Chapter 4
Unsupervised Clustering of Natural Images in Automatic Image Annotation Systems

Margarita Favorskaya, Lakhmi C. Jain and Alexander Proskurin

Abstract The chapter is devoted to automatic annotation of natural images joining the strengths of text-based and content-based image retrieval. The Automatic Image Annotation (AIA) is based on the semantic concept models, which are built from large number of patches receiving from a set of images. In this case, image retrieval is implemented by keywords called as Visual Words (VWs) that is similar to text document retrieval. The task involves two main stages: a low-level segmentation based on color, texture, and fractal descriptors (a shape descriptor is less useful due to great variety of visual objects and their projections in natural images) and a high-level clustering of received descriptors into the separated clusters corresponding to the VWs set. The enhanced region descriptor including color, texture (with the high order moments—skewness and kurtosis), and fractal features (fractal dimension and lacunarity) has been proposed. For the VWs generation, the unsupervised clustering is a suitable approach. The Enhanced Self-Organizing Incremental Neural Network (ESOINN) was chosen due to its main benefits as a self-organizing structure and on-line implementation. The preliminary image segmentation permitted to change a sequential order of descriptors entering in the ESOINN as the associated sets. Such approach simplified, accelerated, and decreased the stochastic variations of the ESOINN. Our experiments demonstrate acceptable results of the VWs clustering for a non-large natural image sets. Precision value of clustering achieved up to 85–90 %. Our approach show better precision values and execution time as compared with fuzzy

M. Favorskaya · A. Proskurin
Institute of Informatics and Telecommunications, Siberian State Aerospace University, 31 Krasnoyarsky Rabochy, Krasnoyarsk 660037, Russian Federation
e-mail: favorskaya@sibsau.ru

A. Proskurin
e-mail: proskurin.av.wof@gmail.com

L.C. Jain (✉)
Bournemouth University, Fern Barrow, Poole, UK
e-mail: Lakhmi.Jain@canberra.edu.au

L.C. Jain
University of Canberra, Canberra ACT 2601, Australia

R. Kountchev and K. Nakamatsu (eds.), *New Approaches in Intelligent Image Analysis*, Intelligent Systems Reference Library 108,
DOI 10.1007/978-3-319-32192-9_4

c-means algorithm and classic ESOINN. Also issues of parallel implementation of unsupervised segmentation in OpenMP and Intel Cilk Plus environments were considered for processing of HD-quality images. Execution time has been increased on 26–32 % using the parallel computations.

Keywords Unsupervised clustering · Visual words · Self-organizing incremental neural network · Automatic image annotation · Image features · Image segmentation

4.1 Introduction

Nowadays, the image browsing and retrieval are the embedded WWW tools, which are available for many users in anytime and anywhere. However, the retrieval systems require the development of efficient software tools that is caused by the increasing visual data growth. The image retrieval systems have three frameworks: text-based (since 1970s), content-based (since 1980s), and automatic annotation (since 2000s). In Text-Based Image Retrieval (TBIR) systems, the images are manually annotated by text descriptors [1]. This leads to inaccuracy and duration of a user work. The Content-Based Image Retrieval (CBIR) is free from such disadvantages. The queries into Content-Based Retrieval Systems (CBRS) can be different, for example, a retrieval of features (color, shape, spatial location), abstract objects, real objects, or listed events [2–4]. Image retrieval in Automatic Image Annotation (AIA) assumes that the images can be retrieved in the same manner as text documents. The basic idea of the AIA is to implement the unsupervised learning based on semantic concept models extracted from large number of image samples [5–9]. The images can be retrieved by keywords called as Visual Words (VWs). The AIA systems join the advantages of both TBIR and CBIR systems and additionally solve the task of automatic image annotation using semantic labels.

This chapter is devoted to the retrieval of abstract objects, which is based on the VWs extraction from a non-large set of images. The task involves two main stages: a low-level segmentation based on color, texture, and fractal descriptors (a shape descriptor is less useful due to great variety of visual objects and their projections in natural images [10]) and a high-level clustering of received descriptors into the separated clusters corresponding to the VWs set. Sometimes it is useful to divide images in two global categories: natural or urban scenes, and according to such classification tune an unsupervised procedure of extraction of low-level features. The goal is to develop fast and accurate methods, which are suitable for natural images annotation in the AIA framework.

For the VWs detection, the unsupervised clustering is a suitable approach. The Enhanced Self-Organizing Incremental Neural Network (ESOINN) was chosen due to its main benefits in unsupervised clustering and on-line implementation. This network can be trained adaptively and store the previous data with any increasing volume of input information.

The chapter is organized as follows. A brief literature review is provided by Sect. 4.2. The main statements, methods, and algorithms of unsupervised segmentation and unsupervised clustering are detailed in Sects. 4.3 and 4.4, respectively. Section 4.5 presents a discussion of experimental results of precision and computational speed involving experiments with images from the dataset IAPR TC-12 Benchmark [11]. Conclusion and remarks of future development are drawn in Sect. 4.6.

4.2 Related Work

The literature review includes two main issues: the analysis of unsupervised image segmentation for extraction of low-level features (Sect. 4.2.1) and the overview of unsupervised image clustering (Sect. 4.2.2) in order to receive the high-level semantic descriptors.

4.2.1 Unsupervised Segmentation of Natural Images

Any image, especially natural, involves a set of regions with different textures and colors. During the last decade, many heuristic segmentation methods and algorithms have been designed, which can be concerned to three main approaches:

- Region-based approach including grid-based method, when an image is roughly divided into blocks [12], threshold-based methods of gradient gray-scales image [13], contour-based methods evolving a curve around an object [14], methods of morphological watersheds with preliminary image pyramid building in order to detect the centers of crystallization [15–18], region-based methods including a region growing approach [19–21].
- Model-based approach involving graph-based methods, among which a normalized graph cut [22], statistical models using Bayesian model, Markov chain, Expectation Maximization (EM) algorithm, and others [23–25], auto-regressive models [26, 27], clustering algorithms like *k*-means, which are used to classify pixels into different classes [28].
- Structured-based approach using Haralick structural methods for texture segmentation [29], image segmentation by clustering of spatial patterns [30].

A majority of known CBIR and AIA systems use a region-based segmentation as the close technique for a human vision. Let us notice that for the AIA systems, the unsupervised segmentation is strongly recommended approach.

The unsupervised color image segmentation method based on the estimation of Maximum A Posteriori (MAP) on the Markov Random Fields (MRFs) was proposed by Hou et al. [31]. This method works under the assumption that there are n pixels of m ($m \ll n$) colors in the image I, and any two colors, fore and back, are perceptually distinguishable from each other. The authors used the energy functions approximately in the non-iteration style. A new binary segmentation algorithm based on the slightly tuned Lanczos eigensolver was designed.

The effective unsupervised color image segmentation algorithm, which uses the multi-scale edge information and spatial color content, was represented by Celik and Tjahjadi [32]. The multi-scale edge information is extracted using Dual-Tree Complex Wavelet Transform (DT-CWT). The segmentation of homogeneous regions was obtained using a region growing followed by a region merging in the Lightness and A and B (LAB) color space in the research [33]. The authors proposed the edge-preserving smoothing filter, which removes a noise and retains a contrast between regions. The authors show that their approach provides better boundaries of objects than JSEG and mean-shift algorithms. However, the unsupervised color image segmentation works non-well in the textured images. Sometimes the color image segmentation needs in a priory information and has high computational cost. The use of statistical pattern recognition and Artificial Neural Networks (ANN) with multi-layer perceptron topology was suggested by Haykin [34] in order to segment and make a clustering of images into the pre-determined classes.

Also some hybrid methods exist, for example, 2D autoregressive modeling and the Stochastic Expectation-Maximization (SEM) algorithm. The last one was developed by Cariou and Chehdi [27]. The proposed texture segmentation method has three steps. First, 2D causal non-symmetric half-plane autoregressive modeling of the textured image is realized. Second, the parameters of identifiable mixed distributions and the corrected number of classes are calculated using the SEM algorithm. The second step is finalized by coarse, block-like image pre-segmentation. Third, the original image ought to refine the pixel-based segmentation applying the Maximizer of Posterior Marginals (MPM). Using this hierarchical model in a Bayesian framework, the authors obtained a reliable segmentation by means of Gibbs sampling. This approach provided good segmentation/classification results above 90 % of correct classification with maximum value 99.26 %. The disadvantage is the complicated mathematical calculations.

The well-known method of J-image SEGmentation (JSEG) is concerned to the unsupervised segmentation based on a color-texture model. In the pioneer research of Deng and Manjunath [35], the given color-texture patterns and the estimations of their homogeneity were used. First, the image colors are quantized to several representative classes in the color space without considering the spatial distributions of the colors. Then pixel values are replaced by their corresponding color class labels to form a class-map of the image (J-image). The received class-map can be represented as a special type of homogeneous color-texture regions. Second, a spatial segmentation is executed into this class-map without considering the corresponding pixel color similarity. This work became the basis of following modifications and improvements.

An improved version, combining the classical JSEG algorithm with a local fractal estimator, permits to improve the boundary detection [36]. A model of the texture features using a mixture of Gaussian distributions, which components can be degenerate or nearly-degenerate, was developed by Yang et al. [37]. The authors show the efficiency of their simple agglomerative clustering algorithm derived from a lossy data compression approach. Using 2D texture filter banks or simple fixed-size windows, the algorithm effectively segments an image minimizing the overall coding length of the feature vectors.

Statistical Region Merging (SRM) algorithm based on perceptual growing and region merging was proposed by Nock and Nielsen [38]. An unsupervised GSEG algorithm [21] is based on color-edge detection, dynamic region growth, and multi-resolution region merging procedure. A Partion-based SEGmentation (PSEG) algorithm uses a hierarchical approach, according to which the spatially connected regions group together based on the mean vectors and covariance matrices of a multi-band image [39]. Also the authors introduced the inner and the external measures based on Gaussian distribution, which estimate the goodness for each portion in the hierarchy.

The approach for color–texture segmentation based on graph cut techniques finds optimal color–texture segmentation by regarding it as a minimum cut problem in a weighted graph [40]. A texture descriptor called as texton was introduced to efficiently represent texture attributes of the given image, which is derived from the complex Gabor filtered images estimated in various directions and scales. In the research [40], the texton feature is defined as a magnitude of textons rather than a histogram of textons, which makes it highly effective to apply the graph cut techniques. The problem of color-texture segmentation is formulated in terms of energy $E(\cdot)$ minimization with graph cuts by Eq. 4.1, where A is the data and smoothness constraint, Θ denotes the mixture model parameters, $\lambda > 0$ specifies a balance between a data term $U(A, \Theta)$ and a prior term $V(A)$.

$$E(A, \Theta) = \lambda \cdot U(A, \Theta) + V(A) \tag{4.1}$$

The segmentation energy should be minimized with respect to the labeling A and the model parameter Θ. This method provides better precision and recall results in comparison with JSEG algorithm. The following essential extension of multilayer graph cut approach using multivariate mixed Student's t-distribution and regional credibility merging one can find in [41].

The Blobworld segmentation is widely used method. It is closed to the JSEG algorithm. The pixel clustering is executed in a color-texture-position feature space. First, a common distribution of these features is modeled by a Gaussian mixture. Second, the EM algorithm estimates the parameters of received model. The pixel-cluster membership produces a resulting coarse segmentation of the objects. Vogel et al. proposed the adapting version of Blobworld algorithm, which was called BlobContours segmentation [42]. The idea of displaying the intermediate segmented images as the layers lays in the basis of BlobContours segmentation. Each EM iteration is displayed as a layer, and the user can examine, which layer is the best one. The flood-fill algorithm calculates the average true color for each region instead of using the connected component algorithm from the original Blobworld. However, the blob-approach has a restricted application in the unsupervised segmentation.

Many authors use the measures of similarity to estimate and compare the experimental results. Some of such measures one can find in [43].

4.2.2 Unsupervised Clustering of Images

The good decision for extraction of high-level semantic features is the use of semi-supervised or unsupervised machine learning techniques. The goal of supervised learning is to determine the output values, and the goal of unsupervised learning is to re-distribute the input data into classes and describe these classes. Support Vector Machine (SVM) classification, Bayesian classification, and decision tree technique are concerned to supervised methods, which form the high-level results from the low-level features. In this research, the unsupervised methods are considered for clustering of low-level features into the VWs representation. The traditional *k*-means clustering, fuzzy *c*-means, and clustering based on Self-Organizing Neural Network (SONN) including their modifications are often applied approaches in the CBRS. Let us discuss some approaches for such clustering.

The color moments and Block Truncation Coding (BTC) were used in [44] to extract features as the inputs of *k*-means clustering algorithm. The basis of color moments (mean, standard deviation, and skewness) uses assumption that a distribution of color in an image can be interpreted as a probability distribution. An image is split into R, G, and B components separately, the average values of each component are determined. Then the features are calculated as a set of color moments for R, G, and B values, which are higher and lower the corresponding averages. Such heuristic algorithm can not provide a high accuracy of clustering because color features are computed in a whole image. The closed approaches one can find in [45, 46].

The application of Radial-Based Function Neural Network (RBFNN) for semantic clustering was proposed by Rao et al. [47]. The authors applied the hierarchical clustering algorithm to group the images into classes based only on the color RGB-content; however, the result of received accuracy is absent in research [47].

Self-Organizing Fuzzy Neural Network (SOFNN) can be concerned to a special type of the SONN. The first group of Fuzzy Neural Network (FNN) with the self-tuning capabilities requires the initial rules prior to train. The second group of the FNN is able to automatically create the fuzzy rules from the training data set. In opposite of a traditional clustering, when classes are disjointed, a fuzzy clustering suggests so called soft clustering scheme. In this case, each pattern is associated with every class by a membership function, in other words each class is a fuzzy set of all patterns. A traditional clustering can be obtained from a fuzzy clustering using a threshold of a membership value. The most popular fuzzy clustering algorithm is a Fuzzy C-Means (FCM) algorithm. It is better than the *k*-means algorithm avoiding local minimums. The design of membership functions is the most important problem in a fuzzy clustering because they determine the similarity decomposition and the centroids of classes. An incremental clustering is based on the assumption that it is possible to consider instances one at a time and assign them to the existing classes.

In research [48], the SOFNN was proposed as extended RBFNN, which is a functional equivalent to Takagi-Sugeno-Kang fuzzy systems. First, a self-organizing clustering approach is used to form the structure and obtain the initial values of parameters in a network. Second, a hierarchical on-line self-organizing learning paradigm is employed to adjust the parameters and the structure of the SOFNN. The algorithm of incremental learning was developed, which is capable to generate automatically fuzzy rules according to a simple error criterion based on the differences between calculated and desired output values.

Tung and Quek suggested a Generic Self-Organizing Fuzzy Neural Network (GenSOFNN), which overcomes the drawbacks of fuzzy neural network approach connecting with the necessity of prior knowledge such as a number of classes [49]. The proposed GenSOFNN did not require a pre-definition of the fuzzy rules. The authors show that its training cycle takes place in a single pass of the training data and demonstrated the on-line applications of the GenSoFNNs.

Three new learning algorithms for Takagi-Sugeno-Kang fuzzy system based on a training error and a genetic algorithm were proposed by Malek et al. [50]. First two algorithms involve two stages. In the first stage, the initial structure of the FNN was created by estimating the optimum points of training data in input-output space using *k*-nearest neighbor algorithm and *c*-means methods, respectively. This stage keeps adding new neurons based on an error-based algorithm. In the second stage, the redundant neurons were recognized and removed using a genetic algorithm. Third algorithm built the FNN by a single stage using a modified version of error algorithm. These algorithms were evaluated using two examples: by function of two nonlinear inputs and identification of nonlinear dynamic system.

Fuzzy clustering can be applied with other techniques, for example, invariant moments as the invariant shape features [51]. One of the connected problems is a semantic gap removal, which appears between low-level and high-level features because the images, which are identical in a spatial domain, can be non-identical in a semantic domain [52].

For our experiments, two ways were chosen: without preliminary segmentation of natural images and with preliminary segmentation, description of the last one is located in next Sect. 4.3.

4.3 Preliminary Unsupervised Image Segmentation

Segmentation of natural images is a complicated task due to a great set of regions with various colors and textures. The basic JSEG algorithm [35] with some modification was applied in order to obtain good segmentation results. The segmentation task can be interpreted as an optimization task for search of such division of image, which possesses the predetermined properties according to some functional.

The authors of research [35] referred this functional as *J*-functional, which estimates a quality of segmentation based on a color distribution. However, the direct optimization of *J*-functional is a high resource task. The JSEG algorithm uses a greedy algorithm of optimization, which searches local optimums in each of iterations, calculating *J*-functional in a neighborhood of each pixel.

Two independent steps including color quantization and spatial segmentation are used in this method. In order to extract only a few representative colors, the colors in image are coarsely quantized. For natural images, 10–20 colors are enough for good segmentation. Each pixel is replaced by corresponding color class label. The image of labels is called a class-map, which can be interpreted as a special texture. Each point belongs to a color class in a class-map. In natural images, such classes usually have the overlapping distributions. Under such assumptions, the authors of research [35] consider Z as the set of all N data points in a class-map, $z = (x, y)$, where x, y are spatial coordinates, $z \in Z$ with the mean m provided by Eq. 4.2, on the one hand,

$$m = \frac{1}{N}\sum_{z \in Z} z \tag{4.2}$$

and, on the other hand, suppose that Z is classified into classes Z_i, $i = 1, \ldots, C$ with mean m_i of the N_i points in class Z_i as it is written in Eq. 4.3.

$$m_i = \frac{1}{N_i}\sum_{z \in Z_i} z \tag{4.3}$$

The total variance S_T of class-map points is determined by Eq. 4.4.

$$S_T = \sum_{z \in Z} \|z - m\|^2 \tag{4.4}$$

The total variance of points S_W belonging to the same class is defined by Eq. 4.5.

$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C}\sum_{z \in Z} \|z - m_i\|^2 \tag{4.5}$$

Then *J*-functional can be calculated by Eq. 4.6.

$$J = (S_T - S_W)/S_W \tag{4.6}$$

If value of J is large, then the color classes are more separated from each other, and points inside a class are strongly connected between themselves. The average $\bar{J}$ can be defined by Eq. 4.7, where J_k is *J*-functional over region k, M_k is a number of points in region k, N is a total number of points in a class-map.

$$\bar{J} = \frac{1}{N}\sum_{k} M_k J_k \tag{4.7}$$

A better segmentation means a lower value of $\bar{J}$. Equation 4.6 is a criterion of minimization of segmentation. However, the global optimization of $\bar{J}$ is impossible because any image can be segmented by various ways. Instead of this, J-image is generated, where pixel values correspond to local J-values, which are calculated over small window centered in the pixels. The local J-values become large near a region boundary. The J-image can be represented as a 3D map containing valleys and hills, which are correspond to the region insides and the region boundaries. The size of local window is a multi-scale parameter. A window with small sizes (9 × 9 pixels) is useful to detect edges, and a window with large sizes is used for boundary detection.

A spatial segmentation as a second step of the JSEG algorithm is based on a region-growing method. The pixels of J-image with minimal functional values are accepted as the seeds. A growing process is realized by jointing the neighbor pixels to the seeds. As a result, an initial segmentation is received sometimes with small over-segmented regions. To avoid this artifact, such regions are merged based on color similarity. The agglomerative procedure is applied: the distance values between two neighbor regions are calculated according to a color histogram, the pairs of regions with minimal distance value is merged, then a color histogram is recalculated and the distance values are update. The procedure is repeated until the predetermined maximum distance value between regions will not be achieved.

Our improvement of JSEG results deals with decreasing the original image in four times (the upper level of image pyramid with Gaussian blurring), application of JSEG algorithm to small-sized image, and following stretching transformation of J-image to the initial sizes of original image. A convolution of the transformed J-image with original image provides a final segmentation. The segmentation results for images 38019, 38225, 38755, 39986, and 38756 are represented in Fig. 4.1. The test images were taken from DB IAPR TC-12 Benchmark [11]. The size of original images is 480 × 360 pixels. The size of decreased images was 240 × 180 pixels.

Our approach provides better visual segmentation results without considering the non-significant for the AIA small regions in original images.

4.4 Feature Extraction Using Parallel Computations

In this research, the JSEG algorithm was chosen as a pre-segmentation stage and realized in the designed software tool. Consider the main color, texture, and fractal features extraction (Sects. 4.4.1, 4.4.2 and 4.4.3, respectively) from a pre-segmented

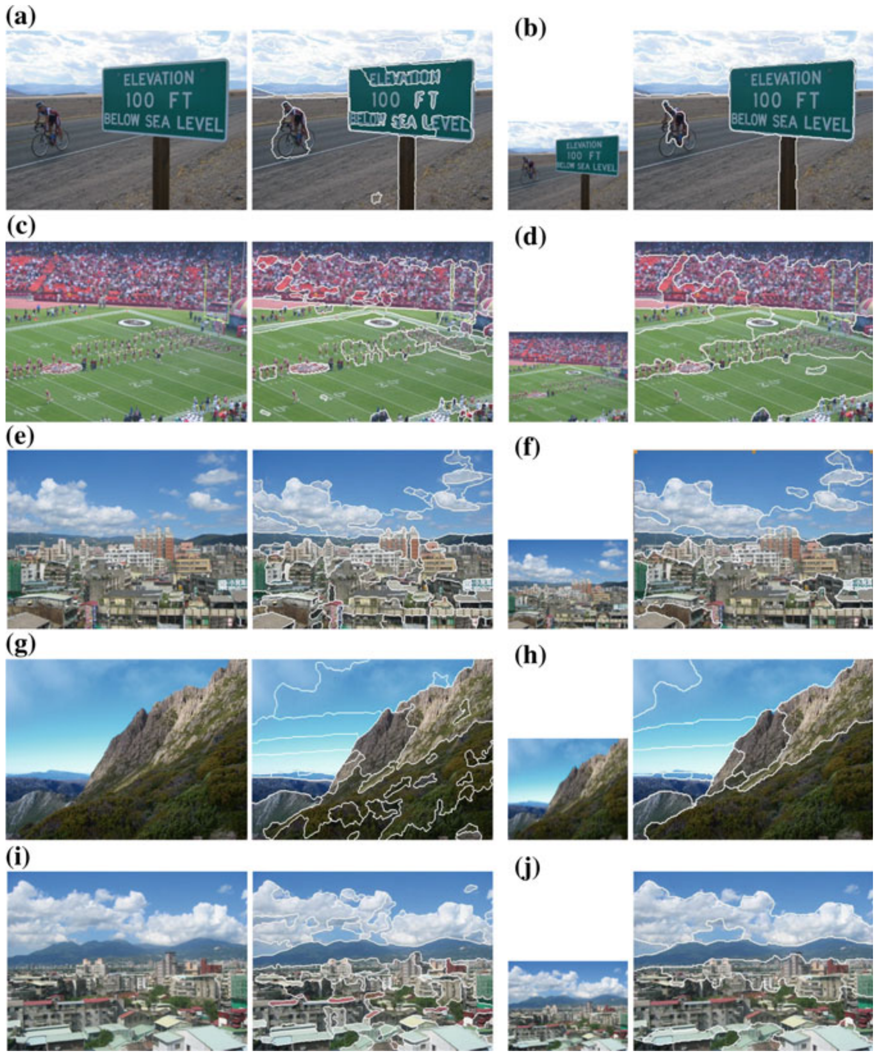


Fig. 4.1 Visual results of JSEG algorithm: **a**, **c**, **e**, **g**, **i** original images and JSEG results; **b**, **d**, **f**, **h**, **j** the resized in four times original images and JSEG results

image in order to create a common image descriptor as a set of region features. The enhanced region descriptor is built in Sect. 4.4.4. Section 4.4.5 provides a description of parallel computations of features.

4.4.1 Color Features Representation

A number of important color features, which are extracted from images or regions, have been proposed in literature including Color Histogram (CH) [53], Color Moments (CM) [54], Color Coherence Vector (CCV) [55], Color Correlogram [56], among others. Notice that MPEG-7 standard restricts a number of color features including Dominant Color Descriptor (DCD), Color Layout Descriptor (CLD), Color Structure Descriptor (CSD), and Scalable Color Descriptor (SCD) [57].

The color moments such as mean, standard deviation, and skewness are the simplest and popular features. They are applied to each component of color spaces mentioned below:

- Red, Green, Blue (RGB).
- Lightness and A and B are the color-opponent dimensions based on nonlinearly compressed CIE (International Commission on Illumination; usually abbreviated CIE for its French name, Commission internationale de l'éclairage) XYZ coordinates (LAB).
- Lightness, Uniform chromaticity scale, Valence (LUV). CIE LUV and CIE LAB were adopted simultaneously by the CIE.
- Hue, Saturation, Value (HSV) or Hue, Saturation, Lightness (HSL).
- Hue, Min, Max, Difference (HMMD).

In current research, the color features (mean, standard deviation, and skewness for each color component) are extracted as the low-level features of each region in HSV-color space. According to the theory of moments, normalized mean μ_c, normalized standard deviation σ_c, and normalized skewness θ_c (values of these parameters are normalized relative to a pixel amount into current region) are calculated for each HSV-component by Eqs. 4.8–4.10, where p_i^c is a pixel value of corresponding color component, NP is a number of pixels in a current region. Let us remember that preliminary image segmentation was executed using the JSEG algorithm.

$$\mu_c = \frac{1}{NP}\sum_{i=1}^{NP} p_i^c \tag{4.8}$$

$$\sigma_c = \frac{1}{NP}\left(\sum_{i=1}^{NP}\left(p_i^c - \mu_c\right)^2\right)^{\frac{1}{2}} \tag{4.9}$$

$$\theta_c = \frac{1}{NP}\sum_{i=1}^{NP}\left(p_i^c - \mu_c\right)^3 \tag{4.10}$$

As a result, nine color features are received and included as FC_0, …, FC_8 components in feature vector, describing a current region.

4.4.2 Calculation of Texture Features

The calculation techniques for texture features are very different. Often statistical texture features are based on moments or local statistical measures such as the six Tamura texture features [58]. The Tamura features include coarseness, directionality, regularity, contrast, line-likeness, and roughness. First three characteristics are more significant, and second three ones are the secondary parameters. The MPEG-7 standard has employed regularity, directionality, and coarseness as the texture browsing descriptor [57]. Unfortunately, the Tamura and the MPEG-7 texture descriptors are non-invariant to a scale.

Also it is possible to calculate the statistical features using a Gray-Level Co-occurrence Matrix (GLCM) [59]. The GLCM provides information about the positions of pixels having similar gray level values. Each element of such matrix contains a number of all pairs of pixels separated by displacement vector **d**, which includes gray levels i and j. Haralick et al. [60] suggested a set of 14 textural features extracted from a co-occurrence matrix. Homogeneity, contrast, and entropy are the main parameters, which are calculated from the GLCM. However, the experiments show that these parameters do not make essential contribution into improvement of CBIR accuracy but increase a computational cost.

The spectral characteristics based on 2D wavelet transform and a Gabor transform have high cost for the CBIR and the AIA. The advantage of Gabor transform is an invariance to a scale. Galloway [61] introduced five original features of run-length statistics, which were built using the analysis of image gray levels. At present, run-length statistics have a historical meaning.

Let z be a random value of intensity, $h(z_i)$ is its histogram, $i = 0, 1, 2, \ldots, Q–1$, Q is a number of brightness levels. Statistical features into a current image region such as normalized average AV, normalized dispersion DS, normalized homogeneity HM, normalized smoothness SM and improved normalized smoothness ISM, normalized entropy EN and improved normalized entropy IEN, normalized skewness SK, and normalized kurtosis KR are provided by Eqs. 4.11–4.19, where SR is a region area, μ_3 and μ_4 are moments of 3rd and 4th orders, σ^3 and σ^4 are standard deviation in 3rd and 4th degrees. All these values are normalized relative to a region area SR.

$$AV = \frac{1}{SR}\sum_{i=0}^{Q-1} z_i h(z_i) \tag{4.11}$$

$$DS = \frac{1}{SR}\sum_{i=0}^{Q-1} (z_i - AV)^2 h(z_i) \tag{4.12}$$

$$HM = \frac{1}{SR}\sum_{i=0}^{Q-1} h^2(z_i) \tag{4.13}$$

$$SM = \frac{1}{SR}\left(1 - \frac{1}{1 + DS\Big/(Q-1)^2}\right) \tag{4.14}$$

$$ISM = \begin{cases} -\log SM, & \text{if } SM > 0 \\ 10, & \text{if } SM = 0 \end{cases} \tag{4.15}$$

$$EN = -\frac{1}{SR}\sum_{i=0}^{Q-1} h(z_i)\,\log_2 h(z_i) \tag{4.16}$$

$$IEN = EN/\log_2 Q \quad Q > 1 \tag{4.17}$$

$$SK = \frac{1}{SR}\frac{\mu_3}{\sigma^3} = \frac{1}{SR}\cdot\sum_{i=1}^{Q-1}\left(\left(\frac{z_i - AV}{\sqrt{DS}}\right)^3 \cdot h(z_i)\right) \tag{4.18}$$

$$KR = \frac{1}{SR}\frac{\mu_4}{\sigma^4} - 3 = \frac{1}{SR}\cdot\sum_{i=1}^{Q-1}\left(\left(\frac{z_z - AV}{\sqrt{DS}}\right)^4 \cdot h(z_i)\right) - 3 \tag{4.19}$$

If parameter $SM = 0$, then its value is forcibly maintained into $NSM = 10$ (small empirical value, differing from 0). Normalized entropy NEN indicates some equalization effect in dark and bright areas of image [62, 63].

Thus, seven texture features (AV, DS, HM, ISM, IEN, SK, and KR) are used as the FT_9, …, FT_{15} components of feature vector, describing a current region.

4.4.3 Fractal Features Extraction

It is well-known, that natural texture surfaces are the spatial isotropic fractals and their 2D intensity function are also fractals. Connected domain A in a topological n-space is self-similarity, when domain A includes N separated non-overlapping and self-similarity copies, and each of copies is reduced by a coefficient r along all coordinate axes. Fractal dimension FD of connected domain A is determined by Eq. 4.20.

$$FD = \log N/\log(1/r) \tag{4.20}$$

Usually fractal surfaces demonstrate a statistical self-similarity, when each of N copies is identical to an original surface by all statistical features. However, to determine a dimension of fractal texture region using Eq. 4.20 is difficult and sometimes impossible. In research [64], two ways for definition of fractal dimension FD were investigated using a cube cover and based on the probability estimations. Let us calculate a measure of domain A on a set R^n. Suppose that domain A is covered by n-ary cube with sizes L_{max}. If domain A is a reduced copy by coefficient r, then $N = r^{-FD}$ sub-cubes exist. Therefore, a number of cubes with sizes $L = r \cdot L_{max}$, which are necessary to cover a whole domain, is determined by Eq. 4.21.

$$N(L) = 1/r^{FD} = [L_{max}/L]^{FD} \tag{4.21}$$

A simple procedure to determine fractal dimension *FD* by Eq. 4.21 involves a cover of connected domain *A* by a grid from *n* cubes with a side length *L* and calculation a number of non-empty *K* cubes. Then fractal dimension *FD* is determined from a line slope of {log *L*; –log *N*(*L*)} in R^n space.

Another way to determine fractal dimension *FD* uses a probability approach. Let *P*(*m*, *L*) be a probability that *m* points into a cube with length side *L* are located near a random point of connected domain *A*. Let total number of points into connected domain *A* be equal *M* (in our case, a connected domain *A* is an image). If a grid from cubes with a length side *L* is imposed in an image, then a number of cubes, including *m* points, is determined as $(M/m) \cdot P(m, L)$ and will be proportional to a power dependence L^{-FD}.

However, various fractal structures with a similar fractal dimension *FD* can have very different textures. The term "lacunarity" was introduced by Mandelbrot [65] to describe such fractals. Mandelbrot proposed some procedures to define a lacunarity *FL*, the most known of which has a view of Eq. 4.22, where *M* is a weight of fractal structure, $\langle M \rangle$ is an estimated weight.

$$FL = \left\langle (M/\langle M \rangle - 1)^2 \right\rangle \tag{4.22}$$

Lacunarity *FL* demonstrates the difference between a weight of fractal structure and an estimated weight. This feature is a statistical characteristic of the second order and changes in a following manner. Lacunarity has low value for a fine-grained textures and high value for a coarse-grained textures. Weight of fractal structure *M* is a function of parameter *L* (Eq. 4.23), where *k* is a proportional coefficient [65].

$$M(L) = kL^{FD} \tag{4.23}$$

Also lacunarity *FL* can be estimated based on a probability approach. Probability *P*(*m*, *L*) includes data for average distortion of weight in fractal structure. Therefore, lacunarity *FL* can be calculated by Eq. 4.24.

$$FL(L) = \frac{M^2(L) - |M(L)|^2}{|M(L)|^2} = \frac{\sum_{m=1}^{N} m^2 P(m,L) - \left|\sum_{m=1}^{N} m\,P(m,L)\right|^2}{\left|\sum_{m=1}^{N} m\,P(m,L)\right|^2} \tag{4.24}$$

Lacunarity estimating by Eq. 4.24 is well for textures with large area but it is non-useful for small area image regions. Let us simplify Eq. 4.24 by introduction of function $C(L)$ provided by Eq. 4.25, where $M_D(L)$ is an average density of weight into a cube with a length side L, $N_P(L)$ is a quotient of division the number cubes with a length side L, which are necessary for a full cover of fractal structure, on the number of points into this fractal structure.

$$C(L) = \frac{M_D(L) - N_P(L)}{M_D(L) + N_P(L)} \tag{4.25}$$

If the smallest texton is less then L, then a weight of fractal structure will distributed uniformly into each cube. In this case, values $M_D(L)$ and $N_P(L)$ have close values, and $C(L) \to 0$. If the smallest texton is large than L, then $C(L) \to 1$. If L value increases, then $C(L) \to 1$ for all fractal structures. Therefore, function $C(L)$ will include the data about textons in both cases.

Two fractal features FD and FL are two components FF_{16} and FF_{17} of feature vector describing a current region.

4.4.4 Enhanced Region Descriptor

Using parameters from Sects. 4.4.1–4.4.3, one can construct a region vector $\mathbf{RF} = \{FC_0, \ldots, FC_8, FT_9, \ldots, FT_{15}, FF_{16}, FF_{17}\}$, which later will be transformed to Region Descriptor $\mathbf{RD}_{ij}$. Values of $\mathbf{RD}_{ij}$ are normalized to the intervals of input values of neural network, where i is a counter of regions in an image j, j is a counter of images in an image set. As a result, an image descriptor $\mathbf{ID}_j = \{\mathbf{RD}_{1j}, \ldots, \mathbf{RD}_{ij}, \ldots\}$ and a set descriptor $\mathbf{SD} = \{\mathbf{ID}_1, \ldots, \mathbf{ID}_j, \ldots\}$ will be constructed. The extended region descriptor is our contribution in the unsupervised clustering for image annotation problem. For simplicity, denote a set of Region Descriptor $\{\mathbf{RD}_{ij}\}$ as a weight input vector $\mathbf{W}_x$, because region descriptors enter to the inputs of classical ESOINN randomly.

A transition from low-level features to high-level semantics is usually tracked by reducing the "semantic gap", which includes four categories:

- Object ontology to define high-level concepts.
- Introduction a relevant feedback into retrieval loop for continuous learning of users' intention.
- Generation semantic templates to support high-level image retrieval.
- Supervised or unsupervised learning methods to associate low-level features with query concepts.

Our choice deals with the last one due to high possibilities of self-organizing approach. Let us remark that a redundancy is eliminated during segmentation stage in order to avoid an over-segmentation of natural images.

4.4.5 Parallel Computations of Features

The parallelizing of program code includes the types mentioned below:

- Parallelizing of data means a multiple execution of the same algorithm with various input data. Data are divided into fragments, and each fragment is processed by an allocated computer core.
- Functional parallelizing is a parallel execution of sets of operations by functional feature. Simple example of such functional decomposition is a decomposition of task into subtasks such as input of initial data, processing, output of results, visualization of results, etc. Functional parallelizing is achieved using sequential or sequential-parallel "conveyor" between subtasks. Each subtask provides a parallelizing of data inside.
- Algorithmic parallelizing finds such fragments in algorithm, which can be executed in parallel. Synthesis of parallel algorithms based on algorithmic parallelizing is called an algorithmic decomposition. During algorithmic decomposition, it would like to divide a task into large and rarely connecting branches with homogeneous distribution of data processing along the branches. Main distinction between algorithmic and functional parallelizing is in following. Functional parallelizing merges only functional close operators from algorithm, and algorithmic parallelizing does not consider a functional similarity of operators.

For implementation of parallel algorithms, some standards are available, among which OpenMP standard [66] is used for parallelizing of program code in languages C, C++, and Fortran. Also the extension of language C++ with parallel possibilities called as Intel Cilk Plus [67] is developed.

In OpenMP standard, a paralleling is executed explicitly by insert the special directives and by call the additional functions in a program code. The standard OpenMP realizes the parallel computations in the multi-thread mode, when the "main" thread creates a set of sub-threads, and a current task is distributed between the sub-threads. First, a program is executed in "sequential" area with single "main" thread (process). Second, several sub-threads are generated in "parallel" area, and the program code is distributed between them. Third, all sub-threads except the "main" thread are finalized, and again a "sequential" area is continued. The standard OpenMP supports the embedding of parallel areas.

The Intel Cilk Plus environment is a dynamic thread scheduler, including a set of keywords. Keywords inform a compiler about the application of scheme scheduling. A parallel Cilk-program creates a task queue. The "executors" capture the tasks, and free thread performs a current task. In the Intel Cilk Plus environment, the semantics of sequential program is supported. However, a program can be executed in sequential or parallel modes due to available resources. Use of extended index notation is an essential difference in comparison with OpenMP standard that provides a paralleling of vector instructions of processor.

The enhanced region descriptor involves color, texture, and fractal features calculated in a neighborhood of considered pixel. Calculation of color and textural features (Eqs. 4.8–4.19) can be implemented in a parallel mode. Fractal features requires a separate non-parallel computation. The color and texture features are computed by two steps. First, stochastic data acquisition is accomplished in a neighborhood of current pixel: the normalized means of color channels are calculated using Eq. 4.8, and local histogram is built based on texture features. Second, color and texture features (Eqs. 4.9–4.19) are calculated directly. Two basic cycles are implemented in parallel mode. There are an external cycle for image with sizes $(w/k_w) \times (h/k_h)$, where w and h are width and height of image, respectively, k_w and k_h are width and height of image segment, respectively, and the internal cycles for segment with sizes $k_w \times k_h$.

For parallel computation of texture features, a whole image is divided into segments, and a processing of segments is distributed between cores of processor. A way of image partitioning in vertical/horizontal bands or rectangle blocks determines a structure of parallel procedure. In order to increase a computational cost, a parallelizing of external cycles in whole image is required. For this purpose, a processor directive "#pragma omp parallel for" in the case of OpenMP standard and a keyword "cilk_for" in the case of the Intel Cilk Plus environment can be applied. The calculations of color and texture features do not connected. Therefore, the additional parallel areas in random access memory can be determined for color and texture features separately.

4.5 Clustering of Visual Words by Enhanced SOINN

As a result of features extraction (Sect. 4.3), any image can be represented as a set of regions with corresponding region vectors **RF** = $\{FC_0, \ldots, FC_8, FT_9, \ldots, FT_{15}, FF_{16}, FF_{17}\}$ as a collection of color, texture, and fractal features. Direct comparison of feature sets in a metric space is not preferable due to segmentation errors and noises. Therefore, a clustering methodology is a single way to receive good results. In literature review (Sect. 4.2), it was shown that the unsupervised clustering is more suitable for the VWs detection, and among unsupervised clustering methods the SOINN was chosen.

The clustering procedure groups the regions of all annotated images into subsets (VWs) in such manner that the regions with similar features are grouped together, while the regions with different features belong to the different classes. Formally, a clustering structure **S** is represented as a set of subsets **C** = $\{C_1, \ldots, C_K\}$, Eq. 4.26. Consequently, any element in **S** belongs to one and only one subset.

$$\mathbf{S} = \bigcup_{k=1}^{K} C_i \quad \text{and} \quad C_i \cap C_j = 0 \quad \text{for} \quad i \neq j \qquad (4.26)$$

The ESOINN proposed by Furao et al. [68] is applied as the useful unsupervised clustering technique in many applications: robots navigation [69, 70] microarray data analysis [71], multi-agent systems [72], among others. The basic concepts of ESOINN are discussed shortly in Sect. 4.5.1, and algorithm of ESOINN is presented in Sect. 4.5.2.

4.5.1 Basic Concepts of ESOINN

The ESOINN was developed to overcome the main disadvantages of the two-layer SOINN as mentioned below:

- The separated training of the first layer and the second layer.
- The second layer is unsuitable for on-line incremental training: the changing of training results in the first layer causes the re-training of the second layer.
- The necessity of user-determined parameters, if a within-class insertion appears.
- The SOINN cannot separate a set with the high-density overlapping areas.

The ESOINN is adapted using a single-layer network structure. To build an edge between nodes, the ESOINN adds a condition to judge, and after some training iterations it separates nodes to the different subclasses deleting edges, which lie in the overlapping areas. The ESOINN achieves the within-class insertion slightly but it is more suitable for on-line or even life-long training tasks than two-layer SOINN.

A single layer of ESOINN is continuously adapted according to the input data structure defining a number and a topology of classes. When an input vector enters, the ESOINN finds two nearest nodes as the winner and the second winner by the predetermined metric. Using a threshold criterion of similarity (the maximum distances between vectors owing to the same cluster), the network judges: an input vector belongs to the winner or the second winner cluster or not. A distribution of input data is unknown, and a threshold criterion is updated adaptively for each separate node. A threshold criterion for node T_i is calculated by Eq. 4.27, where N_i is a set of neighbor nodes, $\mathbf{W}_i$ and $\mathbf{W}_j$ are the weight vectors of nodes i and j, respectively.

$$T_i = \max_{j \in N_i} \left\| \mathbf{W}_i - \mathbf{W}_j \right\| \tag{4.27}$$

If a node i has not the connected neighbor nodes, then a threshold criterion Eq. 4.27 is transformed in Eq. 4.28, which is defined as a minimum distance between nodes, where N is a set of all network nodes.

$$T_i = \min_{j \in N \setminus \{i\}} \left\| \mathbf{W}_i - \mathbf{W}_j \right\| \tag{4.28}$$

An input vector is inserted as the first node of new class, if distance between an input vector and the winner or the second winner is more than a threshold value between the winner and the second winner. If an input vector belongs to the cluster of the winner or the second winner, then an edge between the winner and the second winner is created with 0 “age”, and the “age” of all edges linked to the winner is increased by 1.

Then a density p_i of the winner is updated by Eq. 4.29, where $\overline{d_i}$ is a mean value of distances between node j and its neighbor nodes.

$$p_i = 1 \Big/ \left(1 + \overline{d_i}\right)^2 \tag{4.29}$$

If a mean value of distances between node j and its neighbor nodes is large, then a number of nodes and a density p_i of node i will have small values, and vice versa. For each iteration λ, only a density of winner-node is calculated. The accumulated density h_i of winner-node is provided by Eq. 4.30, where n is a total number of iterations (calculated as $n = LT/\lambda$, LT is a total number of input vectors), K is a number of iterations, when a density value for node i exceeds 0.

$$h_i = \frac{1}{K} \cdot \sum_{l=1}^{n} \sum_{k=1}^{\lambda} p_i \tag{4.30}$$

After re-calculation of density, a counter of wins M_i (for a winner-node) is increased by 1. The change of weight vectors of the winner ΔW_i and its neighboring nodes ΔW_j ($j \in N_i$) are determined by Eqs. 4.31 and 4.32, where $\mathbf{W}_x$ is a weight input vector.

$$\Delta W_i = \frac{1}{M_i} \cdot (\|\mathbf{W}_x\| - \|\mathbf{W}_i\|) \tag{4.31}$$

$$\Delta W_j = \frac{1}{100 \cdot M_i} \cdot \left(\|\mathbf{W}_x\| - \|\mathbf{W}_j\|\right) \tag{4.32}$$

Then all edges, the “age” of which is higher a threshold value age_{max}, are removed. If number of input vectors does not achieved λ iterations, then a following input vector is submitted. Otherwise, the overlaps between classes are detected and removed by including additional subclasses.

One can find the detailed description of algorithms for separation a composite class into subclasses, a building the edges between nodes, and a classifying nodes to the different classes in research [69].

4.5.2 *Algorithm of ESOINN Functioning*

The algorithm of the ESOINN functioning includes the steps mentioned below.

Step 1. Set a minimal number of the predetermined segments into a set of images. Number of the pre-determined segments is defined using the JSEG algorithm.

This is our first proposed distinction in comparison with the ESOINN, which initialization is started always from two random nodes. (Let the ESOINN with our color-texture-fractal descriptor be called as the dESOINN.)

Step 2. Input a weight input vector $\mathbf{W}_x$. The second proposed distinction connects with the order of ranked vectors $\{\mathbf{W}_x\}$ at the inputs of the dESOINN. Vectors $\{\mathbf{W}_x\}$ is sorted according to the results of previous segmentation. Therefore, the winner and the second winner are defined at the first steps, and at the following steps the dESOINN is trained using the remaining samples from current segment. Sometimes an input vector $\mathbf{W}_x$ cannot be associated with the current winner owing to coarse segmentation errors. Such input vector ought to be rejected from a sample. This approach reinforces the current winner and makes the stochastic dESOINN more stable. Then the input vectors $\{\mathbf{W}_x\}$ concerning to another segment are clustered by the dESOINN. The proposed approach is especially useful for clustering of non-large set of natural images.

Step 3. Define the nearest node (the winner) a_1 and the second nearest node (the second winner) a_2. If a distance between the input vector $\mathbf{W}_x$ and nodes a_1 or a_2 exceeds threshold values T_i calculating by Eqs. 4.27 and 4.28, then the input vector is considered as a new node and added to a node set. Go to Step 2.

Step 4. Increment the "age" of all edges connecting with a node a_1 by 1.

Step 5. Define the edge creation necessary between nodes a_1 and a_2.

Step 6. Recalculate the accumulated density h_i of the winner-node by Eq. 4.30.

Step 7. Increment the counter of wins M_i by 1.

Step 8. Calculate the weight vectors of the winner ΔW_i and its neighboring nodes ΔW_j using Eqs. 4.31 and 4.32.

Step 9. Remove the edges, "age" of which has more value than a pre-determined parameter age_{max}.

Step 10. If a number of the input vectors $\mathbf{W}_x$ is multiple to a parameter λ, then it is required to update the subclasses for each node and remove the "noisy" nodes using Eqs. 4.33 and 4.34, where N is a number of nodes in a node set, c_1 and c_2 are the empirical coefficients. Equations 4.33 and 4.34 are used, if a node has two or one neighbors, respectively

$$h_i < c_1 \cdot \sum_{j=1}^{N_a} h_j / N \tag{4.33}$$

$$h_i < c_2 \cdot \sum_{j=1}^{N_a} h_j / N \tag{4.34}$$

In experiments, the non-large sets of images were used. Therefore, the additional condition Eq. 4.35 was introduced to remove the single nodes, where c_3 is an empirical coefficient.

$$h_i < c_3 \cdot \sum_{j=1}^{N_a} h_j / N \tag{4.35}$$

Step 11. If a clustering process is finished, then it is needed to determine a number of classes, the output sample vectors for each class, and stop the algorithm.
Step 12. Go to Step 2, if the ESOINN continues to work.

4.6 Experimental Results

For experiments, 120 images from the dataset IAPR TC-12 Benchmark [11] were selected as the 10 sets including 12 images in each set. An example set (12 images and their segmented prototypes by JSEG algorithm with removal small size fragments, Set NN 01) are presented in Fig. 4.2.

The first type of experiments was directed to obtain the precision estimations. The average values of color-texture-fractal features for segments from a set of images, representing in Fig. 4.2, are summarized in Table 4.1. As a result, five clusters (VWs) were determined by the dESOINN as it is show in Fig. 4.3.

As one can see, the clusters from Fig. 4.3 represent three types of objects—"Houses", "Sky", and "Water". However, the dESOINN divided the cluster "Houses" into three clusters. This decision reflects the differences in values of color and texture features (Table 4.1).

Three algorithms were compared: fuzzy c-means, the ESOINN, and the dESOINN. The last one begins its work by use a predetermined minimum number of clusters provided by the JSEG algorithm. For initialization, the following parameters were applied: $\lambda = 50$, $age_{\max} = 5$, $c_1 = 0.01$, $c_2 = 0.3$, and $c_3 = 1.05$.

The average precision of algorithms PRC was calculated by Eq. 4.36, where C is a total number of clusters, NTP_i is a number of true positive examples (true detected regions) and NFP_i is a number of false positive examples (false detected regions) in cluster i relatively the expert estimations.

$$PRC = \frac{1}{C} \sum_{i=1}^{C} \frac{NTP_i}{NTP_i + NFP_i} \tag{4.36}$$

All calculations had been repeated 100 times, and then a precision was averaged out. The parameter "Number of clusters" was chosen as the most frequent value during clustering. The generalized estimations of precision and execution time for fuzzy c-means, ESOINN, and dESOINN algorithms are summarized in Table 4.2. For experiments, PC Acer JM50-HR with a single processor core, Intel Core i5-2430 M 2,4 GHz, RAM Kingston 1333 MHz (PC3-10700) DDR3 8 GB, VC NVIDIA GeForce GT 540 M, 1 GB, SSD Smartbuy, 128 GB was used.

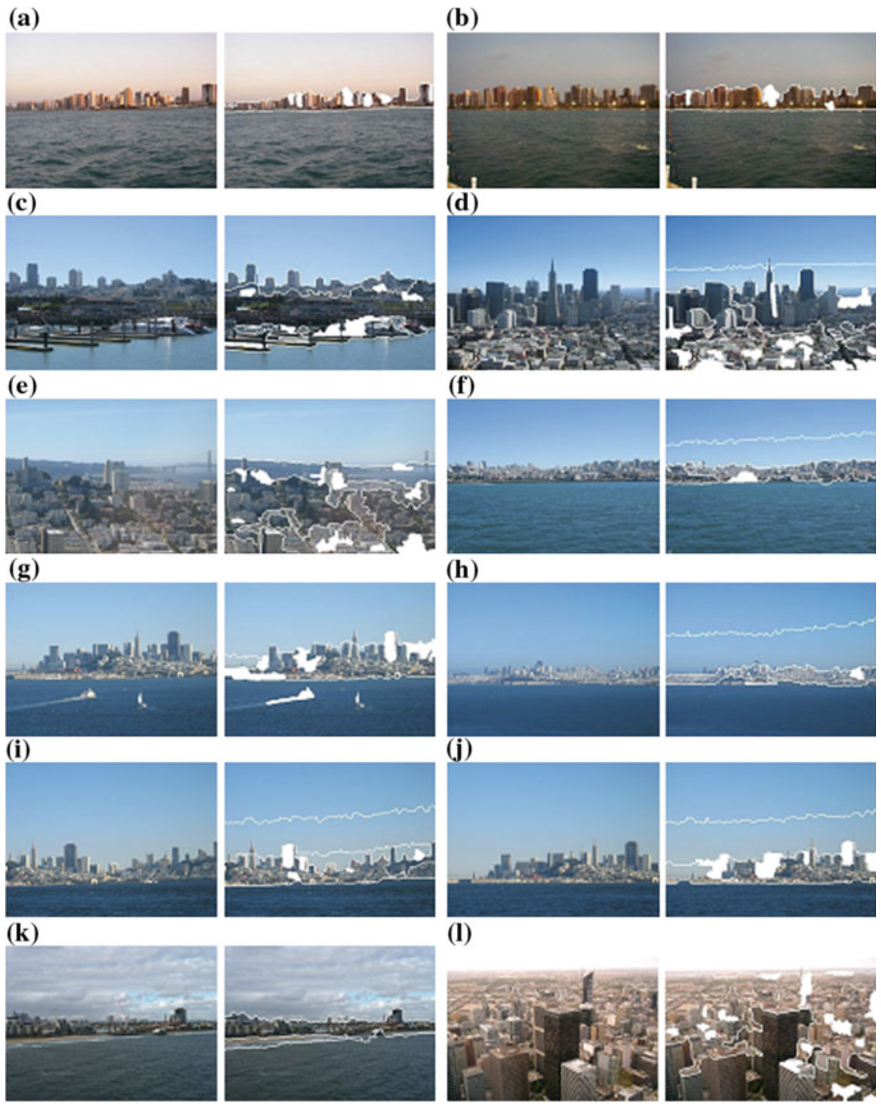


Fig. 4.2 The original images and their segmented prototypes from the database IAPR TC-12 Benchmark: **a** image 2954, **b** image 2956, **c** image 38056, **d** image 38060, **e** image 38063, **f** image 38097, **g** image 38129, **h** image 38183, **i** image 38273, **j** image 38277, **k** image 39458, **l** image 40417

The data from Table 4.2 shows that a precision of VWs using the ESOINN and the dESOINN is better than received by fuzzy *c*-means algorithm. Also the determined clusters are close for human perception. A creation of VWs by ESOINN or dESOINN is slowly in 3–5 times against fuzzy *c*-means algorithm for small

Table 4.1 The average values of color-texture-fractal features of Set NN 01

Feature	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
FC_0 (mean H)	0.552435	0.568377	0.458251	0.0596694	0.285296
FC_1 (mean S)	0.324153	0.534038	0.160444	0.0120252	0.185824
FC_2 (mean V)	0.84274	0.444355	0.366143	0.952904	0.492806
FC_3 (st_deviation H)	0.00832996	0.00378337	0.0365661	0.0267154	0.0854395
FC_4 (st_deviation S)	0.00842481	0.030075	0.0393522	0.00138649	0.0738243
FC_5 (st_deviation V)	0.00783369	0.0274488	0.0433435	0.000830239	0.120126
FC_6 (skewness H)	0.00011649	–7.84582e-07	–0.000495041	0.00225364	–2.34724e-05
FC_7 (skewness S)	–2.24345e-06	–5.33866e-05	0.000101857	2.29497e-08	0.000930306
FC_8 (skewness V)	−1.35649e-06	4.79419e-05	9.35369e-05	−1.85642e-10	0.00132605
FT_9 (mean)	0.84274	0.444355	0.366143	0.952904	0.492806
FT_{10} (variance)	0.000128232	0.00102089	0.00232532	1.8876e-06	0.0162171
FT_{11} (homogeneity)	0.243134	0.0658959	0.0429404	0.844896	0.0175021
FT_{12} (im_smoothness)	3.892003	2.992619	2.635473	5.724094	1.801059
FT_{13} (im_entripy)	0.301775	0.54859	0.624985	0.0471946	0.77565
FT_{14} (skewness)	–0.0365511	0.227385	0.078331	0.270309	0.603937
FT_{15} (kurtosis)	0.242881	0.928843	0.838573	1.24215	1.0476
FF_{16} (fractal_dim)	2.43106	2.34751	2.51404	2.21268	2.88331
FF_{17} (lacunarity)	0.000203711	0.00578365	0.0225221	2.29768e-06	0.0867445

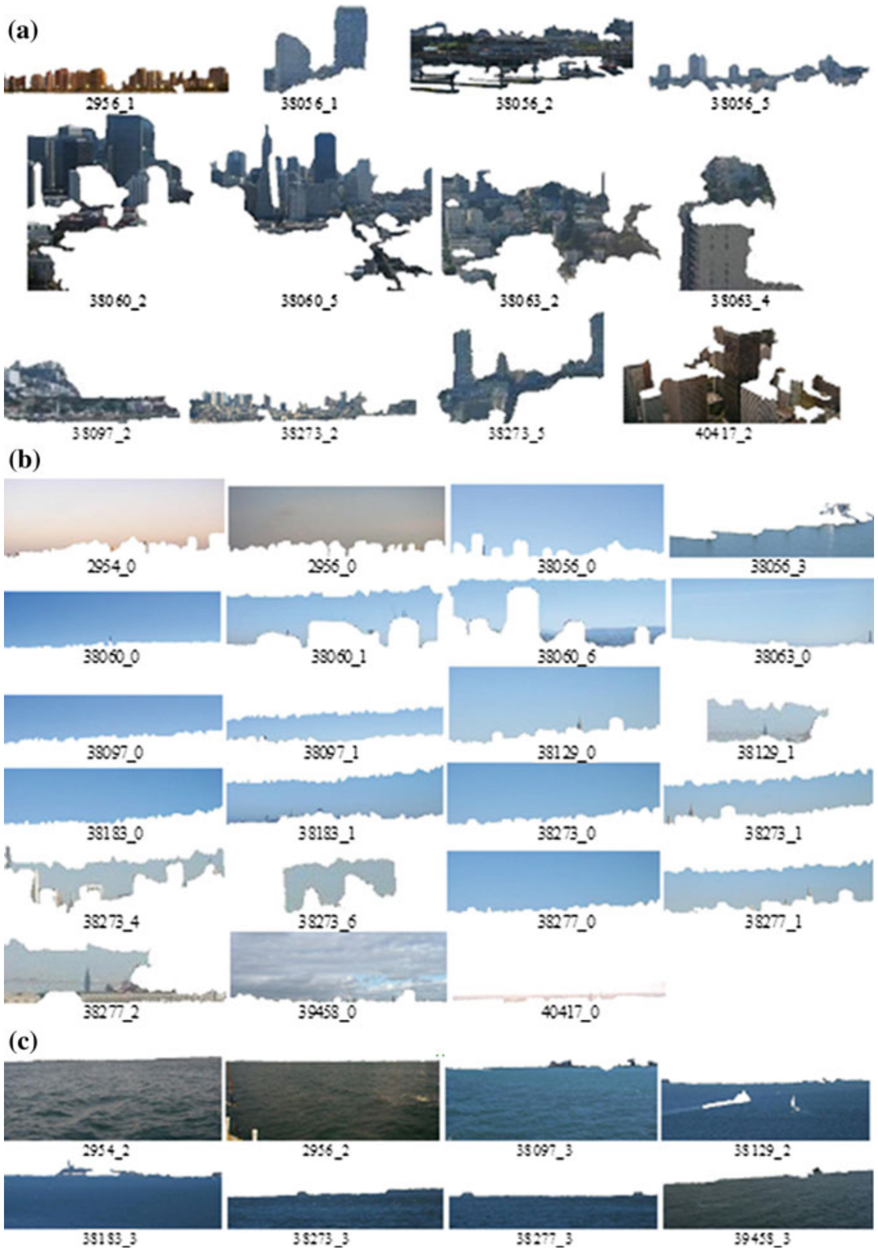
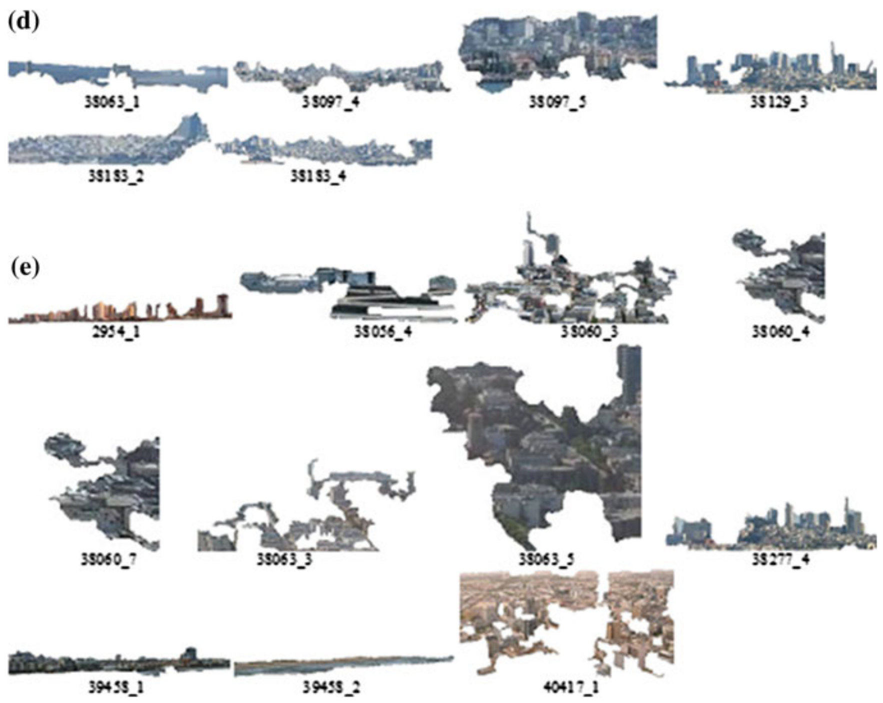


Fig. 4.3 Five clusters of VWs determined by dESOINN, Set NN 01: **a** cluster 0, which includes 12 segments 2956_1, 38056_1, 38056_2, 38056_5, 38060_2, 38060_5, 38063_2, 38063_4, 38097_2, 38273_2, 38273_5, 40417_2, **b** cluster 1, which includes 23 segments 2954_0, 2956_0, 38056_0, 38056_3, 38060_0, 38060_1, 38060_6, 38063_0, 38097_0, 38097_1, 38129_0, 38129_1, 38183_0, 38183_1, 38273_0, 38273_1, 38273_4, 38273_6, 38277_0, 38277_1, 38277_2, 39458_0, 40417_0, **c** cluster 2, which includes 8 segments 2954_2, 2956_2, 38097_3, 38129_2, 38183_3, 38273_3, 38277_3, 39458_3, **d** cluster 3, which includes 6 segments 38063_1, 38097_4, 38097_5, 38129_3, 38183_2, 38183_4, **e** cluster 4, which includes 11 segments 2954_1, 38056_4, 38060_3, 38060_7, 38060_7, 38063_3, 38063_5, 38277_4, 39458_1, 39458_2, 40417_1



◀ **Fig. 4.3** (continued)

number of samples. However, the unsupervised clustering with large sets of images will be promising. The main benefits of ESOINN algorithm are a possibility of the unsupervised clustering and the on-line implementation. This network can be trained by novel data adaptively, and it stores the previous data with any increasing volume of input information.

The additional experiments provided a comparison of precision and execution time results with large number of annotated images. Seven sets with 50, 100, 150, 200, 250, 300, and 350 images from the dataset IAPR TC-12 Benchmark [11] were selected and tested by basic ESOINN, DBSCAN algorithm [73], X-Means algorithm [74], and dESOINN. The experiments were implemented using the same PC Acer JM50-HR (Table 4.3).

For the ESOINN initiation, the same parameters were tuned as in the main experiment. The calculations had been repeated 100 times, and then the precision values were averaged. The often received values are chosen as the parameter "Number of clusters".

The plots in Fig. 4.4 show the generalized precision and time dependences for these four algorithms. One can see that the dESOINN algorithm provides better results for large number of annotated images.

Table 4.2 The comparative results of precision and execution time for fuzzy *c*-means, ESOINN, and dESOINN algorithms

Set NN	VWs, determined by expert	Fuzzy *c*-means			ESOINN			dESOINN		
		Number of clusters	Precision, %	Time, ms	Number of clusters	Precision, %	Time, ms	Number of clusters	Precision, %	Time, ms
00	5	5	65.53	79	5	65.13	193	5	**67.28**	186
01	3	6	81.04	98	6	83.22	195	5	**84.66**	182
02	5	5	63.90	68	5	65.54	194	5	**66.63**	193
03	4	4	68.41	56	4	71.16	182	4	**71.47**	182
04	3	5	82.59	91	4	88.84	222	4	**90.58**	215
05	5	6	69.07	162	6	69.13	242	6	**70.68**	236
06	4	4	72.23	69	4	**73.89**	183	5	73.71	175
07	4	4	70.93	65	5	**74.27**	275	4	73.84	251
08	6	8	67.16	162	8	65.50	223	8	**69.52**	223
09	6	6	73.79	114	6	79.07	244	6	**81.83**	232

Table 4.3 The comparative results of precision and execution time for ESOINN, DBSCAN, X-Means, and dESOINN algorithms

Number of images	VWs, determined by expert	ESOINN			DBSCAN			X-Means			dESOINN		
		Number of clusters	Precision, %	Time, ms	Number of clusters	Precision, %	Time, ms	Number of clusters	Precision, %	Time, ms	Number of clusters	Precision, %	Time, ms
50	45	50	64.39	1896	46	74.7	18447	60	78.83	1887	57	90.91	1346
100	67	67	62.37	6620	80	75.2	73824	74	80.24	5406	71	86.5	4004
150	91	102	61.74	12687	118	72.43	164172	88	78.18	17544	91	84.78	7446
200	111	123	61.51	19631	127	71.08	294536	116	75.05	27336	109	85.87	12083
250	136	149	62.69	35671	146	70.5	465530	134	73.52	35598	138	83.02	18753
300	155	161	61.12	48397	144	69.24	657850	165	75.6	42738	156	83.61	26034
350	165	170	59.87	60404	174	67.59	878639	175	72.92	56049	165	82.38	33843

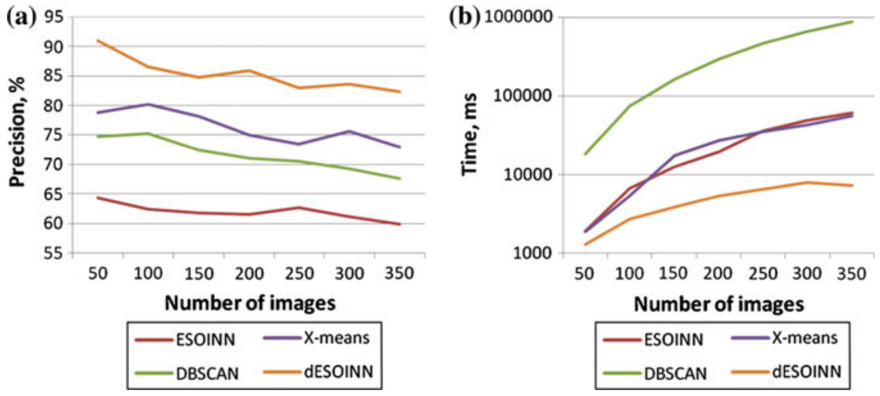


Fig. 4.4 Generalized dependences for ESOINN, DBSCAN, X-Means, and dESOINN algorithms: **a** precision, **b** execution time

The second type of experiments was devoted to increase a computational speed by parallel processing. A version of non-optimized algorithm was realized in Microsoft Visual C++ 2010 package. Versions of optimized algorithm were implemented in OpenMP and Intel Cilk Plus environments. The test dataset contained six sets, each set included 10 images. These six sets were formed from images with different resolutions, i.e. 1920 × 1080 pixels, 2560 × 1600 pixels, 2800 × 2100 pixels, 3840 × 2160 pixels, 3646 × 2735 pixels, and 4096 × 3072 pixels. A test parallel processing was executed in personal computer and servers with different configurations.

For mentioned above six sets of images, the mean values of processing time were estimated using various paralleling algorithms. Four samples, including 40 chosen randomly images, were formed. Each randomly chosen image was processed 25 times. Such methodology permits to decrease the influence of external factors such as activity of background task of operating system and available free hardware resources. Experiments show that a computational speed increases on 26–32 % using parallelizing algorithms. Hardware characteristics influence on computational speed directly. For example, a processing time using Intel Core i5-3450 (3.1 GHz) was in 2–2.2 times less than a processing time using Intel Core 2 Quad Q6600 (2.4 GHz).

A processing time of non-optimized algorithm was defined as 1 in order to calculate a relative speedup factor for optimized algorithms. The speedup factors were calculated for all six sets of images using 2, 3, and 4 threads. Average values of speedup factors for parallelizing algorithms are represented in Fig. 4.5.

Also additional tests were executed in order to compare OpenMP and Intel Cilk Plus environments. The results are drawn in Table 4.4. One can see the advantage of Intel Cilk Plus environment.

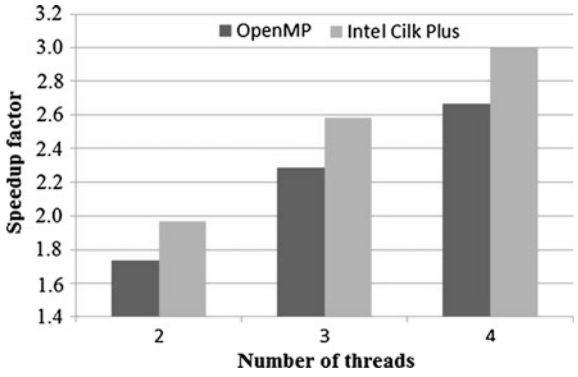


Fig. 4.5 Mean values of speedup factors

Table 4.4 Values of speedup factors

Image sizes	Number of threads								
	OpenMP			Intel Cilk Plus			Differences in values of speedup factors, %		
	2	3	4	2	3	4	2	3	4
2 MB	1.75	2.29	2.68	1.93	2.51	2.92	10.27	9.79	9.04
4 MB	1.74	2.30	2.64	1.96	2.58	2.99	12.90	12.11	13.38
6 MB	1.73	2.27	2.65	2.00	2.61	3.01	15.40	14.77	13.34
8 MB	1.74	2.29	2.69	1.94	2.58	2.99	11.05	12.35	11.18
10 MB	1.72	2.28	2.66	1.99	2.61	3.02	15.92	14.58	13.58
12 Mb	1.73	2.27	2.66	1.98	2.62	3.05	14.17	15.35	14.58
Mean value	1.74	2.28	2.67	1.97	2.58	3.00	13.29	13.16	12.52

4.7 Conclusion and Future Development

In this chapter, the AIA issues were investigated by the VWs extraction from the restricted image sets. The enhanced feature set describing an image region was suggested, which includes color, texture, and fractal features. Three algorithms were compared: fuzzy *c*-means, the ESOINN, and color-texture-fractal descriptor ESOINN (dESOINN) in the precision estimation and the execution time. The precision of VWs using the ESOINN and the dESOINN is higher then fuzzy *c*-means algorithm provides. The experiments demonstrate that the unsupervised clustering of large sets of images based on the neural network approach is promising than other algorithms.

Parallelizing algorithms permit to increase essentially an image processing including images of HD-quality. A computational speed was increased on 26–32 % using algorithms with parallel implementation. Also experiments show that Intel Cilk Plus environment provides the speedup values on 9–14 % higher in comparison with OpenMP environment due to effective balance of loading.

Future work will be directed on development of algorithms, annotating not only natural images but also complicated urban scenes. Also an image categorization is a useful procedure in the AIA systems. The experiments with other datasets will be executed in future.

References

1. Tamura, H., Yokoya, N.: Image database systems: a survey. Pattern Recogn. **17**(1), 29–43 (1984)
2. Jain, A.K., Vailaya, A.: Image retrieval using color and shape. Pattern Recogn. **29**(8), 1233–1244 (1996)
3. Antani, S., Kasturi, R., Jain, R.: A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. Pattern Recogn. **35**(4), 945–965 (2002)
4. Islam, M.M., Zhang, D., Lu, G.: Automatic categorization of image regions using dominant color based vector quantization. In: IEEE International Conference on Digital Image Computing: Techniques and Applications (DICTS'2008), pp. 191–198 (2008)
5. Chang, E., Goh, K., Sychay, G., Wu, G.: CBSA: content-based soft annotation for multimodal image retrieval using Bayes point machines. IEEE Trans. Circ. Syst. Video Technol. **13**(1), 26–38 (2003)
6. Cusano, C., Ciocca, G., Schettini, R.: Image annotation using SVM. In: Proceedings of SPIE, SPIE internet imaging, vol. 5304, pp. 330–338 (2004)
7. Carneiro, G., Chan, A.B., Moreno, P.J., Vasconcelos, N.: Supervised learning of semantic classes for image annotation and retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **29**(3), 394–410 (2007)
8. Liu, Y., Zhang, D., Lu, G.: Region-based image retrieval with high-level semantics using decision tree learning. Pattern Recogn. **41**(8), 2554–2570 (2008)
9. Cavus, O., Aksoy, S.: Semantic scene classification for image annotation and retrieval. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G. C., Loog, M. (eds) Structural, Syntactic, and Statistical Pattern Recognition, pp. 402–410. Springer, Berlin (2008)
10. Rao, C., Kumar, S.S., Mohan, B.C.: Content based image retrieval using exact Legendre moments and support vector machine. Int. J. Multimedia Appl. **2**(2), 69–79 (2010)
11. Index of /imageclef/resources. IAPR TC-12 Benchmark. http://www-i6.informatik.rwth-aachen.de/imageclef/resources/iaprtc12.tgz. Accessed 20 Dec 2014
12. Maree, R., Geurts, P., Piater, J., Wehenkel, L.: Random subwindows for robust image classification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005), vol. 1, pp. 34–40 (2005)
13. Balasubramanian, G.P., Saber, E., Misic, V., Peskin, E., Shaw, M.: Unsupervised color image segmentation by dynamic color gradient thresholding. In: SPIE 6806, Human Vision and Electronic Imaging XIII, pp. 68061H–68061H-9 (2008)
14. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. Image Process. **10**(2), 266–277 (2001)
15. Vanhamel, I., Pratikakis, I., Sahli, H.: Multiscale gradient watersheds of color images. IEEE Trans. Image Process. **12**(6), 617–626 (2003)
16. Makrogiannis, S., Vanhamel, I., Fotopoulos, S., Sahli, H., Cornelis, J.: Watershed-based multiscale segmentation method for color images using automated scale selection. J. Electron. Imaging **14**(3), 1–16 (2005)
17. Jung, C.R.: Combining wavelets and watersheds for robust multiscale image segmentation. Image Vis. Comput. **25**(1), 24–33 (2007)

18. Favorskaya, M.N., Petukhov, N.Y.: Comprehensive calculation of the characteristics of landscape images. J. Opt. Technol. **77**(8), 504–509 (2010)
19. Mansouri, A.R., Mitiche, A., Vazquez, C.: Multiregion competition: a level set extension of region competition to multiple region image partitioning. Comput. Vis. Image Underst. **101**(3), 137–150 (2006)
20. Alnihou, J.: An efficient region-based approach for object recognition and retrieval based on mathematical morphology and correlation coefficient. Int. Arab. J. Inf. Technol. **5**(2), 154–161 (2008)
21. Ugarriza, L.G., Saber, E., Vantaram, S.R., Amuso, V., Shaw, M., Bhaskar, R.: Automatic image segmentation by dynamic region growth and multi-resolution merging. IEEE Trans. Image Process. **18**(10), 2275–2288 (2009)
22. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Intell. **22**(8), 88–905 (2000)
23. Carson, C., Belongie, S., Greenspan, H., Malik, J.: Blobworld: image segmentation using expectation-maximization and its application to image querying. IEEE Trans. Pattern Anal. Mach. Intell. **24**(8), 1026–1038 (2002)
24. Tu, Z., Zhu, S.C.: Image segmentation by data-driven Markov chain Monte Carlo. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 657–673 (2002)
25. Rui, S., Jin, W., Chua, T.S.: A novel approach to auto image annotation based on pairwise constrained clustering and semi-Naıve Bayesian model. In: 11th International Conference on Multimedia Modelling, pp. 322–327 (2005)
26. Alata, O., Ramananjarasoa, C.: Unsupervised textured image segmentation using 2-D quarter-plane autoregressive model with four prediction supports. Pattern Recogn. Lett. **26**(8), 1069–1081 (2005)
27. Cariou, C., Chehdi, K.: Unsupervised texture segmentation/classification using 2-D autoregressive modeling and the stochastic expectation-maximization algorithm. Pattern Recogn. Lett. **29**(7), 905–917 (2008)
28. Mezaris, V., Kompatsiaris, I., Strintzis, M.G.: An ontology approach to object-based image retrieval. In: International Conference on Image Processing (ICIP'2003), vol. 2, pp. 511–514 (2003)
29. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Trans. SMC–3 **6**, 610–621 (1973)
30. Xia, Y., Feng, D., Wang, T., Zhao, R., Zhang, Y.: Image segmentation by clustering of spatial patterns. Pattern Recogn. Lett. **28**(12), 1548–1555 (2007)
31. Hou, Y., Lun, X., Meng, W., Liu, T., Sun, X.: Unsupervised segmentation method for color image based on MRF. In: International Conference on Computational Intelligence and Natural Computing (CINC'2009), vol. 1, pp. 174–177 (2009)
32. Celik, T., Tjahjadi, T.: Unsupervised color image segmentation using dual-tree complex wavelet transform. Comput. Vis. Image Underst. **114**(7), 813–826 (2010)
33. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
34. Haykin, S.O.: Neural Networks and Learning Machines, 3edn. McMaster University, Prentice-Hall, Canada (2008)
35. Deng, Y., Manjunath, B.S.: Unsupervised segmentation of color-texture regions in images and video. IEEE Trans. Pattern Anal. Mach. Intell. **23**(8), 800–810 (2001)
36. Komati, K.S., Salles, E.O.T., Filho, M.S.: Unsupervised color image segmentation based on local fractal dimension and J-images. In: IEEE International Conference on Industrial Technology (ICIT'2010), pp. 303–308 (2010)
37. Yang, A.Y., Wright, J., Ma, Y., Sastry, S.S.: Unsupervised segmentation of natural images via lossy data compression. Comput. Vis. Image Underst. **110**(2), 212–225 (2008)
38. Nock, R., Nielsen, F.: Statistical region merging. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1452–1458 (2004)
39. Martınez-Uso, A., Pla, F., Garcıa-Sevilla, P.: Unsupervised color image segmentation by low-level perceptual grouping. Pattern Anal. Appl. **16**(4), 581–594 (2013)

40. Kim, J.S., Hong, K.S.: Color–texture segmentation using unsupervised graph cuts. Pattern Recogn. **42**(5), 735–750 (2009)
41. Yang, Y., Han, S., Wang, T., Tao, W., Tai, X.C.: Multilayer graph cuts based unsupervised color-texture segmentation using multivariate mixed Student's t-distribution and regional credibility merging. Pattern Recogn. **46**(4), 1101–1124 (2013)
42. Vogel, T., Nguyen, D.Q., Dittmann, J.: BlobContours: adapting Blobworld for supervised color- and texture-based image segmentation. In: Multimedia Content Analysis, Management, and Retrieval, SPIE, vol. 6073, pp. 158–169 (2006)
43. Unnikrishnan, R., Hebert, M.: Measures of similarity. In: IEEE Workshop on Application of Computer Vision, vol. 1, pp. 394–400 (2005)
44. Silakari, S., Motwani, M., Maheshwari, M.: Color image clustering using block truncation algorithm. Int. J. Comput. Sci. Issues **4**(2), 31–35 (2009)
45. Kekre, H.B., Mirza, T.: Content based image retrieval using BTC with local average thresholding. In: International Conference on Content Based Image Retrieval (ICCBIR'2008). Niagara Falls, Canada, pp. 5–9 (2008)
46. Chakravarti, R., Meng, X.: A study of color histogram based image retrieval. In: IEEE 6th International Conference on Information Technology: New Generations, pp. 1323–1328 (2009)
47. Rao, P.S., Vamsidhar, E., Raju, G.S.V.P., Satapat, R., Varma, K.V.S.R.P.: An approach for CBIR system through multi layer neural network. Int. J. Eng. Sci. Technol. **2**(4), 559–563 (2010)
48. Long, X., Su, D., Hu, R.: Incremental leaning algorithm for self-organizing fuzzy neural network. In: IEEE 7th International Conference on Computer Science & Education (ICCSE'2012), pp. 71–74 (2012)
49. Tung, W.L., Quek, C.: GenSoFNN: A generic self-organizing fuzzy neural network. IEEE Trans. Neural Netw. **13**(5), 1075–1086 (2002)
50. Malek, H., Ebadzadeh, M.M., Rahmati, M.: Three new fuzzy neural networks learning algorithms based on clustering, training error and genetic algorithm. Appl. Intell. **37**(2), 280–289 (2011)
51. Hao, P., Ding, Y., Fang, Y.: Image retrieval based on fuzzy kernel clustering and invariant moments. In: 2nd International Symposium on Intelligent Information Technology Application (IITA'2008), Shanghai, vol. 1, pp. 447–452 (2008)
52. Wang, H., Mohamad, D., Ismail, N.A.: Semantic Gap in CBIR: automatic objects spatial relationships semantic extraction and representation. Int. J. Image Process. **4**(3), 192–204 (2010)
53. Swain, M.J., Ballard, D.H.: Color indexing. Int. J. Comput. Vis. **7**(1), 11–32 (1991)
54. Dubey, R.S., Choubey, R., Bhattacharjee, J.: Multi feature content based image retrieval. Int. J. Comput. Sci. Eng. **2**(6), 2145–2149 (2010)
55. Pass, G., Zabith, R.: Histogram refinement for content-based image retrieval. In: IEEE Workshop on Applications of Computer Vision, pp. 96–102 (1996)
56. Huang, J., Kuamr, S., Mitra, M., Zhu, W.J., Zabih, R.: Image indexing using color correlogram. In: International Conference on Computer Vision and Pattern Recognition (CVPR'1997), San Juan, Puerto Rico, pp. 762–765 (1997)
57. Manjunath, B.S., Salembier, P., Sikora, T.: Introduction to MPEG-7: Multi-media Content Description Language. John Wiley & Sons Ltd., New York (2002)
58. Tamura, H., Mori, S., Yamawaki, T.: Texture features corresponding to visual perception. IEEE Trans. Syst. Man Cybern. **8**(6), 460–473 (1978)
59. Selvarajah, S., Kodituwakku, S.R.: Analysis and comparison of texture features for content based image retrieval. Int. J. Latest Trends Comput. **2**(1), 108–113 (2011)
60. Haralick, R.M., Shanmugum, K., Dinstein, I.: Textural features for image classification. IEEE Trans. Syst. Man Cybern. **3**(6), 610–621 (1973)
61. Galloway, M.M.: Texture analysis using gray level run lengths. Comput. Graph. Image Process. **4**(2), 172–179 (1975)

62. Favorskaya, M., Damov, M., Zotin, A.: Intelligent texture reconstruction of missing data in video sequences using neural networks. In: Tweedale, J.W., Jain, L.C. (eds.) Advanced Techniques for Knowledge Engineering and Innovative Applications, pp. 163–176. Springer, Berlin (2013)
63. Favorskaya, M., Damov, M., Zotin, A.: Accurate spatio-temporal reconstruction of missing data in dynamic scenes. Pattern Lett. Recogn. **34**(14), 1694–1700 (2013)
64. Favorskaya, M.N., Petukhov, N.Y.: Recognition of natural objects on air photographs using neural networks. Optoelectron. Instrum. Data Process. **47**(3), 233–238 (2011)
65. Mandelbrot, B.B.: The Fractal Geometry of Nature. Freeman, San Francisco (1982)
66. Slabaugh, G., Boyes, R., Yang, X.: Multicore Image Processing with OpenMP. IEEE Sig. Process. Mag. **27**(2), 134–138 (2010)
67. Saleem, S., Lali, I.U., Nawaz, M.S., Nauman, A.B.: Multi-core program optimization: parallel sorting algorithms in Intel Cilk Plus. Int. J. Hybrid Inf. Technol. **7**(2), 151–164 (2014)
68. Furao, S., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Netw. **20**(8), 893–903 (2007)
69. Tangruamsub, S., Tsuboyama, M., Kawewong, A., Hasegawa, O.: Mobile robot vision-based navigation using self-organizing and incremental neural networks. In: International Joint Conference on Neural Networks (IJCNN'2009), pp. 3094–3101 (2009)
70. Najjar, T., Hasegawa, O.: Self-organizing incremental neural network (SOINN) as a mechanism for motor babbling and sensory-motor learning in developmental robotics. In: Rojas, I., Joya, G., Gabestany, J. (eds.) Advances in Computational Intelligence, pp. 321–330. Springer, Berlin (2013)
71. De Paz, J.F., Bajo, J., Rodríguez, S., Corchado, J.M.: Computational intelligence techniques for classification in microarray analysis. In: Bichindaritz, I., Vaidya, S., Jain, A., Jain, L.C. (eds.) Computational Intelligence in Healthcare 4. Studies in Computational Intelligence, pp. 289–312. Springer, Berlin (2010)
72. De Paz, J.F., Rodríguez, S., Bajo, J.: Multiagent systems in expression analysis. In: Demazeau, Y., Pavón, J., Corchado, J.M., Bajo, J. (eds.) 7th International Conference on Practical Applications of Agents and Multi-Agent Systems. Advances in Intelligent and Soft Computing, pp. 217–226. Springer, Berlin (2009)
73. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: Density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining (KDD‘1996), pp. 226–231. AAAI Press (1996)
74. Ishioka, T.: An expansion of X-means for automatically determining the optimal number of clusters. In: IASTED International conference on Computational Intelligence (CI'2005), Calgary, Alberta, Canada, pp. 91–96 (2005)