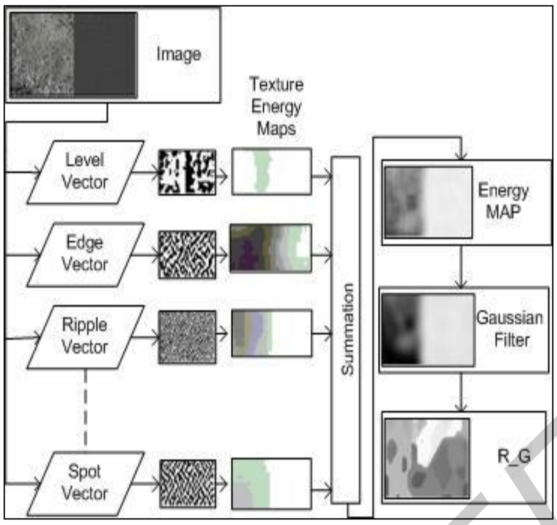

Fig. 3.Texture image segmentation based on the energy map

Texture image segmentation based on classification, skeletonization and crossing of contour elements

A method for texture image segmentation based on classification and skeletonization and crossing of contour elements. The essence of the method consists the contouring of the image, determining the position of contour elements of the image (points, lines, and shapes) of various types, transformation closely spaced similar contour elements into binary area objects, binary coding mutual position obtained polygon objects within the boundaries of the original image segmentation are resulting as matrix code (Fig. 4).

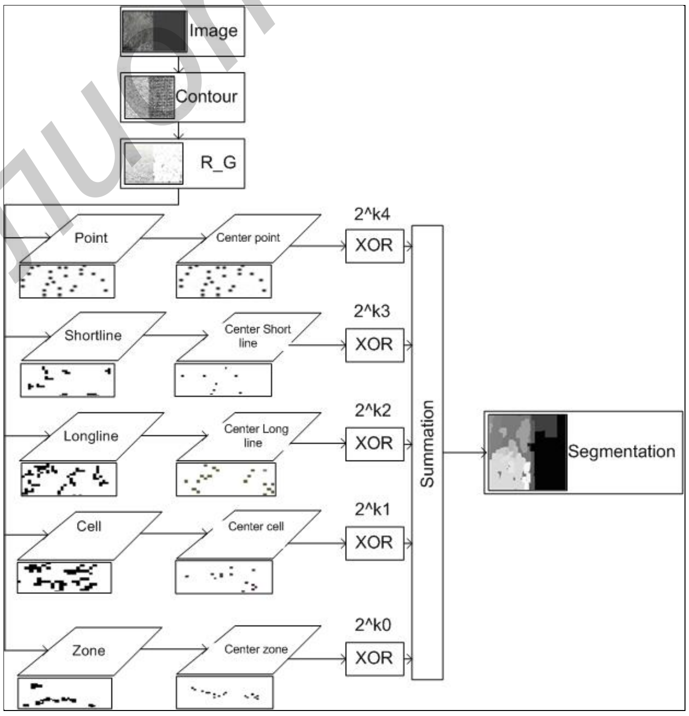

Fig. 4. Algorithm of texture image segmentation based on classification, skeletonization and crossing of contour elements

Algorithm of texture image segmentation based on classification, skeletonization and crossing of contour elements consists of the following steps.

Step 1. Contour filtering for the image. In this step may be any filter. In order to achieve high speed processing at a sufficiently high quality selection of contour lines is suggested to use the results of the union of the two loops filter: Roberts [8] and Prewitt [9], allocating brightness drops diagonally, horizontally and vertically (Fig. 5). The result is a binary matrix in which the elements correspond to single isolated contour points forming lines of varying difficulty and length, smaller area features.

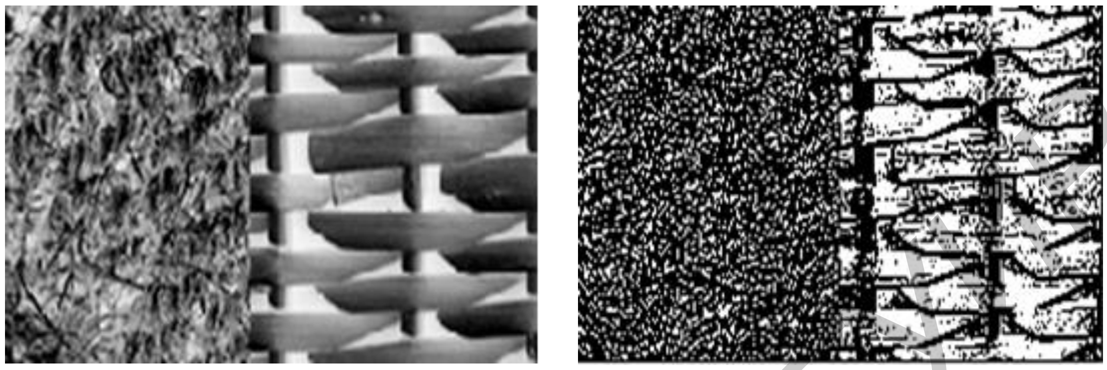
a *б*

Fig. 5. Contour filtering for the image: *a* – original image; *b* – image contour by Roberts and Prewitt filters

Step 2. Segmentation of contour element. To perform this step, the algorithm may be any segmentation method. Most effective in this case is the method of region growing (Fig. 5), which provides high speed and accuracy. As a result of this step is formed by matrix, in which background elements corresponds to zero segment number, and other elements.

Step 3. Classification of the contour elements. At this step of the algorithm are allocated points, consisting of one or two pixels; short lines formed by two or three pixels; long line having two end points and formed by four or more pixels (limiting the number of pixels in the line length can be determined by processing the histogram of lengths of lines based on the size of the processed image) (Fig. 6); «Cell», formed by the intersection of several contour lines; «Spot» a small area objects with no endpoints; large area objects. All identified contour elements except large area objects are discussed later as a texture image. As a result of this step, each contour elements having it is number, is associated with the identifier (Descriptor), of the plurality of identifiers, relating it to a certain class. In the Table are shown the identification parameters used for the labeling of contour element are shown. Each row of Table is histogram of the number of neighboring contour pixels of different contour elements. Contour element «Point» (row 1) does not have end points and contains less than 5 neighbors contour pixels. Contour element «Short direct line» (row 2) comprises two endpoints and a small number neighbors contour pixels. Contour element «Short curvy line» (row 3) has two endpoints with a number of neighboring contour pixels. Contour element «Spiral» (row 4) is shorter curvy line with sharp bends, which has two end points and points with more than two neighbors contour pixels in places bends. Contour element «Cell» (row 5) has three or more end points and a plurality of contour pixels with a large number of neighbors. Contour element «Spot» (row 6) does not have end points, but contains a large number of contour pixels and number of neighbors more 4 pixels. Contour element «long direct line» (row 7) has two end points and a large number neighbors contour pixels.

Step 4. Classe the contour elements. A result of this step of the algorithm is formed by a plurality of bit planes, the unit elements each of which indicates the position of the respective contour elements of a certain class.

Step 5. Discretization of contour elements. At this step the contour elements replacing each bit plane a plurality of equidistant points. Each point and short line is assigned one point. Long lines are replaced by series points. «Cell» and «spots» are assigned to the grid points. The distance between the points in the series and the mesh is approximately the same. This makes it possible to align the input contour elements of different types to the energy generated at the next step of the integral of the binary image (Fig. 7).The result is a plurality point bit planes.

Identification parameters and identifiers contour elements

Classes of Contour elements	Identification parameters								
	Number of neighbors contour pixels								Number of end points
	1	2	3	4	5	6	7	8	
Point	Less than 5 neighbors of contour pixels								
Short direct line	2	2	1	0	0	0	0	0	2
Short curvy line	2	1	0	2	3	0	1	0	2
Spiral	2	0	0	7	3	1	4	2	2
Cell	4	1	2	10	4	8	3	9	4
Spot	0	0	9	1	3	10	5	12	0
Long direct line	2	14	5	1	0	1	0	0	2

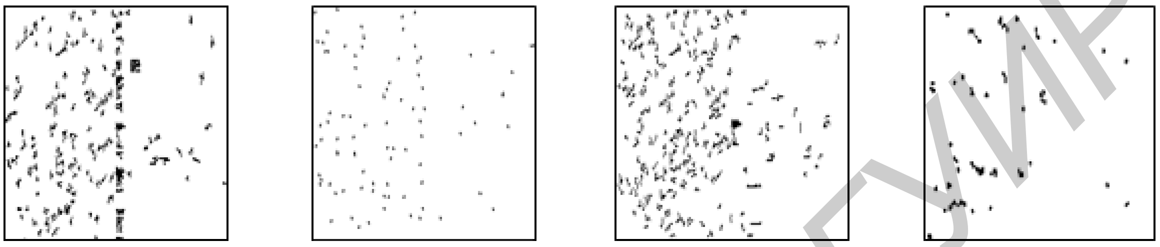
a *b* *c* *d*

Fig. 6. Classification of the contour elements according to number of neighbors of contour pixels and end points: *a* – «Cell»; *b* – «Point»; *c* – «Long line»; *d* – «Spot»

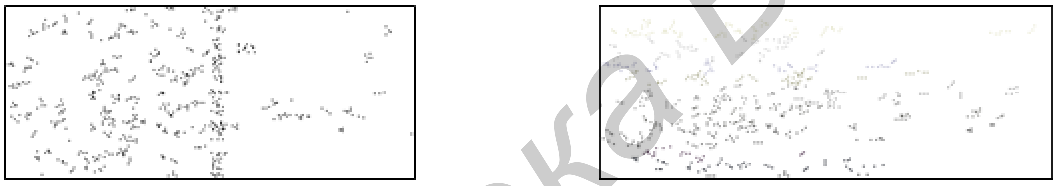
a *b*

Fig. 7. Discretization of contour elements: *a* – cell is broken and replaced to number of contour pixels; *b* – long line is broken and replaced to number of contour pixels

Step 6. Determine histogram of the size of mask. As a result for this step it is found the histogram that starts with mask 3×3 and this size are increasing by 2(5×5, 7×7, 9×9,…….) until it finds neighbor for this contour element and finely result will be histogram and according for this histogram it will be found the size of the mask which will be used for next step.

Step 7. Segmentation each of contour element. As a result of this step is a bit plane in which each contour element have number according to the size of the mask in the above step. For this step it can be used any effective and fast segmentation method for this step, the method of region growing is used.

Step 8. Checking and comparison of the results of (Step 5) and (Step 7). According to this step to fill and return each contour element after (Step 5) but in this case according to the size of mask and the pixel that will be in the region of the mask will take the same number in the matrix of (step 7) as shown in the (Fig. 8).

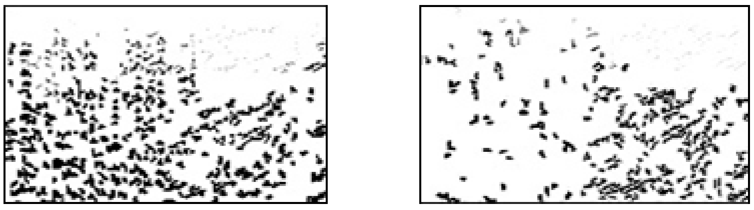
а *б*

Fig. 8. Replace contour element according to compere (Step 5 and 7): *a* – cell with new numbers according to the size of mask; *b* – long line with new numbers according to the size of mask

Step 9. Flooding each of contour pixel. Replace each contour pixel by the value of the contour element. In this step each region has new number and it will indicate to texture region as shown in the (Fig. 9).

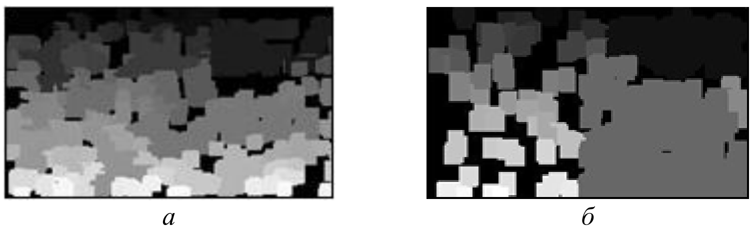

Fig. 9. Flooding each pixel according to the size of the mask: *a* – cell after flooding; *b* – long line after flooding

Step 10. Logical combination of classes. A result of this step is performed by the construction of the resulting integrated image obtained a result of the concatenation of the previous step integrated binary images. In the simple case, the number of bit planes in the resulting integral image replace with the initial binary number of integral images. In the simple case, the number of bit planes in the resulting integral image coincides with the initial binary number of integral images. If necessary, the logical association of contour elements of certain classes (points, short lines, cells and spots) before the formation of the resulting integrated image is a logical addition of the corresponding binary integral images as shown in the (Fig. 10).

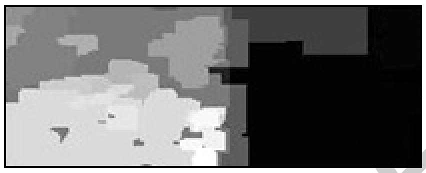

Fig. 10. Logical combination of classes

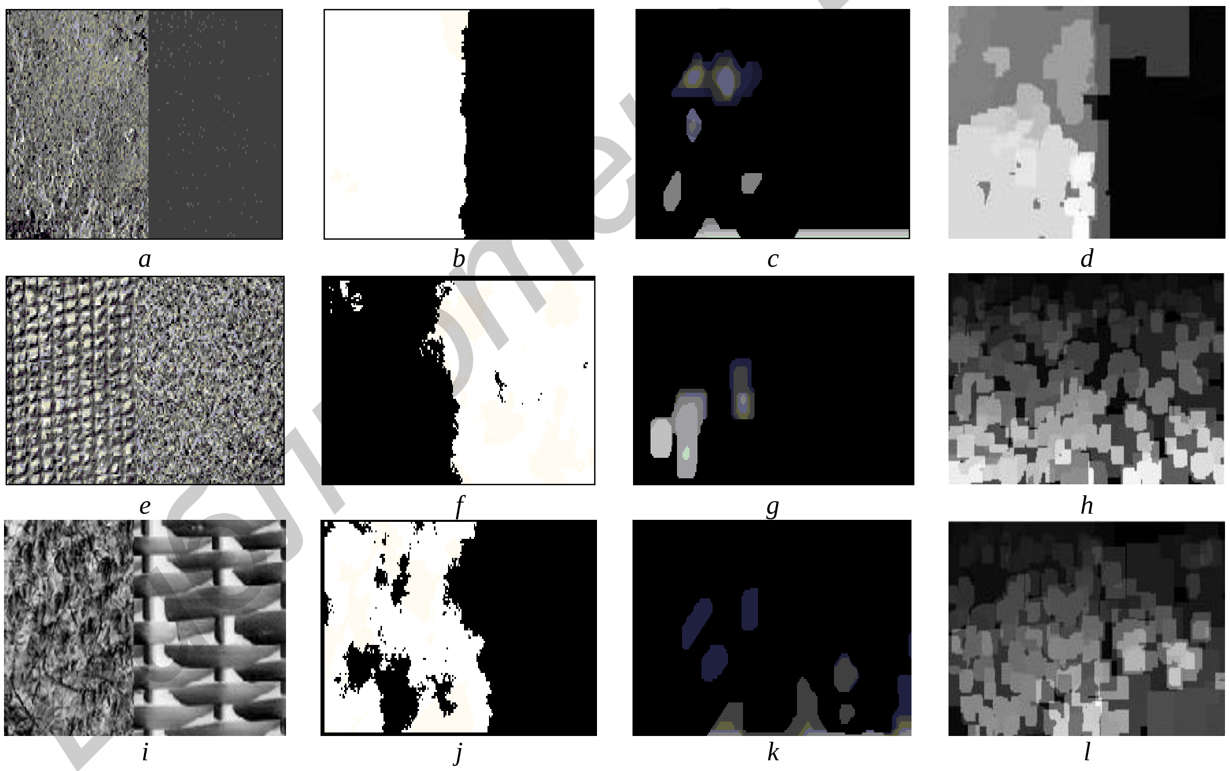

Fig. 11. Testes and results of image texture segmentation: *a* – Test 1; *b* – Test 1 Texture images segmentation based on classification of contour elements and logical addition of classes; *c* – Test 1 segmentation by energy map; *d* – Test 1 Segmentation by proposed method; *e* – Test 2; *f* – Test 2 Texture images segmentation based on classification of contour elements and logical addition of classes; *g* – Test 2 segmentation by energy map; *h* – Test 2 segmentation by proposed method; *i* – Test 3; *j* – Test 3 Texture images segmentation based on classification of contour elements and logical addition of classes; *k* – Test 3 segmentation by energy map; *l* – Test 3 segmentation by proposed method

Conclusion

A texture image segmentation based on classification, skeletonization and crossing of contour elements determine the position of contour elements in the image of different types (points, lines, and shapes) and determine the distance between each contour pixel according to the distance determine the size of the mask which will be used to flood each of contour elements and get different texture region. The proposed method compared with the method based on the energy map which depends on the summation for result of different filters and with the method of texture images segmentation based on classification of contour elements and logical addition of classes which depends on the summation for different binary plans. The proposed method provides less segmentation error comparing with the method based on the energy map and the method of texture images segmentation based on classification of contour elements and logical addition of classes. For comparing test images, published in the test data bases Brodatz and UIUCTex, were used.

References

1. *Ashraf M., Safari B.D., Zaki N.* // International Journal of Computer Science& Information Technology (IJCSIT). 2011. Vol. 3. №5. P. 99-106.
2. *Elder J., Zucker S.* // Conf. Computer Vision, Cambridge. England. 2005. Vol. I. P. 399-412.
3. *Jacobs D.* // IEEE Trans. Pattern Anal. Mach. Intell. 1996. Vol. 18. №1. P. 23-37.
4. *Lee D-Ch., Shchenk T.* // A Collection of Papers Presented At the XVII Congress of ISPRS. 1992. №48. P. 75-80.
5. *Ertuğrul Ö.* // International Journal of Intelligent Information Systems. 2014. Vol. 3. №2. P. 13-18.
6. *Kumar M., Mehta K. K.* // International Journal of Computer Technology and Applications. 2011. Vol. 2. №4. P. 855-859.
7. *Sharma Ritu, Sharma Rajesh* // International Journal of Innovative Research in Computer and Communication Engineering. 2014. Vol. 2. P. 5686-5692.
8. *Muthukrishnan R., Radha M.* // International Journal of Computer Science & Information Technology (IJCSIT). 2011. Vol. 3. №6. P. 259-267.
9 *Shrivakshan G., Chandrasekar C.* // IJCSI International Journal of Computer Science Issues. 2012. Vol. 9. №1. P. 269-276.