EXTRACTION OF HOMOGENEOUS FINE-GRAINED TEXTURE SEGMENTS IN VISUAL IMAGES

A. Goltsev[*], *V. Gritsenko*[*], *D. Húsek*[†]

Abstract: A new heuristic algorithm is proposed for extraction of all homogeneous fine-grained texture segments present in any visual image. The segments extracted by this algorithm should comply with human understanding of homogeneous fine-grained areas. The algorithm sequentially extracts segments from more homogeneous to less homogeneous ones. The algorithm belongs to a region growing approach. So, for each segment, an initial seed point of this segment is found. Then, from this initial pixel, the segment begins to expand occupying its adjacent neighborhoods. This procedure of expansion of the segment continues till the segment reaches its borders. The algorithm examines neighboring pixels using texture features extracted in the image by means of a set of texture windows. The segmentation process terminates when the image contains no more sizable homogeneous segments. The segmentation procedure is fully unsupervised, i.e., it does not use a priori knowledge on either the type of textures or the number of texture segments in the image. Using black and white natural scenes, a series of experiments demonstrates efficiency of the algorithm in extraction of homogeneous fine-grained texture segments and the segmentation looks reasonable "from a human point of view".

Key words: *texture feature, texture window, homogeneous fine-grained texture segment, extraction of texture segment, texture segmentation, "float" coding method*

Received: June 22, 2016
Revised and accepted: September 21, 2017
DOI: 10.14311/NNW.2017.27.024

1. Introduction

The problem of image segmentation is very important in the tasks involving analysis and recognition of different natural visual scenes, such as landscapes, satellite photographs, microscopic medical images, robotics, etc. (e.g. [3, 6, 11, 16, 30, 50]]). A rapid image segmentation system should be useful for the task of abandoned object detection [56].

*Alexander Goltsev – Corresponding author; Vladimir Gritsenko; International Research and Training Center of Informational Technologies and Systems, National Academy of Sciences of Ukraine, Pr. Glushkova 40, Kiev 03680 Ukraine, E-mail: agoltsev@adg.kiev.ua

†Dušan Húsek; Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodarenskou vezi 271/2,182 07 Prague 8, Czech Republic, E-mail: dusan@cs.cas.cz

Generally, any object is recognized, to a large degree, by its shape. Therefore, in order to recognize an object in a real-world visual scene, it is necessary, at first, to detect its borders in the image. In natural images, texture borders are, often, borders of objects. Thus, the first operation of the object recognition procedure in a natural scene should be division of the image into separate texture segments. When texture segments are discovered, it becomes possible to start solving such tasks as object-ground separation and object recognition. However, according to a widespread belief, object recognition based on low-level cues cannot and should not aim to produce a complete final correct recognition. Mid- and high-level knowledge should be used to either confirm these segments or select some of them for further analysis [50].

The idea to start the analysis of any visual image from extraction of all homogeneous texture areas is conventional. For example, "The task of partitioning a natural image into regions with homogeneous texture, commonly referred to as image segmentation, is widely accepted as a crucial function for high-level image understanding, significantly reducing the complexity of content analysis of images" [42].

In the majority of works presented in literature, statistical analysis of input images is performed for description, recognition and segmentation of textures. In the statistical approach, the image is usually processed by a sliding window within which various statistical characteristics are measured, such as energy, entropy, uniformity, contrast, dispersion, inverse difference moment, total average, total dispersion, total entropy, difference dispersion, difference entropy and correlation, etc. (e.g. [4, 8, 14, 38, 54]).

Lately, so called "superpixels" have been used in many works (e.g. [1, 19, 36]). Superpixel algorithms group image pixels into perceptually meaningful atomic regions, which can be used to replace the rigid structure of the pixel grid. They capture image redundancy, and greatly reduce the complexity of subsequent image processing tasks.

In general, a problem of texture segmentation implies that the segmentation procedure is performed without any preliminary knowledge of the input image. In this formulation, the problem is very complicated. In the approach which falls into category of unsupervised texture segmentation the segmentation algorithm uses some universal texture features to extract any texture regions (e.g. [9, 15, 40, 47, 52, 53, 55]). For the majority of the unsupervised segmentation algorithms it is necessary to specify the number of the texture segments to be extracted.

The complexity of the texture segmentation problem may be considerably facilitated by means of providing a segmentation device with a limited number of distinctive patches of those textures that should be recognized and segmented in the image. Using these patches, it is possible to measure characteristics of the indicated textures and, then, to adjust parameters of the segmentation device according to them. Adjustment of segmentation parameters may be done by means of supervised learning. In this approach, the segmentation device is preliminarily learnt on a training set of texture class samples (e.g. [2, 5, 41, 51]). A number of supervised feed-forward neural network-based models for texture segmentation and recognition are of the same approach (e.g. [10, 20–23, 31, 32, 48, 51]). These models often use supervised neural network classifiers, such as [31–34, 37, 46], and may use associative (assembly) neural networks [17, 18, 20–23].

The textures are often subdivided into fine-grained and coarse (-grained) ones (e.g. [49]). Image regions representing objects with discontinuities in depth, in material, or in illumination, etc., correspond to coarse texture segments. And fine-grained texture is characterized by a smooth variation of image brightness and fine granularity. Texture segmentation of images may be performed both by means of coarse texture region extraction and by extraction of homogeneous fine-grained texture segments. The presented attempt belongs to the latter approach. It is worth noting that description of textures as "coarse" or "fine-grained" is obviously relative. Indeed, by increasing the degree of image resolution, many fine-grained segments may turn into coarse ones. Therefore, in the presented case, the size and image resolution are fixed.

It is well known that the "texture" is an intuitive concept which does not have any formal or commonly accepted definition (e.g. [48]). Therefore, there is no possibility to evaluate formally and objectively any texture segmentation results. Various researchers understand different things by the term "texture". In many publications, the term "texture" is used in relation to the areas occupied by the whole objects such, for example, as faces, flowers, animals, trees, buildings, bicycles and so on. This understanding of the texture notion is predominant in the world. Therefore, all well-known texture and object segmentation and recognition databases are constructed for testing different segmentation and recognition algorithms within this understanding of texture.

In contrast to this understanding of texture, we consider only "homogeneous texture segments" in our attempt. That is, under the term "texture segment" we mean only such image area all small parts of which have similar texture characteristics (texture features), and we use the expression "homogeneous texture segment" to emphasize that.

The aim of our research was to develop a texture segmentation algorithm that would provide only a partial texture segmentation of any input image, but the segmentation should be reasonable "from a human point of view". The expression "partial texture segmentation" means that the proposed algorithm is intended for extraction of only homogeneous fine-grained texture segments present in the analyzed image. In other words, the task is to extract large homogeneous fine-grained texture segments in the image in such a way as humans do.

The proposed segmentation algorithm belongs to a region growing approach (e.g. [12, 29, 39, 44]). So, for each segment, an initial seed point of this segment is found. Then, from this initial pixel, the segment begins to expand occupying its adjacent neighborhoods. This procedure of expansion of the segment continues till the segment reaches its borders. During the process of expansion of the extracted segment, in order to decide whether the neighboring pixel should be added to the segment, or not, the algorithm examines the pixel using texture features extracted in the image by means of a set of texture windows. A texture window is a comparatively small square frame sliding inside the image and used to evaluate texture characteristics of all image points. The window-based method for evaluation of textures is the most prevalent technique now.

The paper is organized as follows. Section 2 gives an overview of the complete segmentation procedure and, in particular, briefly describes the proposed algorithm. The set of used texture features and its representation are described in

Section 3. The procedure of dilation of an arbitrary one-valued area in a binary image is explained in Section 4. Section 5 presents the formation of exemplary feature pattern characterizing the sought-for texture segment. Section 6 contains the algorithm of comparison between the feature pattern of texture window and the exemplary pattern. A short description of the algorithm for extraction of one homogeneous fine-grained texture segment is given in Section 7. Full description of this algorithm is found in Appendix A. Section 8 demonstrates several experiments on texture segmentation of natural images by the algorithm. Sections 9 and 10 are devoted to discussion and conclusions. We summarize all concepts (terms) that are crucial for proposed algorithms understanding, in Appendix B, to be easily available in the course of reading the paper.

2. An overview of the texture segmentation process

The proposed algorithm extracts all homogeneous fine-grained texture segments in sequential and iterative process, one segment per iteration. In each iteration, first of all, an initial seed pixel is found which certainly belongs to the most homogeneous fine-grained texture segment present in the currently considered region of the image. A set of texture features is extracted from this pixel and its close surroundings. We postulate that the found seed pixel is representative of the sought-for most homogeneous texture segment and the extracted feature set adequately characterizes this segment. The procedure of finding the representative seed pixel and the characterizing feature set is described in detail in [25] and, therefore, it is not considered here. Earlier (and outdated) version of the procedure is presented in [24].

The algorithm is fully unsupervised; it processes any input image without any a priori knowledge of either the type of textures or the number of texture segments in the image. As mentioned above, the algorithm sequentially extracts only fine-grained homogeneous texture segments. This peculiarity does not allow it to extract coarse texture segments as entire areas. Moreover, the algorithm should divide every coarse texture region into a number of homogeneous fine-grained texture segments. This inherent peculiarity distinguishes the proposed algorithm from related techniques that extract both fine-grained texture segments and coarse ones. All homogeneous fine-grained texture segments are sequentially extracted by the algorithm starting from more homogeneous to less and less homogeneous ones.

The segmentation process terminates when the image contains no more sizable homogeneous segments. After the segmentation procedure is completed, the coarse texture and non-texture regions of the image remain not extracted.

The essence of the procedure of finding the representative seed pixel is as follows. The image is covered with a number of test windows. In each of them, a degree of texture homogeneity is measured. The test window with maximal degree of homogeneity is determined and a representative pixel seed is detected within this test window.

The found representative seed pixel is gradually grown to the boundaries of the currently extracted most homogeneous fine-grained texture segment taking into account the characterizing feature set. As a result of this expansion procedure, an approximately homogeneous and (usually) simply connected fine-grained texture area is extracted.

After extraction of the current segment its area is excluded from further consideration. Thereby, each subsequent segment will not intersect with the segments that have been extracted earlier. At the next iteration, extracting the next texture segment starts again with the search for a new representative seed pixel belonging to a new most homogeneous segment present in the remainder of the image.

At the final stage of the segmentation procedure, averaged texture characteristics of all extracted texture segments are analyzed and compared with each other with the aim to merge those texture segments that belong to the same texture class.

3. Texture features and their representation

It is well known in the pattern recognition field, that the choice of features used by the recognition technique strongly influences the results of the recognition (segmentation) process. Accordingly, the texture segmentation results strongly depend on how well the chosen set of features describes the textures. Below, we describe the feature set used for experiments presented in this paper. However, it is worth noting that the algorithm for extraction of homogeneous fine-grained texture segments proposed in the paper can work with any other feature set, provided that this set is represented in the format of normalized binary vectors (see below).

In this paper we consider only grayscale images, so that every input image is an integer matrix of $I \times J$ size every element of which represents the brightness value (in the range of 0–255) of the corresponding image pixel. In order to evaluate texture characteristics of different image points, square texture windows of the same size are applied.

A set of M texture features (feature types) is computed for every texture window. The window-based method for feature evaluation of textures is the most prevalent technique now. The texture features extracted from texture windows serve for estimation of similarity and/or difference between the corresponding windows. The brightness values of all pixels of the window take part in computation of every texture feature. All texture features computed in a texture window are associated with its central pixel.

The whole input image is covered by a large number of overlapping texture windows, let us designate this number as N. Let us stipulate (for simplification) that there are N texture windows and that every image pixel is a center of the corresponding texture window. So, we introduce a coordinate system of the window centers of $N = I \times J$ pixels with indices i and $j (i = 1, 2, 3, \ldots, I; j = 1, 2, 3, \ldots, J)$. At the beginning of the segmentation process of a given image, all texture features are computed in all N texture windows covering the image and, then, are saved for further usage.

The following set of texture features is used. First, the feature set includes a brightness histogram of all pixels of a texture window. Each bin of this histogram represents the number of the window pixels that have brightness values within a certain interval. Brightness histogram of a texture window gives not unique but rather specific description of the texture: the histograms of different windows of the same texture are similar, while the histograms of the windows belonging to different textures are dissimilar almost in all natural images. Of course, it is possible to design different artificial textures with the same brightness histograms, but in natural images, it is highly improbable to find contiguous areas of dissimilar textures that have the same brightness histograms. In the described case, the brightness histogram consists of 12 bins.

Second, the feature set includes a histogram of orientations of all pixels of a texture window. This histogram is computed, based on filtering the image by the Scharr filter, as described in [28]. As a result of filtering the initial brightness matrix of the image by the Scharr filter, two real values are assigned to every image pixel. These values define vertical and horizontal brightness differences around the pixel. They are calculated from the brightness values of those 8 pixels that directly surround the considered one. Vertical and horizontal differences are used to evaluate the angle of brightness orientation around each image pixel. The angle is a real number from the interval $[0, \pi]$. This interval is divided into V angle ranges of π/V size each, where V denotes the number of bins in the orientation histogram. The orientation histogram is necessary to distinguish texture segments that contain differently oriented strokes. Orientation histogram consists of 9 bins.

The next feature is a characteristic of brightness non-uniformity within a texture window. This feature is computed as follows. As above, the whole initial brightness matrix is processed by the Scharr filter and vertical and horizontal brightness differences are determined for each image pixel. For every pixel, these values are squared, summarized, and the square root is taken from this sum. Then, the values obtained for each pixel of the window are summarized into a single resulting value which serves as the brightness non-uniformity characteristic of the window. This feature indicates how many brightness inhomogeneous spots are within the texture window. The feature is useful to distinguish texture patches of the same mean brightness but with different inner structure.

The last texture feature is the mean brightness of a texture window. Evidently, it is a very distinctive characteristic of textures, as well.

So, the total of $M = 23$ texture features are computed for each texture window. Every feature value computed in a texture window is a real number (a scalar). All feature values are normalized to a range K. All these M normalized values of texture features computed in a texture window constitute a feature pattern describing the corresponding texture window; this pattern is associated with a proper pixel of the image. To represent a feature pattern of one texture window in a binary format, M binary (column) vectors of K elements each are used. Each of these M vectors comprises $M \times K$ binary elements. Let us call such a set of $M \times K$ binary elements a binary feature cluster. Full description of textural characteristics of each image is represented by the set of N such clusters (according to the overall number of texture windows covering the image).

In the experiments with the computer program simulating the texture segmentation algorithm, each feature cluster consists of $M = 23$ binary (column) vectors of $K = 20$ elements each. The choice of a relatively small vector size ($K = 20$ binary elements) in the experiments is explained by the desire to reduce maximally the processing time which obviously increases in proportion to the number of cluster vectors and their sizes.

Within each cluster, the vectors are numbered separately in the same way, so that the vectors of different clusters that represent the same texture features of different windows have the same numbers inside the clusters. Hereinafter, index m is used for indexing vectors within the feature clusters.

The "thermometer" coding method [45] is used to represent the normalized feature values in the binary vectors. Every normalized feature value is encoded by a set of one-valued elements in every vector. The "thermometer" coding method implies that this set of one-valued elements includes all elements of the vector between the numbers 0 and U, where U is the normalized feature value to be represented in the vector. So, the larger the feature value U, the higher the continuous set of one-valued elements rises in the vector. Let us underline that according to the name of the method, the continuous set of one-valued elements of the vector always begins with the element number 0. This description is depicted in Fig. 1, in which the feature value $U = 7$ is represented by the "thermometer" coding method in the binary vector of $K = 13$ elements.

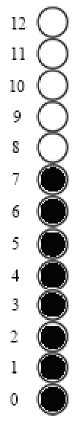

Fig. 1 *An example of representation of the feature value by the "thermometer" coding method in the binary vector.*

Here, it is worth to explain the sense of introduction of the thermometry representation of the texture feature values. In the proposed segmentation algorithm, the pattern of texture features extracted from each texture window is compared with some etalon feature pattern, as will be described below. By means of such a comparison, the degree of similarity between these patterns is evaluated.

As mentioned above, the majority of the used texture features are presented in a bar graph format. We consider that the most reasonable method for evaluation of similarity between two feature patterns presented in the format of histograms is to measure the overlap between the corresponding bins of the histograms. Thereby, for easy measurement of the overlap between the corresponding bins, we introduce the "thermometry" coding method to represent the feature values fixed in each bin of the histograms. Expediency of using the "thermometry" coding method exactly in the binary format will become clear in Section 5.

So, all M normalized feature values associated with a texture window are represented by different continuous sets of one-valued elements in each of M vectors of the binary feature cluster. Full pattern of one-valued elements in a cluster provides an integral feature description of the texture in the corresponding texture window. All N binary feature clusters contain the feature representation of the whole input image (of all N texture windows). Let us introduce a binary four-dimensional array $\mathbf{G}[i][j][m][k]$ of $I \times J \times M \times K$ elements for formal description of such a feature representation of an image.

Actually, the array $\mathbf{G}$ is composed of $N = I \times J$ binary feature clusters. In this array, the indices (i, j) are coordinates of the (i, j)-th image pixel, and the (i, j)-th window, and the (i, j)-th feature cluster. The index m $(m = 0, 1, 2, \ldots, M)$ determines the feature type number and the vector number inside the (i, j)-th cluster. The index k $(k = 0, 1, 2, \ldots, K)$ indicates the element number within the m-th binary vector.

Thus, as a result of the feature extraction procedure, the original input image is transformed into the binary four-dimensional array $\mathbf{G}[i][j][m][k]$ which contains feature description of the whole image in a binary format. The array $\mathbf{G}$ is computed only once at the beginning of the texture segmentation process and saved after that for further usage.

Fig. 2 demonstrates an example of the whole aggregate of N feature clusters (the array $\mathbf{G}$) which describes the input image. The figure depicts each feature

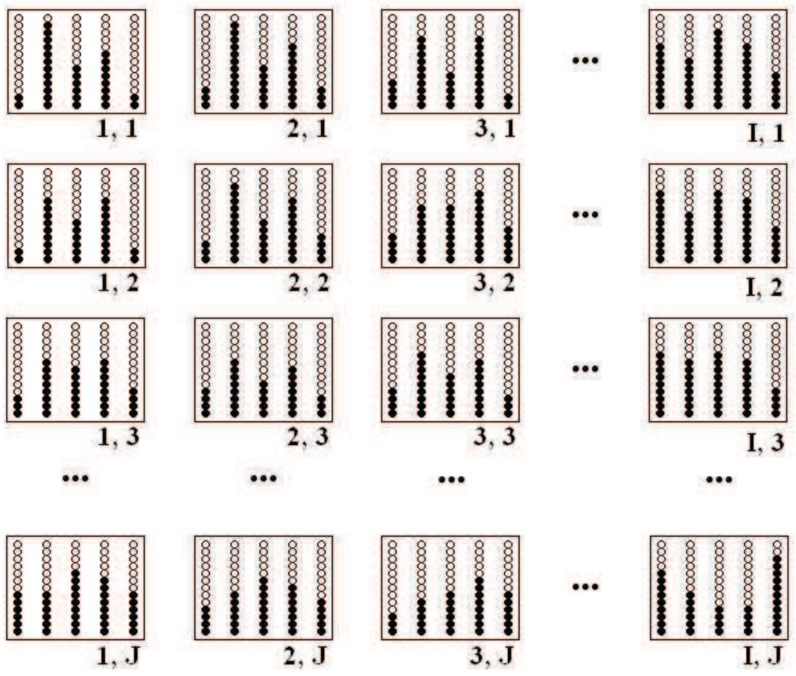

Fig. 2 *A schematic picture of the aggregate of N feature clusters (the array* **G***) describing the input image.*

cluster as consisting of $M = 5$ vectors each with $K = 13$ binary elements. Some patterns of one-valued elements representing all texture feature values computed in all N texture windows are shown in the figure: the one-valued elements are depicted by black circles. Numbers placed at the right bottom corner of every cluster are coordinates of clusters (and respective texture windows).

4. Dilation of an arbitrary area in a binary matrix

The simplest version of the basic binary morphology operation used in the proposed algorithm is called dilation. The procedure of dilation of one-valued patterns represented by a binary matrix is often used in the texture segmentation process.

Let us consider a binary matrix $\mathbf{B}_0$ which contains an arbitrary number of one-valued elements. Let results of one step of dilation of the matrix $\mathbf{B}_0$ be recorded in a binary matrix $\mathbf{B}_1$ of the same size as $\mathbf{B}_0$. At first, the matrix $\mathbf{B}_0$ is copied into the matrix $\mathbf{B}_1$. Each one-valued pixel of the matrix $\mathbf{B}_0$, which does not belong to the matrix borders, has eight contiguous elements. It does not matter which of these surrounding elements are one-valued and which are zero-valued. One step of the procedure assigns one-value to all eight elements of the matrix $\mathbf{B}_1$ that surround every corresponding one-valued element of the matrix $\mathbf{B}_0$. Let us denote one step of the dilation procedure as Φ, so that $\mathbf{B}_1 = \Phi(\mathbf{B}_0)$. The second step of the dilation procedure produces the binary matrix $\mathbf{B}_2$ of the same size, $\mathbf{B}_2 = \Phi(\mathbf{B}_1)$. Let us designate the operation of sequential applying of H dilation steps to the binary matrix $\mathbf{B}$ by $\Phi_H(\mathbf{B})$. Also, let us designate the binary matrix obtained as a result of sequential application of these H steps to the initial binary matrix $\mathbf{B}_0$ by $\mathbf{B}_H$. Thus, $\mathbf{B}_H = \Phi_H(\mathbf{B}_0)$.

A series of sequential steps of dilation of the initial one-valued area gives rise to a certain increase in its size with an approximate retaining its shape.

Fig. 3 illustrates a single step dilation procedure. It shows dilation of an exemplary pattern of one-valued elements by a step of the procedure in the binary matrix of 20×20 elements. Black elements depict the pattern before dilation and grey elements show the appended one-valued elements after accomplishment of the procedure.

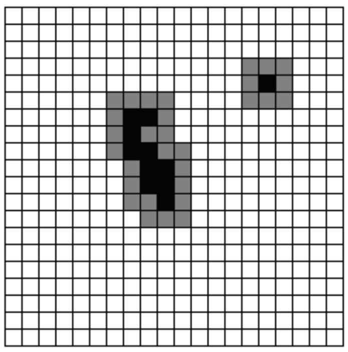

Fig. 3 *An example of a single step of dilation.*

5. Formation of exemplary feature pattern describing the sought-for texture segment

As mentioned above, the process of extraction of all homogeneous fine-grained texture segments is sequential and iterative. At each iteration (excepting the first one), the extraction algorithm processes not the whole image, but only a part of it. To represent the image region that is currently under processing, a binary matrix $\mathbf{Q}[i][j]$ of $N = I \times J$ elements is introduced. Let us underline that the same matrix $\mathbf{Q}$ is used in the whole process of extraction of all homogeneous fine-grained texture segments present in the input image. One-valued elements of the matrix $\mathbf{Q}$ indicate the pixels of the currently considered image region. At the beginning of the segmentation process (before the first iteration) all elements of the matrix $\mathbf{Q}$ are set one-valued.

In each n-th iteration, the algorithm proposed in [25] (and briefly described in Section 2 of this paper) determines a single seed pixel which is a representative of the most homogeneous fine-grained segment present in the currently considered image area marked by one-valued elements of the matrix $\mathbf{Q}$. Let us designate coordinates of this seed pixel as i^*, j^*. We postulate that the set of texture features extracted from the corresponding texture window characterizes this segment. This feature set is considered as the feature description of the sought-for texture segment. In order to get a more complete feature description of the segment, the (i^*, j^*)-th pixel is expanded into some square by means of H sequential dilation procedures exemplified in Fig. 3.

To describe this process, a binary matrix $\mathbf{P}[i][j]$ of $N = I \times J$ elements is introduced. At the beginning of the n-th iteration, all elements of the matrix $\mathbf{P}(n)$ are set zero-valued except for the (i^*, j^*)-th one which is set one-valued: $\mathbf{P}[i^*][j^*] = 1$. The procedure of H steps is applied to the matrix $\mathbf{P}$, which leads to the formation of some square area of one-valued elements: $\mathbf{P}^H = \Phi^H(\mathbf{P})$. Then, those one-valued elements of the area that do not belong to the currently processed domain $\mathbf{Q}$ are excluded from it. The remaining part of the square area $\mathbf{P}^H$ is designated as $\mathbf{P}^{\text{EXMPL}}$ and is named, hereinafter, an exemplary seed patch. The procedure of the exemplary seed patch restriction is described by the following:

$$\mathbf{P}^{\text{EXMPL}}[i][j] = \mathbf{P}^H[i][j] \wedge \mathbf{Q}[i][j], \tag{1}$$

where $i = 0, 1, 2, \ldots, I; j = 0, 1, 2, \ldots, J; \wedge$ is conjunction; $\mathbf{Q}$ is the binary matrix serving for representation (by its one-valued elements) of the currently considered image region.

So, the exemplary seed patch has, in a general case, more complex shape than a square. The exemplary seed patch is used to obtain more complete and representative feature description of the sought-for texture segment by means of combining all features extracted from all texture windows of the exemplary seed patch that immediately surround the (i^*, j^*)-th seed pixel.

An integer matrix $\mathbf{W}[m][k]$ of $M \times K$ elements is used to describe combination of all patterns extracted from the exemplary seed patch $\mathbf{P}^{\text{EXMPL}}$. In contrast to the array $\mathbf{G}[i][j][m][k]$, which is built only once for the whole segmentation process, the matrix $\mathbf{W}[m][k]$ is formed anew for each n-th iteration. The procedure of the formation of the matrix $\mathbf{W}$ is described in a (C++ like) pseudo-code in Algorithm 1.

Algorithm 1 Combining all feature patterns extracted from the exemplary seed patch $\mathbf{P}^{\text{EXMPL}}$.

```
Input: binary matrix $\mathbf{P}^{\text{EXMPL}}[I][J]$,   // Exemplary seed patch for the $n$-th
                                      // iteration.
  binary array $\mathbf{G}[I][J][M][K]$;           // Feature description of the whole image.
Output: integer matrix $\mathbf{W}[M][K]$;        // Exemplary feature pattern for the $n$-th
                                      // iteration.
  $\mathbf{W} = 0$;                              // Zeroing the matrix $\mathbf{W}$.
  for $(i = 0; i < I; i++)$                  // Cycle through $X$ coordinates of the
                                      // image.
    for $(j = 0; j < J; j++)$                // Cycle through $Y$ coordinates of the
                                      // image.
      if $(\mathbf{P}^{\text{EXMPL}}[i][j] == 1)$             // Testing:  if the pixel belongs to the
                                      // patch $\mathrm{P}^{\text{EXMPL}}$.
        for $(m = 0; m < M; m++)$            // Cycle through all features (all
                                      // vectors).
          for $(k = 0; k < K; k++)$          // Cycle through all elements of a vector.
            if $(\mathbf{G}[i][j][m][k] == 1)$         // Testing if there is a binary feature
                                      // value.
              $\mathbf{W}[m][k]++$;                 // Incrementing the element of the matrix.
            end if
          end for
        end for
      end if
    end for
  end for
```

So, due to the combining procedure, the exemplary feature pattern that is represented in the integer matrix $\mathbf{W}$ becomes the extended feature description of the sought-for texture segment. And each (m, k)-th element of the matrix $\mathbf{W}$ contains the number of binary elements of the array $\mathbf{G}$ (texture features presented in the array $\mathbf{G}$ in the binary thermometer format) that are found in all texture windows of the exemplary seed patch $\mathbf{P}^{\text{EXMPL}}$. Fig. 4 illustrates this description; it shows a demo sample of the exemplary feature pattern which is depicted as consisting of $M = 5$ vectors of $K = 13$ integer elements each. In this figure, the numbers of corresponding texture features (one-valued elements of the corresponding vectors) collected from all over the exemplary seed patch in the matrix $\mathbf{W}$ are depicted by the intensity of black color: white circle indicates that this feature value is absent in the exemplary seed patch. It should be apparent that owing to the thermometry coding of all texture feature values, the elements at the bottom of the vectors (columns) of the matrix $\mathbf{W}$ will always have larger values than at the top of these vectors (columns).

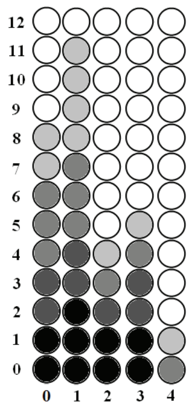

Fig. 4 *A demo sample of the exemplary feature pattern distributed in the integer matrix* $\mathbf{W}[M][K]$.

6. Comparisons between the exemplary pattern and feature patterns of all texture windows

As described above, the process of extraction of all homogeneous fine-grained texture segments is sequential and iterative, one segment per iteration. In the n-th iteration, the segmentation process starts from comparisons between the exemplary feature pattern $\mathbf{W}[m][k]$ (being an integer matrix) and each feature pattern represented in a binary feature cluster (being a structural part of the array $\mathbf{G}$) of the currently considered image region $\mathbf{Q}$. Note that we consider the four-dimensional array $\mathbf{G}[I][J][M][K]$ as $N = I \times J$ binary feature clusters (see Fig. 2); and every (i, j)-th feature cluster comprises M binary vectors of K elements each.

An integer matrix $\mathbf{E}[i][j]$ of $N = I \times J$ elements is introduced to represent results of the comparisons; the matrix is set zero-valued at the beginning of each iteration. The procedure of the matrix $\mathbf{E}$ formation is described in a (C++ like) pseudo-code in Algorithm 2.

As follows from Algorithm 2, within the procedure of the matrix $\mathbf{E}$ formation a number of comparisons is sequentially performed. Each comparison evaluates similarity between two feature patterns. These patterns are: the exemplary feature pattern $\mathbf{W}$ and the currently considered (i, j)-th feature pattern distributed in the binary feature cluster $\mathbf{G}[i][j]$. Actually, every comparison evaluates a measure of texture similarity between the exemplary seed patch $\mathbf{P}^{\text{EXMPL}}$ and the (i, j)-th texture window. The evaluated value of similarity is represented in the (i, j) element of the matrix $\mathbf{E}$. So, as a result of the matrix $\mathbf{E}$ formation all over the image, every (i, j)-th element of the matrix $\mathbf{E}$ reflects the degree of similarity between the (i, j)-th texture window and the exemplary seed patch $\mathbf{P}^{\text{EXMPL}}$. Therefore, those elements of the matrix $\mathbf{E}$ that correspond to the sought-for texture segment will have much higher values than the elements representing other image regions. We take advantage of that in order to extract the areas of texture segments as described below.

Algorithm 2 Formation of the matrix **E**.

```
Input: binary matrix $\mathbf{Q}[I][J]$,          // Currently considered image region.
  binary array $\mathbf{G}[I][J][M][K]$,           // Feature description of the whole image.
  integer matrix $\mathbf{W}[M][K]$;               // Exemplary feature pattern.
Output: integer matrix $\mathbf{E}[I][J]$;         // Degree of similarity between feature
                                                   // patterns of all feature clusters and
                                                   // the exemplary feature pattern of the
                                                   // currently extracted texture segment.
  $\mathbf{E} = 0$;                                // Zeroing the matrix $\mathbf{E}$.
  for $(i = 0; i < I; i++)$                        // Cycle through $X$ coordinates of the
                                                   // image.
    for $(j = 0; j < J; j++)$                      // Cycle through $Y$ coordinates of the
                                                   // image.
      if $(\mathbf{Q}[i][j] == 1)$                 // Testing: if the pixel belongs to the
                                                   // currently considered image region.
        for $(m = 0; m < M; m++)$                  // Cycle through all features.
          for $(k = 0; k < K; k++)$                // Cycle through all elements representing
                                                   // a feature in the array $\mathbf{G}$ and through
                                                   // all elements of vectors within the
                                                   // integer matrix $\mathbf{W}$ (see Figs. 2 and 4).
            if $((\mathbf{G}[i][j][m][k] == 1) \&\& (\mathbf{W}[m][k] > 0))$
              $\mathbf{E}[i][j] = \mathbf{E}[i][j] + \mathbf{W}[m][k]$;
            else if $((\mathbf{G}[i][j][m][k] == 1) \&\& (\mathbf{W}[m][k] == 0))$
              $\mathbf{E}[i][j] = \mathbf{E}[i][j] - 1$;
            else if $((\mathbf{G}[i][j][m][k] == 0) \&\& (\mathbf{W}[m][k] > 0))$
              $\mathbf{E}[i][j] = \mathbf{E}[i][j] - \mathbf{W}[m][k]$;
            end if
          end for
        end for
      end if
    end for
  end for
```

Figs. 5 and 6 illustrate this description. Fig. 5A is the same as Fig. 4; it shows the exemplary feature pattern fixed in the matrix **W** which characterizes the currently extracted texture segment. Fig. 5B demonstrates the feature pattern extracted from any texture window that belongs to the sought-for texture segment; the pattern is represented in the corresponding binary feature cluster **G**. In these figures, the matrix **W** and the feature cluster are depicted as consisting of $M = 5$ vectors of $K = 13$ elements each. As seen in Fig. 5, both feature patterns are very similar.

Fig. 6A is the same as Fig. 4 and Fig. 5A; it shows the exemplary feature pattern fixed in the matrix **W** which characterizes the currently extracted texture segment. Fig. 6B demonstrates the feature pattern extracted from any texture window which does not belong to the sought-for texture segment; the pattern is fixed in the corresponding feature cluster **G**. (Again, the matrix **W** and the feature cluster are depicted as consisting of $M = 5$ vectors of $K = 13$ elements each.) As seen in Fig. 6, both feature patterns are quite dissimilar.

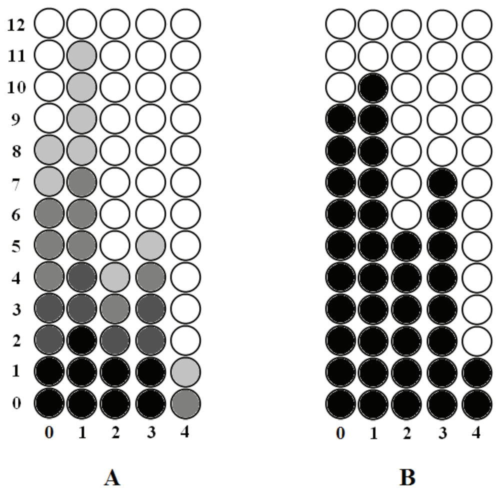

Fig. 5 ***A*** – *A demo sample of the exemplary feature pattern fixed in the integer matrix* $\mathbf{W}[M][K]$.
B – *A demo sample of one feature pattern extracted from any texture window belonging to the sought-for texture segment; the pattern is represented in the corresponding binary feature cluster (the array* $\mathbf{G}$*).*

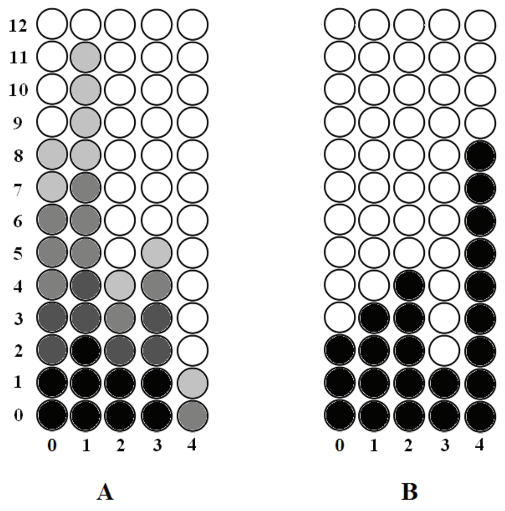

Fig. 6 ***A*** – *A demo sample of the exemplary feature pattern fixed in the integer matrix* $\mathbf{W}[M][K]$.
B – *A demo sample of one feature pattern extracted from any texture window that does not belong to the sought-for texture segment; the pattern is represented in the corresponding binary feature cluster (the array* $\mathbf{G}$*).*

Fig. 7 shows a demonstration experiment with a real texture image. The figure illustrates the first iteration of the segmentation procedure. The left part of Fig. 7 is the input image. The texture segment extracted by the segmentation algorithm in the first iteration is the park road. Location of the exemplary seed patch $\mathbf{P}^{\mathrm{EXMPL}}$, which has been found by the algorithm described in [25], is indicated by a black square in the left part of the figure. The left part of the figure presents the matrix $\mathbf{E}$ computed according to Algorithm 2 at the first iteration of the segmentation process. Thus, in this iteration, the matrix $\mathbf{E}$ is computed specially for the extraction of the park road texture segment in the image. In the figure, the values of the matrix elements are expressed by the intensity of white. As seen in the right part of Fig. 7, the elements of the matrix $\mathbf{E}$ corresponding to the park road texture segment have much higher values than all other its elements.

We consider the value of each element of the matrix $\mathbf{E}$ (corresponding to the image pixel) as a measure of its belonging to the sought-for texture segment. This value is used by the segmentation algorithm to make a decision about inclusion (or exclusion) of the image pixel into the segment.

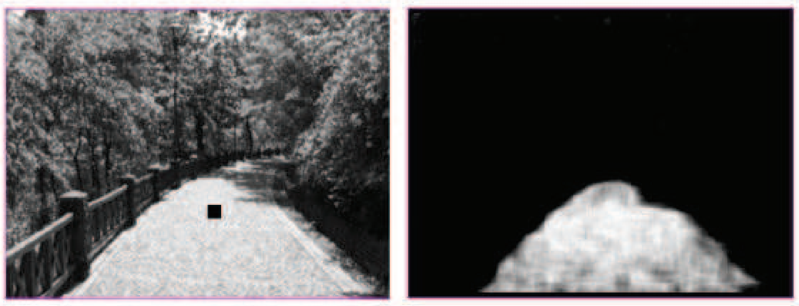

Fig. 7 *A landscape photograph of a park (the left part) and the matrix* $\boldsymbol{E}$ *computed at the first iteration of the segmentation process (the right part).*

7. An overview of the algorithm for extraction of one homogeneous fine-grained texture segment

This section contains a short overview of the algorithm extracting one homogeneous fine-grained texture segment (full description of the algorithm is given in Appendix A). As mentioned above, one homogeneous and (usually) simply connected texture segment is extracted in each iteration. The extraction process starts from finding the representative seed pixel and corresponding set of texture features that characterize the most homogeneous fine-grained texture segment present in the currently considered image region $\mathbf{Q}$. The set of texture features characterizing the most homogeneous texture segment is presented in the exemplary seed patch $\mathbf{P}^{\mathrm{EXMPL}}$.

Let us introduce a binary matrix $\mathbf{R}[i][j]$ of $N = I \times J$ size to represent pixels of the n-th texture segment extracted by the algorithm in the n-th iteration. At the beginning of the n-th iteration, all elements of the matrix $\mathbf{R}$ are set zero-valued.

And during the extraction process, those pixels that are found belonging to the n-th segment will be represented by one-valued elements of the matrix $\mathbf{R}$.

At the beginning of extraction of the n-th texture segment, the exemplary seed patch $\mathbf{P}^{\mathrm{EXMPL}}$ is used as an origin from which the extracted segment is gradually expanded to its boundaries. So, at the beginning of the extraction process, the exemplary seed patch $\mathbf{P}^{\mathrm{EXMPL}}$ is rewritten into the matrix $\mathbf{R} : \mathbf{R}[i][j] = \mathbf{P}^{\mathrm{EXMPL}}[i][j]$.

The extraction process consists of a number of expansion steps. Each expansion step starts from the dilation procedure described in Section 4. The expansion steps are combined in series. In every iteration, the number of series of expansion steps is limited.

In every expansion step, at first, the outer contour is found which circumscribes the area of the n-th segment that has been determined by this moment. Then, the values of those elements of the matrix $\mathbf{E}$ that correspond to the contour pixels are compared to some variable threshold $L(t)$. This is done in order to find out whether these pixels have enough similar texture characteristics to those of the exemplary seed patch $\mathbf{P}^{\mathrm{EXMPL}}$. In other words, the comparison between the value $\mathbf{E}[i][j]$ and the threshold $L(t)$ is used in order to make a decision whether the corresponding pixel belongs to the sought-for n-th segment or not. If the (i, j)-th contour pixel under test has enough similar texture characteristics, it is appended to the previously defined pixels of the n-th segment (that is expressed by assigning unit value to the corresponding (i, j)-th element of the matrix $\mathbf{R}$. If the contour element $\mathbf{E}[i][j]$ is less than $L(t)$, the corresponding (i, j)-th element of the matrix $\mathbf{R}$ remains zero-valued.

Such a sequential expansion of the sought-for n-th segment continues during T series of expansion steps. For the first series (at $t = 0$) the threshold value L is equal to $E^{\max}$. For each new expansion series, the threshold $L(t)$ decreases slightly. So, during the whole expansion procedure, the threshold value L decreases from $L^{\max} = E^{\max}$ to $L^{\min}$ (for the last series, at $t = T$). Actually, in the experiments with the computer model of the texture segmentation algorithm, the threshold value L decreases from $E^{\max}$ to $E^{\max}/2$.

At the end of the expansion process, the whole area of the segment becomes marked by one-valued elements of the matrix $\mathbf{R}$ meaning that one approximately homogeneous and (usually) simply connected fine-grained texture segment is delineated in the image.

When the process of extraction of the n-th texture segment is completed, its pixels are excluded from the subsequent consideration by subtracting them from the currently considered image region $\mathbf{Q}$, that is:

$$\mathbf{Q}[i][j] = \mathbf{Q}[i][j] \oplus \mathbf{R}[i][j], \tag{2}$$

where $i = 0, 1, 2, \ldots, I; j = 0, 1, 2, \ldots, J; \oplus$ is exclusive disjunction (XOR).

It is evident that due to this procedure, each subsequent segment will not intersect with the segments that were extracted earlier.

Extraction of the next $(n + 1)$-th texture segment is performed according to the above description. As always, the extraction process starts with finding the representative seed pixel and a set of texture features characterizing the most homo geneous fine-grained texture segment which is present in the remained unprocessed image region that is indicated by one-valued elements of the matrix **Q**.

Together with the set of characterizing texture features, the algorithm described in [25] evaluates the degree of homogeneity of the $(n+1)$-th texture segment and outputs it as a value of some parameter Ω. Let us introduce another parameter – a threshold F which may be interpreted as a threshold for the segments that could be called homogeneous. The threshold F may be used to terminate the segmentation procedure when Ω obtained for the next $(n+1)$-th texture segment becomes less than F. And we can be sure that by this moment all noticeable homogeneous fine-grained texture segments have already been extracted in the image.

8. Experiments

The computer program simulating the texture segmentation algorithm has the following basic parameters. Experiments deal with black and white images of 427×320 pixels each. Brightness values of original image pixels range from 0 to 255. Texture window of 15×15 pixels is used. A total of $M = 23$ texture features are extracted from every texture window and are represented in the feature cluster containing $M = 23$ (column) vectors of $K = 20$ binary elements each.

The program has been tested on natural scenes of different types. Examples of texture segmentation results are presented in Figs. 8–17. Every figure consists of two parts (left and right) that are in mutual coordinate correspondence. Input image is placed in both parts of the figures. Texture segments extracted by the algorithm are shown in the right part of each figure by different colors. No more than eight most homogeneous fine-grained texture segments are extracted in the experiments. In all figures, the segment which is extracted first is painted in green. The subsequent segments are painted in the following order of colors: red, dark-blue, violaceous, yellow, light-blue, brown, pink.

The results of texture segmentation shown in Figs. 8–17 are obtained with the same algorithm's parameters without tuning them for each photograph individually.

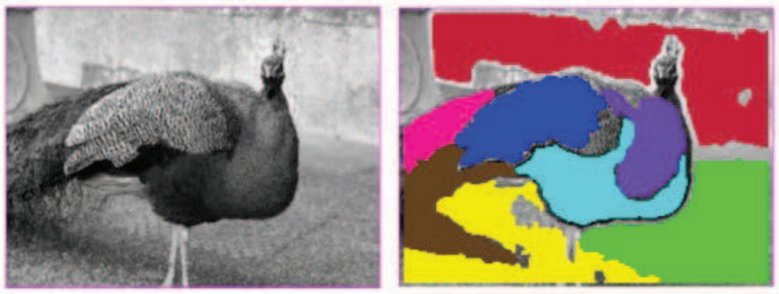

Fig. 8 *A scene with a peacock (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

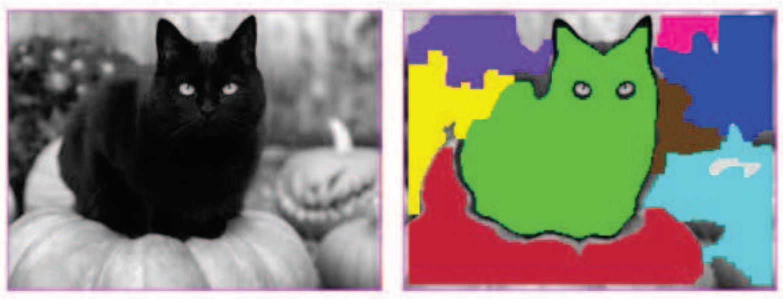

Fig. 9 *A scene with a cat (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

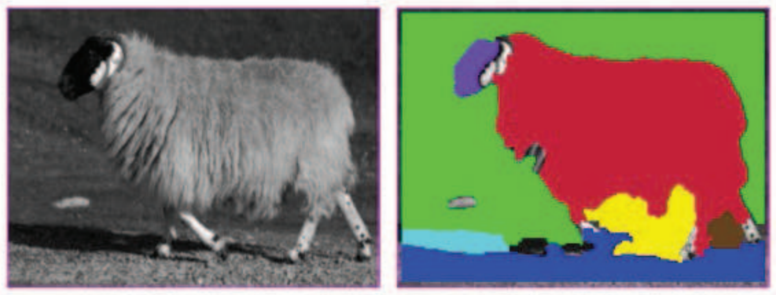

Fig. 10 *A scene with a sheep (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown.*

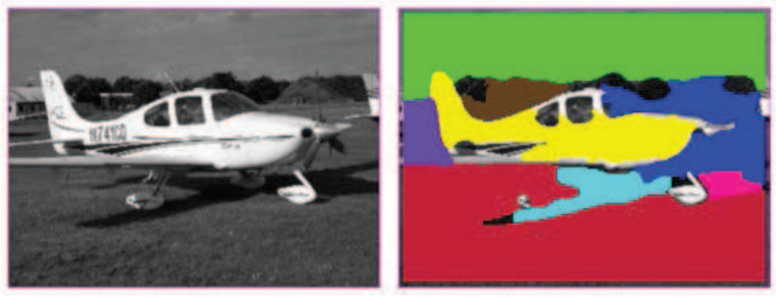

Fig. 11 *A scene with an airplane (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

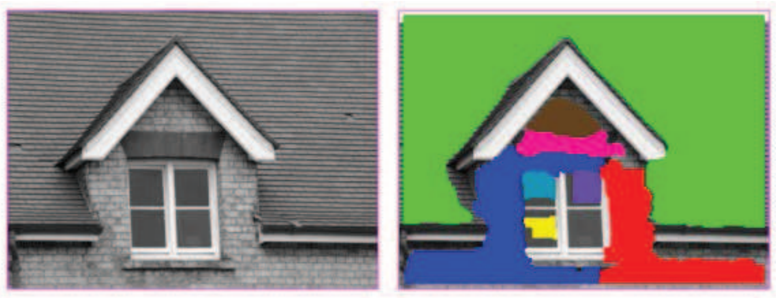

Fig. 12 *A photograph of a roof of a house (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

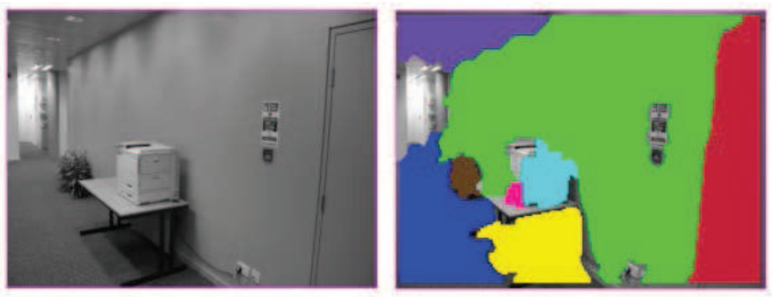

Fig. 13 *A photograph of an office (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

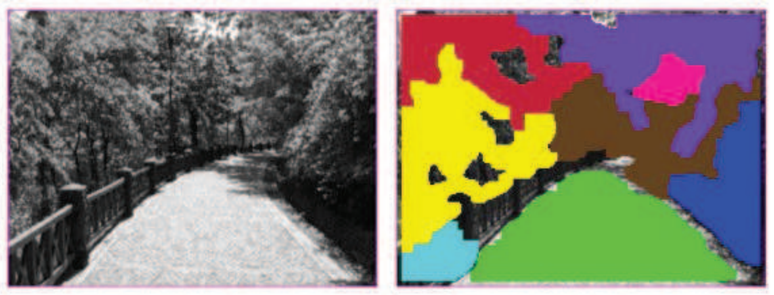

Fig. 14 *A landscape photograph of a park (the left part) that is presented in Fig. 7 and results of extraction of 8 the most homogeneous fine-grained texture segments in it (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

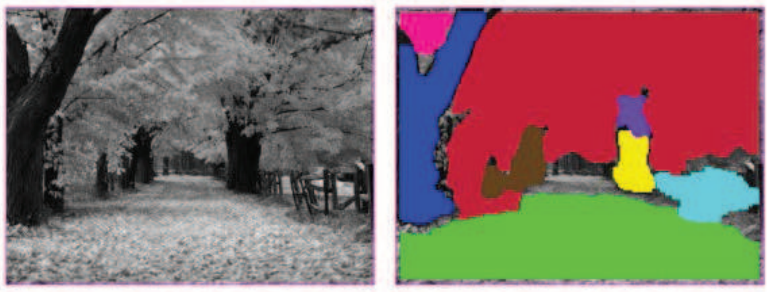

Fig. 15 *A landscape photograph (autumn) (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue, brown, pink.*

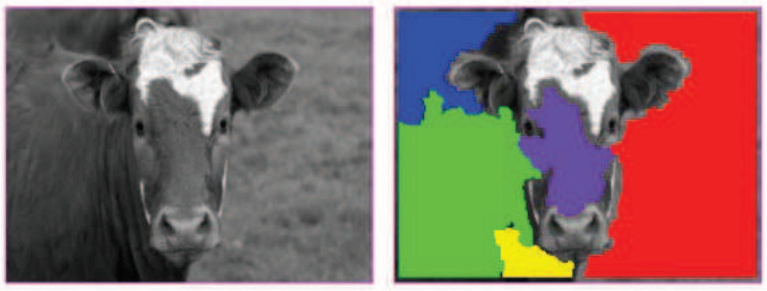

Fig. 16 *A photograph of a cow (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow.*

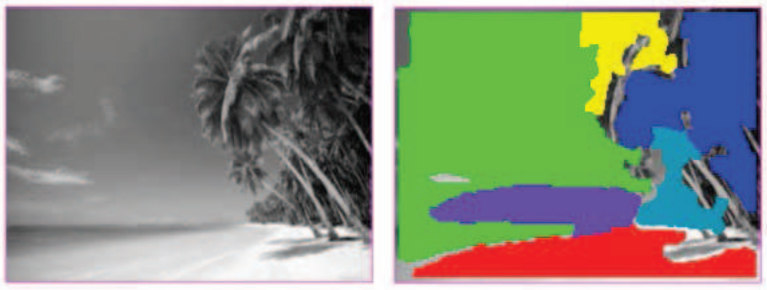

Fig. 17 *A seashore photograph (the left part) and results of its texture segmentation (the right part). The colors of the segments that have been consecutively extracted in this image by the algorithm are in the following order: green, red, dark-blue, violaceous, yellow, light-blue.*

Fig. 17 is included into the set of experiments with the intention to demonstrate restrictions of the algorithm. As seen in the figure, the algorithm cannot catch what is the vague border between the sky and the sea texture segments. Instead, the sky segment (green color) occupies a part of the shore segment (red color), which is quite incorrect "from a human point of view". However, objectively, the textures of the sky, the sea and the shore segments within the considered image region are the same, which leads to the obtained segmentation result.

As seen in Figs. 8–17, the larger the texture segment, the more correctly it is extracted. And, the smaller the texture segment, the worse it is delineated by this algorithm. The reason for this result is evident, it is a rather large size of the texture window (15×15 pixels) compared to the relatively small size of the image (427×320 pixels).

9. Discussion

The proposed segmentation algorithm has the following inherent peculiarity. Extraction of every texture segment is performed in turn, separately from all others. And this also concerns the segments of the same texture class. Due to this peculiarity, separated texture segments of the same texture class will not be attributed to one class by the algorithm. This is illustrated by Fig. 15 in which the segments of the same texture of tree trunks are painted by different colors. This peculiarity leads to the necessity to perform an additional step of the segmentation procedure. That is, at the final stage of the segmentation procedure, the averaged texture characteristics of all extracted texture segments should be compared to each other in order to identify those segments that have similar characteristics and, therefore, belong to the same texture class. This peculiarity is, evidently, a consequence of the fact that the proposed segmentation algorithm belongs to a region growing approach which is a local method with no global view of the problem.

Another peculiarity of the proposed segmentation algorithm is that it reliably provides correct extraction of any homogeneous fine-grained texture segment if it is surrounded from all sides by contrasting areas (for example, by other texture segments with dissimilar texture characteristics). However, if the considered segment is bordered with another segment with similar texture characteristics, the algorithm may merge them into one area.

Different parameters of the segmentation algorithm should be chosen to extract somewhat more or somewhat less homogeneous fine-grained texture segments. In particular, if it is necessary to analyze the texture configuration of the input image in a greater detail, the parameters could be chosen that provide extraction of a larger number of smaller but more homogeneous segments.

The number and shapes of the homogeneous fine-grained texture segments extracted by the algorithm in the input image depend on many circumstances. One of them is the segmentation algorithm's parameters. Another circumstance, which largely defines the number and shapes of the extracted texture segments, is an internal texture structure of the image. This means that if the input image consists of homogeneous fine-grained texture segments with rather different texture characteristics, the segmentation algorithm reliably extracts the same segments, even if the algorithm's parameters vary in a wide range. Vice versa, if any texture segment in the image is bordered with another segment with similar texture characteristics, the algorithm may unite them into one area, as it is demonstrated in Fig. 17.

Also, a very important parameter, which strongly influences the segmentation process, is the texture window size (in relation to the image size). The larger the window size, the less homogeneous and coarser texture segments are extracted by the algorithm. However, a too small texture window does not provide adequate descriptions of textures.

Surely it would be interesting to explore the functionality of this algorithm in relation to the mechanism of vision in humans or other vision-equipped organisms. It is well known that the primary visual cortex of the mammalian brain plays a key role in visual perception because it receives and processes visual information. So, functionally, the primary visual cortex is used for analysis of visual images. The organization of sensory maps in the cortex reflects that of the retina. Peculiarity of the architectural organization of the primary visual cortex is that it contains a huge number of neural columns (e.g. [7, 13, 26, 27, 35]. This columnar organization has the local connectivity, that is connections "up" and "down" are much denser than connections that spread from side to side (between the neighboring columns). The system of feature clusters that consists of a number of binary (column) vectors is, evidently, similar to such a neural columnar architecture. The proposed segmentation algorithm can be, definitely, implemented in a neural network with a column architecture (like the visual cortex) by means of not too complex interactions between neurons of the same horizontal levels within a number of neighboring neural columns through local horizontal neural connections. Of course, all the above reasoning does not mean at all that we are claiming that the same segmentation algorithm is actually used in living organisms. There can be no doubt that in order to establish a link between all these architectural and functional similarities, deep and systematic research would be needed, which is beyond the scope of this research.

The experimental results presented in this paper are applicable only for qualitative assessment of the effectiveness of the segmentation algorithm in such terms as correct (or incorrect) extraction of segments "from a human point of view". There are also the following reasons for this.

The proposed algorithm is intended to extract only homogeneous fine-grained texture segments and it cannot extract a coarse texture segment as a whole region. As mentioned in Section 1, the algorithm divides every coarse texture region into a number of homogeneous fine-grained texture segments. Fig. 16 illustrates this statement. This figure is a black and white version of the 1_22_s image of the MSRC Database. As seen in Fig. 16, the algorithm has found four comparatively large homogeneous texture segments in the horse's area. And the coarse part the horse's area has not been segmented at all. However, the ground through file which accompanies the 1_22_s image in the database contains the "horse" segment (the "horse" ground through) which is the whole horse's region including its body and its head, i.e. including all four extracted homogeneous texture segments and the unextracted part of coarse texture. In most databases used for texture and object segmentation and recognition, the image region occupied by any object is considered as a region of the same texture and is supplied with its ground through.

Therefore, it should be evident that any quantitative evaluation of the proposed segmentation algorithm cannot be performed on such databases.

It is worth to note that humans always use some high-level features for image segmentation, such as the context of the analyzed scene and the knowledge of the considered segment belonging to a certain object which the human has previously recognized in the image. That is why the presence in Figs. 8–17 of segments that are "not correct" from the human point of view, may be explained by the use of man's high-level knowledge and somewhat different sets of texture features used by humans and by the algorithm.

Elaboration of the present segmentation algorithm continues. The texture feature set used in the experimental part of this paper is rather simple. It would be useful to test other features for the present algorithm. For example, LBR (Local Binary Pattern) and uniform LBR operators are now popular for the solution of such tasks as texture classification, face recognition, facial expression recognition and so on (e.g. [43,57]). These operators are used to create an LBR histogram that could be applied as texture features instead of the present brightness histogram.

10. Conclusions

The purpose of this research was to propose an algorithm for partial texture segmentation of visual images. The algorithm segments any input image into a number of non-overlapping homogeneous fine-grained texture areas.

The main advantages of the proposed algorithm are as follows. The algorithm is fully unsupervised, that is, it processes the input image without any a priori knowledge, neither of the type of textures, nor of the number of texture segments in the image. Furthermore, the algorithm sequentially segments arbitrary images of all types. That means no changes of the algorithm's parameters are required for going from one type of image to another. For example, the texture segmentation results shown in Figs. 8–17 are not obtained by means of tuning the algorithm's parameters for each photograph individually, but all these images have been processed with the same parameters.

Another important advantage of the algorithm is that, in most cases, the algorithm extracts homogeneous fine-grained texture segments present in the image in a similar way as humans do. This result is confirmed by a series of experiments that demonstrate the ability of the algorithm to perform delineation of homogeneous fine-grained texture segments in a wide range of images. Of course, not all images are segmented successfully "from a human point of view" (e.g. see Fig. 16).

In addition to this, an image processing by the proposed algorithm results in considerable reduction of the uncertainty of internal structure of the analyzed image. In a typical case, the input image contains a number of homogeneous fine-grained texture segments. And all of them will be extracted by the algorithm. For example, 60–90 percent of the image area might be marked as different homogeneous fine-grained segments, while the remaining 10–40 percent of the image area would consist of coarse textures, boundary and non-texture patches. Therefore, the algorithm can provide helpful additional information about the image that might significantly facilitate solution of such tasks as segmentation, analysis, object-ground separation and, finally, recognition of all objects present in the image.

Appendix A The algorithm of extraction of one homogeneous fine-grained texture segment

As mentioned in Section 7, the binary matrix $\mathbf{R}[i][j]$ of $I \times J$ size is introduced to represent (by its one-valued elements) the n-th texture segment extracted by the algorithm in the n-th iteration. The extraction process is divided into a large number of expansion steps. The expansion steps are combined in series. In every iteration, the number of series of expansion steps is limited; let us designate this limit by T. The series number is indicated by t and the expansion step number (within the current series t) is indicated by s.

Let us consider the t-th series of expansion steps within the process of extraction of the n-th texture segment. For the description of the process we introduce a binary matrix $\mathbf{Z}$ of $N = I \times J$ elements. The pixels belonging to the n-th segment are marked by one-valued elements of the matrix $\mathbf{Z}$. It is necessary to underline that the matrix $\mathbf{Z}$ is introduced to represent only those pixels of the n-th segment that are extracted by the algorithm during the t-th series of expansion steps. In order to express explicitly configurations of one-valued elements representing pixels of the extracted segment in the matrix $\mathbf{Z}$, we use a notation $\mathbf{Z}(t,s)[i][j]$ which designates the matrix $\mathbf{Z}$ during execution of the s-th expansion step of the t-th series.

At the beginning of each t-th series of expansion steps ($s = 0$), the matrix $\mathbf{Z}$ is initialized by the matrix $\mathbf{R}$:

$$\mathbf{Z}(t,0)[i][j] = \mathbf{R}[i][j]. \tag{3}$$

Each expansion step s begins with finding the outer contour which circumscribes the segment that has been extracted by this step and is presented in the matrix $\mathbf{Z}(t,s)$ (by its one-valued elements). Let us introduce an auxiliary binary matrix $\mathbf{A}[i][j]$ of $N = I \times J$ elements for representation of the contour. At every s-th expansion step of the t-th series, the contour $\mathbf{A}$ is computed by the formula:

$$\mathbf{A}[i][j] = \mathbf{Q}[i][j] \wedge (\Phi^1(\mathbf{Z}(t,s)[i][j]) \oplus \mathbf{Z}(t,s)[i][j]), \tag{4}$$

where $i = 0,1,2,\ldots,I; j = 0,1,2,\ldots,J; \wedge$ is conjunction; $\oplus$ is exclusive disjunction (XOR); Φ is one step of the dilation procedure in a binary matrix described in Section 4; $\mathbf{Q}$ is the currently considered image region $\mathbf{Q}$. After the outer contour $\mathbf{A}$ has been determined, the integer values of all elements of the matrix $\mathbf{E}$ corresponding to the contour pixels (one-valued pixels of the matrix $\mathbf{A}$) are compared with the variable threshold $L(t)$. At the s-th expansion step of the t-th series, the pixels corresponding to the contour elements of the matrix $\mathbf{E}$ whose values exceed the threshold $L(t)$ are appended to the configuration of one-valued pixels fixed in the matrix $\mathbf{Z}$ earlier. This is expressed by the following formula:

$$\mathbf{Z}(t,s)[i][j] = \mathbf{Z}(t,s)[i][j] \vee (\mathbf{l}(\mathbf{E}[i][j] - L(t)) \wedge \mathbf{A}[i][j]), \tag{5}$$

where $i = 0,1,2,\ldots,I; j = 0,1,2,\ldots,J$; the designation $\mathbf{l}$ is the unit step function which is defined by the formula:

$$\mathbf{l}(k) = \begin{cases} 1, & \text{for } k > 0, \\ 0, & \text{for } k \le 0. \end{cases}$$

The threshold value L is constant in all expansion steps within the same series, but it decreases from series to series, as follows. For every t-th series of expansion steps, the threshold value $L(t)$ is calculated by the equation: $L(t) = E^{\max} - UtE^{\max}$, where $E^{\max}$ is the maximal value of the matrix $\mathbf{E}$ computed before the n-th iteration and U is some coefficient, $U < 1$. Eq. (5) means that the threshold value L is directly proportional to the maximal value of the matrix $\mathbf{E}$. During T series of expansion steps, the threshold value $L(t)$ decreases from the maximal level $L^{\max} = E^{\max}$ (for the first series, when $t = 0$) to the minimal value $L^{\min}$ (for the last series, when $t = T$). In the experiments with the computer model of the texture segmentation algorithm, the number of series of expansion steps T is 70 and the coefficient U is 0.7. Such values of these two parameters mean that in the process of extraction of the n-th segment the threshold L decreases from the maximal value $E^{\max}$ of the current matrix $\mathbf{E}$ to its minimal level $L^{\min}$, which is 49 % of $E^{\max}$.

Thus, Eq. (5) expresses that in each s-th expansion step of the t-th series the area of the extracted segment (contained in the matrix $\mathbf{Z}$) is sequentially expanded by means of appending those surrounding pixels whose corresponding elements of the matrix $\mathbf{E}$ have enough high values. This process is accompanied by counting the number of appended pixels in each s-th expansion step. Every t-th series continues as long as the extracted segment's area increases. If the area stops to grow, i.e., if the number of appended pixels in the s-th expansion step becomes equal to zero, then the current t-th series is terminated.

Let us designate the final configuration of one-valued pixels of the matrix $\mathbf{Z}$ at the t-th series as $\mathbf{Z}^{\text{fin}}(t)$. When the current t-th series is terminated, the area of one-valued elements fixed in the matrix $\mathbf{Z}^{\text{fin}}(t)$ is analyzed in order to find out whether it is a simply connected area or whether it consists of some number of separate small spots. If the area consists of small spots, all pixels marked by one-valued elements of the matrix $\mathbf{Z}^{\text{fin}}(t)$ are appended to the matrix $\mathbf{R}$, as expressed by the formula:

$$\mathbf{R}[i][j] = \mathbf{R}[i][j] \vee \mathbf{Z}^{\text{fin}}(t)[i][j], \tag{6}$$

where $i = 0, 1, 2, \ldots, I; j = 0, 1, 2, \ldots, J; \vee$ is disjunction.

And in this case (of separate small spots), the procedure of Eq. (6) becomes final in the t-th series of expansion steps. Then, the process of extraction of the n-th texture segment continues in the $(t+1)$-th series of expansion steps in full compliance with the above description starting from Eq. (3).

However, if it is found out that the area of one-valued elements obtained in the matrix $\mathbf{Z}^{\text{fin}}(t)$ constitutes a rather large simply connected region, its averaged texture characteristics are computed and compared with the corresponding averaged texture characteristics of the area fixed in the matrix $\mathbf{R}$. For such a comparison, four pairs of texture characteristics are used that are averaged over all one-valued pixels of both matrices $\mathbf{Z}^{\text{fin}}(t)$ and $\mathbf{R}$. The first pair comprises the mean values of the brightness histograms. The second pair of texture characteristics contains the mean values of the orientation histograms. The third one includes the mean values of brightness non-uniformity characteristics. And the fourth pair encompasses the mean values of brightness of all pixels.

To evaluate texture similarity of the areas $\mathbf{Z}^{\text{fin}}(t)$ and $\mathbf{R}$, absolute differences within all four pairs of the averaged texture characteristics are calculated. If all four differences lie inside some predefined thresholds, then the one-valued area obtained in the matrix $\mathbf{Z}^{\text{fin}}(t)$ is appended to the area fixed in the matrix $\mathbf{R}$ according to Eq. (6). And, then, the process of extraction of the n-th texture segment continues in the $(t+1)$-th series of expansion steps as mentioned above. However, if any difference exceeds the corresponding threshold, one-valued elements of the array $\mathbf{Z}^{\text{fin}}(t)$ are not appended to the matrix $\mathbf{R}$. The grounds for non-inclusion of the simply connected area obtained in the matrix $\mathbf{Z}^{\text{fin}}(t)$ in the sought-for texture segment are evident: all parts of the homogeneous fine-grained segment should have similar averaged texture characteristics. Then, non-included pixels of the area $\mathbf{Z}^{\text{fin}}(t)$ are deleted from the segment's area fixed in the matrix $\mathbf{R}$, that is:

$$\mathbf{R}[i][j] = \mathbf{R}[i][j] \oplus \mathbf{Z}^{\text{fin}}(t)[i][j], \tag{7}$$

where $i = 0, 1, 2, \ldots, I; j = 0, 1, 2, \ldots, J; \oplus$ is exclusive disjunction (XOR).

Let us note that in the absence of the procedure described by Eq. (7) the final texture segment represented by one-valued pixels in the matrix $\mathbf{R}$ must always be simply connected. However, the procedure of Eq. (7) sometimes divides the extracted simply connected texture segment into several disconnected parts.

As mentioned in Section 7, when the process of extraction of the n-th texture segment is completed, its pixels are excluded from the subsequent consideration by means of subtracting them from the currently considered image region $\mathbf{Q}$ that is expressed by Eq. (2).

As mentioned in Section 7, extraction of the next $(n+1)$-th texture segment is performed in full accordance with the above description and Eqs. (2–(7). As always, the extraction process starts with finding the representative seed pixel and a set of texture features characterizing the most homogeneous fine-grained texture segment present in the remained image domain $\mathbf{Q}$.

Appendix B Notations

For the convenience of the reader, the terminology used in this paper is listed here. Of course, it is explained in detail in the paper itself.

Texture window is a rather small square-shaped area of the image (square-shaped image patch). A set of such texture windows covers the whole image. Neighboring texture windows have intersections between one another. Texture window size is of 15×15 pixels.

Texture feature is some characteristic of a texture. Each texture feature is extracted from a texture window and serves to estimate texture similarity and/or difference between different sites of the image. A total of $M = 23$ texture features are extracted from every texture window.

Feature cluster is a set of all texture features extracted from a single texture window. This set of features is represented in a complex of binary (column) vectors. Each feature is represented in its own vector. Each feature cluster contains $M = 23$ (column) vectors of $K = 20$ binary elements each.

Seed point is the image pixel which definitely belongs to the currently extracted texture segment. This pixel is found before the start of the extraction procedure for each texture segment.

Exemplary seed patch is the image area (patch) which is considered representative of the currently extracted texture segment. All pixels of the seed patch are located around the corresponding seed point.

Exemplary feature pattern is a set of all texture features which is extracted from the exemplary seed patch and is considered as a complete feature description of the sought-for texture segment. Exemplary feature pattern is represented in the feature cluster containing $M = 23$ (column) vectors of $K = 20$ gradual elements each.

Image size: $I = 427; J = 320$.

$\mathbf{Q}[i][j]$ is the binary matrix of $N = I \times J$ elements which is used to represent (by its one-valued elements) the image region that is currently under processing.

$\mathbf{G}[\boldsymbol{i}][\boldsymbol{j}][\boldsymbol{m}][\boldsymbol{k}]$ is the binary four-dimensional array of $I \times J \times M \times K$ elements which serves for a formal feature description of the whole input image. Actually, the array $\mathbf{G}$ is composed of $N = I \times J$ binary feature clusters.

$\mathbf{P}^{\mathrm{EXMPL}}[i][j]$ is the exemplary seed patch of $I \times J$ binary elements. It serves for representation (by its one-valued elements) of the exemplary texture windows for the currently extracted segment.

$\mathbf{W}[m][k]$ is the integer matrix of $M \times K$ elements which is used to integrate combination of all feature patterns extracted from the exemplary seed patch $\mathbf{P}^{\mathrm{EXMPL}}$.

$\mathbf{E}[i][j]$ is the integer matrix of $N = I \times J$ elements which represents results of all comparisons between each texture window under consideration and the integrated feature description of the exemplary seed patch $\mathbf{W}[m][k]$.

$\mathbf{R}[i][j]$ is the binary matrix of $N = I \times J$ size to represent pixels of the currently extracted texture segment.

$T = 70$. Parameter T is the number of series of expansion steps in the procedure of extraction of each texture segment. Correspondingly, the coefficient U is 0.7. These values of parameters mean that the threshold L decreases from its maximal value $E^{\max}$ of the current matrix $\mathbf{E}$ to its minimal level $L^{\min}$, which is 49 % of $E^{\max}$.

Acknowledgement

This work was supported in part by Institute of Computer Science, The Czech Academy of Sciences.

References

[1] ACHANTA R., SHAJI A., SMITH K., LUCCHI A., FUA P., SÜSSTRUNK, S. SLIC Superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012, 34(11), pp. 2174–1181, doi: 10.1109/TPAMI.2012.120.

[2] AL-KADI O.S. Supervised texture segmentation: A comparative study. In: *Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Amman, Jordan: 2011, pp. 1–5, doi: 10.1109/AEECT.2011.6132529.

[3] ALKAMA S., CHAHIR Y., BERKANI D. Label maps fusion for the marginal segmentation of multi-component images. *Neural Network World*. 2015, 25(4), pp. 405–426, doi: 10.14311/NNW.2015.25.021.

[4] ALPERT S., GALUN M., BASRI R., BRANDT A. Texture segmentation by multiscale aggregation of filter responses and shape elements. In: *Proceedings of the 9^{th} IEEE International Conference on Computer Vision* (*ICCV*), Nice, France: 2003, pp. 716–723, doi: 10.1109/ICCV.2003.1238418.

[5] ANGELOVA A., ZHU S. Efficient object detection and segmentation for fine-grained recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New Jersey, USA: 2013, pp. 811–818, doi: 10.1109/CVPR.2013.110.

[6] BHOSLE V.V., PAWAR V.P. Texture segmentation: different methods. *International Journal of Soft Computing and Engineering* (IJSCE). 2013, 3(5), pp. 69–74.

[7] BUXHOEVEDEN, D. P. The minicolumn hypothesis in neuroscience. *Brain*, 2002. 125(5), pp. 935–951, doi: 10.1093/brain/awf110.

[8] CAENEN G., FERRARI V., ZALESNY A., VAN GOOL L. Analyzing the layout of composite textures. In: *Proceedings of the 2^{nd} International Workshop on Texture Analysis and Synthesis*. Copenhagen, Denmark: 2002, pp. 15–20.

[9] CLAUSI D.A., DENG H. Design-based texture feature fusion using Gabor filters and co-occurrence probabilities. *IEEE Transactions on Image Processing*. 2005, 14(7), pp. 925–936, doi: 10.1109/TIP.2005.849319.

[10] ÇESMELI E., WANG D.L. Texture segmentation using Gaussian–Markov random fields and neural oscillator networks. *IEEE Transaction on Neural Networks*. 2001, 12(2), pp. 283–286, doi: 10.1109/72.914533.

[11] COMANICIU D., MEER P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002, 24(5), pp. 603–619, doi: 10.1109/34.1000236.

[12] CUI W., GUAN Z., ZHANG Z. An improved region-growing algorithm for image segmentation, In: *Proceedings of International Conference on Computer Science and Software Engineering*. Wuhan, Hubei, China: 2008, Vol. 6, pp. 93–96, doi: 10.1109/CSSE.2008.891.

[13] DEFELIPE J., MARKRAM H., ROCKLAND K.S. The neocortical column. *Frontiers in Neuroanatomy*. 2012, 6, pp. 1–2, doi: 10.3389/fnana.2012.00022.

[14] DONOSER M., BISCHOF H. Using covariance matrices for unsupervised texture segmentation. In: *Proceedings of the 19^{th} International Conference on Pattern Recognition (ICPR 2008)*. Tampa, Florida, USA: 2008, pp. 1–4, doi: 10.1109/ICPR.2008.4761350.

[15] FAUZI M.F.A., LEWIS P.H. A fully unsupervised texture segmentation algorithm. In: *Proceedings of British Machine Vision Conference*, Norwich, UK: 2003, pp. 519–528, doi: 10.5244/C.17.53.

[16] FELZENSZWALB P.F., HUTTENLOCHER D.P. Efficient graph-based image segmentation. *International Journal of Computer Vision*. 2004, 59(2), pp. 167–181, doi: 10.1023/B:VISI.0000022288.19776.77.

[17] FROLOV A. A., HUSEK D., RACHKOVSKIJ D.A. Time of searching for similar binary vectors in associative memory. *Cybernetics and Systems Analysis*. 2006, 42(5), pp. 615–623.

[18] FROLOV A.A., RACHKOVSKIJ D.A., HUSEK D. On information characteristics of Willshaw–like auto–associative memory. *Neural Network World*. 2002, 12(2), pp. 141–157.

[19] FULKERSON B., VEDALDI A., SOATTO S. Class segmentation and object localization with superpixel neighborhoods. In: *Proceedings of the 12-th IEEE International Conference on Computer Vision*. Kyoto, Japan: 2009, pp. 670–677, doi: 10.1109/ICCV.2009.5459175.

[20] GOLTSEV A. An assembly neural network for texture segmentation. *Neural Networks*. 1996, 9(4), pp. 643–653, doi: 10.1016/0893-6080(95)00136-0.

[21] GOLTSEV A.D. *Neural Networks with the Assembly Organization.* Kiev, Ukraine: Naukova Dumka, 2005 (in Russian).

[22] GOLTSEV A., HÚSEK D., FROLOV A. Assembly neural network with nearest-neighbor recognition algorithm. *Neural Network World.* 2005, 15(1), pp. 9–22.

[23] GOLTSEV A., GRITSENKO V. Modular neural networks with Hebbian learning rule. *Neurocomputing*, 2009, 72 (10–12), pp. 2477–2482.

[24] GOLTSEV A., GRITSENKO V. Algorithm of sequential finding the characteristic features of homogeneous texture regions for the problem of image segmentation. *Cybernetics and Computer Engineering.* 2013, 173, pp. 25–34 (in Russian).

[25] GOLTSEV A., GRITSENKO V., KUSSUL E., BAIDYK T. Finding the texture features characterizing the most homogeneous texture segment in the image. *Lecture Notes in Computer Science.* 2015, 9094, Part I, pp. 287–300, doi: 10.1007/978-3-319-19258-1_25.

[26] HUBEL D.H., WIESEL T.N. Brain mechanisms of vision. *Scientific American.* 1979, 241(3), pp. 130–144, doi: 10.1038/scientificamerican0979-150.

[27] HUBEL D.H. *Eye, Brain and Vision.* New York, USA: Scientific American Library, 1988, doi: 10.1002/col.5080130512.

[28] JÄHNE B., SCHARR H., KÖRKEL S. Principles of filter design. In: B. JÄHNE, H. HAUSSECKER, P. GEISSLER, eds. *Handbook of Computer Vision and Applications*: Academic Press, 2000, Vol. 2, pp. 125–152, doi: 10.1016/S0004-3702(00)00028-X.

[29] KAMDI SHILPA, KRISHNA R.K. Image segmentation and region growing algorithm. *International Journal of Computer Technology and Electronics Engineering (IJCTEE).* 2012, 2(1), pp. 103–107.

[30] KHAN M.W. A survey: Image segmentation techniques. *International Journal of Future Computer and Communication.* 2014, 3(2), pp. 89–93, doi: 10.7763/IJFCC.2014.V3.274.

[31] KUSSUL E.M., RACHKOVSKIJ D.A., BAIDYK T.N. On image texture recognition by associative-projective neurocomputer. In: *Proceedings of the ANNIE'91 Conference, Intelligent Engineering Systems through Artificial Neural Networks.* St. Louis, Missouri, USA: 1991, pp. 453–458.

[32] KUSSUL E.M., BAIDYK T.N., LUKOVITCH V.V., RACHKOVSKIJ D.A. Adaptive neural network classifier with multifloat input coding. In: *Proceedings of the 6th International Conference "Neuro-Nimes 93".* Nimes, France: 1993, pp. 209–216.

[33] KUSSUL E.M., KASATKINA L.M., RACHKOVSKIJ D.A., WUNSCH D.C. Application of random threshold neural networks for diagnostics of micro machine tool condition. In: *Proceedings of the IEEE International Joint Conference on Neural Networks "IJCNN'98".* Anchorage, Alaska, USA: 1998, pp. 241–244.

[34] KUSSUL N., SKAKUN S., SHELESTOV A., KRAVCHENKO O., KUSSUL O. Crop classification in Ukraine using satellite optical and SAR images. *International Journal "Information Models and Analyses".* 2013, 2(2), pp. 118–122.

[35] LEISE E.M. (1990). Modular construction of nervous systems: a basic principle of design for invertebrates and vertebrates. *Brain Research Reviews.* 1990, 15(1), pp. 1–23, doi: 10.1016/0165-0173(90)90009-d.

[36] LEVINSHTEIN A., STERE A., KUTULAKOS K.N., FLEET D.J., DICKINSON S.J., SIDDIQI K. TurboPixels: fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2009, 31(12), pp. 2290–2297, doi: 10.1109/TPAMI.2009.96.

[37] LUKOVICH V.V., GOLTSEV A.D., RACHKOVSKIJ D.A. Neural network classifiers for micromechanical equipment diagnostics and micromechanical product quality inspection. In: *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing "EUFIT'97".* 1997, Aachen, Germany: 1997, Vol. 1, pp. 534–536.

[38] MALIK J., BELONGIE S., LEUNG T., SHI J. Contour and texture analysis for image segmentation. *International Journal of Computer Vision (IJCV).* 2001, 43(1), pp. 7–27, doi: 10.1007/978-1-4615-4413-5_9.

[39] MANCAS M., GOSSELIN B., MACQ B. Segmentation using a region-growing thresholding. In: E. R. DOUGHERTY, J. T. ASTOLA, K. O. EGIAZARIAN, eds. *Image Processing: Algorithms and Systems IV*: SPIE Proceedings, 2005, Vol. 5672, pp, 388–398, doi: 10.1117/12.587995.

[40] MAHBUBUR RAHMAN MD. Unsupervised natural image segmentation using mean histogram features. *Journal of Multimedia.* 2012, **7**(5), pp. 332–340, doi: 10.4304/jmm.7.5.332-340.

[41] MELENDEZ J., PUIG D., GARCIA M.A. Multi-level pixel-based texture classification through efficient prototype selection via normalized cut. *Pattern Recognition.* 2010, 43(12), pp. 4113–4123, doi: 10.1016/j.patcog.2010.06.014.

[42] MOBAHI H., RAO S.R., YANG A.Y., SASTRY S.S., MA YI. Segmentattion of natural images by texture and boundary compression. *International Journal of Computer Vision.* 2011, 95(1), pp. 86-98, doi: 10.1007/s11263-011-0444-0.

[43] OJALA T., PIETIKAINEN M., MAENPAA M. Multiresolution grayscale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2002, 24(7), pp. 971–987, doi: 10.1109/TPAMI.2002.1017623.

[44] PENG QIONG, 2015. An image segmentation algorithm research based on region growth. *Journal of Software Engineering.* 2015, 9(3), pp. 673–679, doi: 10.3923/jse.2015.673.679.

[45] PENZ P.A. (1987). The closeness code: an input integer to binary vector transformation suitable for neural network algorithms. In: *Proceedings of the IEEE First Annual International Conference on Neural Networks.* San Diego, CA, USA: 1987, pp. 515–522, doi: 10.1109/MEX.1987.4307059.

[46] RACHKOVSKIJ D.A., KUSSUL E.M. DataGen: a generator of datasets for evaluation of classification algorithms. *Pattern Recognition Letters.* 1998, 19(7), pp. 537–544.

[47] ROUSSON M., BROX T., DERICHE R. Active unsupervised texture segmentation on a diffusion based feature space. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03).* Madison, Wisconsin, USA: 2003, Vol. 2, pp. 699–704, doi: 10.1109/CVPR.2003.1211535.

[48] RUIZ-DEL-SOLAR J. Neural-based architectures for the segmentation of textures. In: *Proceedings of the 15th International Conference on Pattern Recognition.* Barcelona, Spain: 2000, Vol. 3, pp. 1080–1083, doi: 10.1109/ICPR.2000.903733.

[49] SEBE N., LEW M.S. Texture features for content-based retrieval. In: M.S. LEW, ed. *Principles of Visual Information Retrieval. Advances in Computer Vision and Pattern Recognition*: Springer, 2001, pp. 51–85, doi: 10.1007/978-1-4471-3702-3_3.

[50] SHI J., MALIK J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2000, 22(8), pp. 888–905, doi: 10.1109/34.868688.

[51] TIVIVE F.H.C., BOUZERDOUM A. Texture classification using convolutional neural networks. In: *Proceedings of IEEE Region 10 Conference.* Hong Kong, China: 2006, pp. 1–4, doi: 10.1109/TENCON.2006.343944.

[52] TODOROVIC S., AHUJA N. Texel-based texture segmentation. In: *Proccedings of the 12th IEEE International Conference on Computer Vision (ICCV).* Kyoto, Japan: 2009, pp. 841–848, doi: 10.1109/ICCV.2009.5459308.

[53] WEI H., BARTELS M. Unsupervised segmentation using Gabor wavelets and statistical features in LIDAR data analysis. In: *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06).* Hong Kong, China: 2006, Vol. 1, pp. 667–670, doi: 10.1109/ICPR.2006.1145.

[54] WOLF L., HUANG X., MARTIN I., METAXAS D. Patch-based texture edges and segmentation. In: *Proceedings of the 9th European Conference on Computer Vision.* Graz, Austria: 2006, Vol. Part II, pp. 481–493, doi: 10.1007/11744047_37.

[55] YANG A.Y., WRIGHT J., MA Y., SHAKAR SASTRY S. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding.* 2008, 110(2), pp. 212–225, doi: 10.1016/j.cviu.2007.07.005.

[56] YANG C.-K., ROTHKRANZ L.J.M. Surveillance system using abandoned object detection. In: *Proceedings of the International Conference on Computer Systems and Technologies – CompSysTech'11*. Vienna, Austria: 2011, pp. 380–386.

[57] ZHIGUANG YANG, HAIZHOU AI. Demographic classification with local binary patterns. *Lecture Notes in Computer Science*. 2007, 4642, pp. 464–473, doi: 10.1007/978-3-540-74549-5_49.