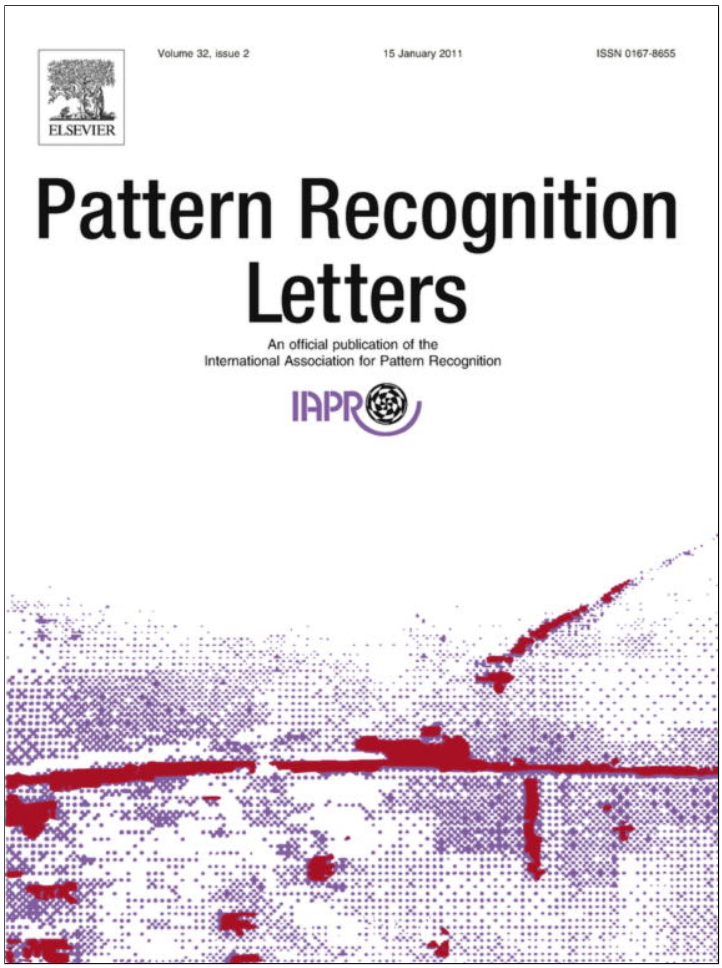

Pattern Recognition Letters 32 (2011) 359–367

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A de-texturing and spatially constrained *K*-means approach for image segmentation

Max Mignotte

DIRO, Univ. de Montréal, C.P. 6128, Succ. Centre-ville, Montréal, Canada H3C 3J7

ARTICLE INFO

Article history:
Received 17 December 2009
Available online 1 October 2010
Communicated by M. Couprie

Keywords:
Unsupervised segmentation
De-texturing
Spatially constrained *K*-means
Berkeley image database

ABSTRACT

This paper presents a new and simple segmentation method based on the *K*-means clustering procedure and a two-step process. The first step relies on an original *de-texturing* procedure which aims at converting the input natural textured color image into a color image, without texture, that will be easier to segment. Once, this de-textured (color) image is estimated, a final segmentation is achieved by a spatially-constrained *K*-means segmentation. These spatial constraints help the iterative *K*-means labeling process to succeed in finding an accurate segmentation by taking into account the inherent spatial relationships and the presence of pre-estimated homogeneous textural regions in the input image. This procedure has been successfully applied on the Berkeley image database. The experiments reported in this paper demonstrate that the proposed method is efficient (in terms of visual evaluation and quantitative performance measures) and performs competitively compared to the best existing state-of-the-art segmentation methods recently proposed in the literature.

1. Introduction

Image segmentation is an important pre-processing step which consists in dividing the image scene into spatially coherent regions sharing similar attributes. This preliminary low-level vision task has been widely studied in the last decades since it is of critical importance in many image understanding algorithms, computer vision and graphics applications. Presently, the problem of finding a fast, simple and automatic method, able to efficiently segment a color textured image remains open.

Due to their simplicity and efficiency, clustering approaches were one of the first techniques used for the segmentation of (textured) natural images (Banks, 1990). After the selection and the extraction of the image features (usually based on color and/or texture and computed on (possibly) overlapping small windows centered around the pixel to be classified), the feature samples, handled as vectors, are grouped together in compact but well-separated clusters corresponding to each class of the image. The set of connected pixels belonging to each estimated class thus defined the different regions of the scene. The method known as *K*-means (or Lloyd's algorithm) (Lloyd, 1982) (and its fuzzy version called fuzzy *C*-means) are some of the most commonly used techniques in the clustering-based segmentation field, and more generally, "by far, the most popular clustering algorithm used in industrial applications and machine learning" (Berkhin, 2002).

Nevertheless, despite its popularity for general clustering, *K*-means suffers from three major shortcomings for the difficult problem of the unsupervised segmentation of textured color image;

(1) First, this clustering utilizes an iterative procedure that converges monotonically to a local minima. This convergence to a bad local solution may be due either (or both) to a bad initialization (i.e., a bad choice of the initial cluster centers) or to the complexity or to the high-dimensional of the dataset (such as the high dimensional feature descriptors required to characterize different color textures).
(2) Such clustering method only involves a partitioning in the feature space without (at all) considering spatial constraints. This considerably reduces the efficiency of this algorithm for the image segmentation issue. Since spatial relationships are essential attributes of any image, spatial constraints should be necessarily taken into account to achieve an efficient noise robust image segmentation.
(3) Finally, the *K*-means assumes, often wrongly, the presence of spherical clusters with equal volumes (or equivalently, the shapes of each feature distribution associated to each class are assumed to be Gaussian with identical variance).

In this paper, we propose an original and simple segmentation strategy based on the *K*-means procedure that remedies these above-enumerated problems. First, in order to apply the *K*-means clustering algorithm in a reduced dimension space, in which the search of well-separated clusters can converge faster to a better local minima (related to a more accurate segmentation map), we propose, in a first step, to *simplify* the input color textured image into a color image without texture. Thanks to this pre-processing

E-mail address: mignotte@iro.umontreal.ca
URL: http://www.iro.umontreal.ca/~mignotte/

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.09.016

step (that we will call in the following a *de-texturing* procedure) the difficult textured image segmentation problem (which necessarily should use a set of high dimensional texture feature descriptors) is reduced to solving a (noisy) color image segmentation which is drastically less complex than a color textured image segmentation problem.

In our application, this de-texturing procedure is simply based on a simple combination of several *K*-means segmentations applied on the input image expressed by different color spaces and using, as input multidimensional feature descriptor, the set of values of the re-quantized color histogram estimated around the pixel to be classified. Once, a de-textured (color) image is estimated, the final segmentation is simply achieved by a spatially-constrained *K*-means segmentation using, as simple cues, a feature vector with the set of color values contained around the pixel to be classified. The spatial constraint allows to take into account the inherent spatial relationships of any image and helps the iterative *K*-means labeling process to succeed in finding an optimal solution, i.e., an accurate segmentation map.

The remainder of this paper is organized as follows: Section 2 describes the proposed model i.e., the so-called de-texturing procedure and the spatially constrained *K*-means segmentation model. Finally, Section 3 presents a set of experimental results and comparisons with existing segmentation techniques.

2. Proposed model

2.1. De-texturing procedure

This de-texturing approach (described in Algorithm 2) is a three-step process as follows:

(1) The first step is simply based on a *K*-means segmentation of the input image using the Manhattan distance (L_1 norm) and as simple cues (i.e., as input multidimensional feature descriptor) the set of values of the re-quantized color histogram, with equidistant binning, estimated around each pixel to be classified. In our application, this local histogram is equally re-quantized (for each of the three color channels) in a $N_b = q^3$ bin descriptor, computed on an overlapping squared fixed-size (N_0) neighborhood centered around the pixel to be classified (see Fig. 1 and Mignotte, 2008 for more details).
(2) The preceding step defines an segmentation of the image into small homogeneous texture regions (each one is defined by the set of connected pixels belonging to a same texture class or equivalently to a same cluster and to this end, we use a simple flood fill algorithm). Given this partition $\mathcal{R}$ into regions of this image (see Fig. 3(b)–(d)), the second step of this de-texturing procedure consists in [1] subdividing all the regions with more than W_0 (=5000 in our application) pixel size into disjoint pieces of size lower than W_0 pixels in order to limit regions with a large number of pixels which could be due to a bad segmentation.[1] To this end, we have used the simple subdivision algorithm presented in pseudo-code in (Mignotte, 2007), and [2] simply replacing each pixel value belonging to each small regions $\mathcal{R}_j$ of $\mathcal{R}$ by its mean color value. This defines a simulation J_c of the input image into a region partition constrained to be spatially piecewise uniform (in the color sense).
(3) The one and two above-mentioned steps are repeated on the input image expressed in different color spaces and the averaging of these different partitions into regions with an uniform (i.e., constant) color level value allows to give our de-textured image (see Fig. 2 and Fig. 3(e)).

Estimation of the bin descriptor for each pixel

$\mathcal{N}_{\mathbf{x}}$	Set of pixel locations $\mathbf{x}$ within the $N_0 \times N_0$ neighborhood region centered at $\mathbf{x}$
$h[\,]$	Bin descriptor : array of N_b floats ($h[0]$, $h[1]$, $\cdots$, $h[N_b-1]$)
N_b	$= q^3$ (bin descriptor length with q integer)
$\lfloor . \rfloor$	Integer part of .

for *each pixel* $\mathbf{x} \in \mathcal{N}_{\mathbf{x}}$ *with color value* $R_{\mathbf{x}}, G_{\mathbf{x}}, B_{\mathbf{x}}$ **do**

- $k \leftarrow q^2 \cdot \lfloor \frac{q \cdot R_{\mathbf{x}}}{256} \rfloor + q \cdot \lfloor \frac{q \cdot G_{\mathbf{x}}}{256} \rfloor + \lfloor \frac{q \cdot B_{\mathbf{x}}}{256} \rfloor$
- $h[k] \leftarrow h[k] + 1/N_0^2$

Fig. 1. Algorithm 1: Bin descriptor estimation for each pixel.

De-Texturing procedure

I_c	Input image expressed in color space c
$\mathcal{C}$	Set of color spaces $= \{rgb, hsv, yiq, xyz, ...\}$
K_0	Number of classes of each K-means
$h[\,]$	Bin descriptor used by K-means
$\mathcal{R}_{I_c}$	Partition into regions obtained from I_c
W_0	Maximal size of each regions
J_c	$\mathcal{R}_{I_c}$ with a mean color value associated to each region corresponding to I_c

for *each color space* $c \in \mathcal{C}$ **do**

- $\mathcal{R}_{I_c} \leftarrow K$-means ($K_0$ classes) on I_c and using as feature vector $h[\,]$ (see Algo. 1)
- $\mathcal{R}_{I_c} \leftarrow$ Subdivide all the regions into small disjoint pieces smaller than W_0 pixels
- $J_c \leftarrow$ Replacement of each pixel value belonging to each region of $\mathcal{R}_{I_c}$ by its mean color value

De-textured image $\leftarrow$ averaging of these different J_c

Fig. 2. Algorithm 2: De-texturing procedure.

Let us note that this de-texturing approach can be viewed as the edge-preserving denoising algorithm proposed in (Mignotte, 2007) for which, respectively:

(1) Each texture applied on each one of these distinct regions is considered as a noise (a de-texturing is thus herein considered as a denoising approach).
(2) Each input (de-textured) image can be modeled by an union of a number of nonoverlapping and distinct regions of uniform (i.e., constant) color level value.
(3) The set of Markov Chain Monte-Carlo (MCMC) simulations of region partition maps used in (Mignotte, 2007) is herein replaced by a set of *K*-means segmentations into regions, on the input image expressed in different color spaces.

[1] After this splitting procedure, for example, a region with a size of 12,000 pixels will be divided into two regions of $W_0 = 5000$ pixels and one region with a size of 2000 pixels. $W_0 = 5000$ is thus the maximum size allowed of each region in the segmentation map.

Fig. 3. From left to right; (a) a natural image from the Berkeley database (numbers 239,007 and 134,052), (b)–(d) its partition into regions obtained by the *K*-means segmentation (K_0 classes) using as feature vector $h[\,]$ (see Algorithm 1) for the input image expressed in respectively, RGB, HSV, YIQ color spaces, (e) the averaging of these different partition into regions (for which we have replaced each pixel value belonging to each region by its mean color value) finally defined the final de-textured image.

2.2. Spatially constrained K-means clustering

As noted above, once a de-textured (color) image is estimated, the final segmentation is simply achieved by a spatially-constrained *K*-means segmentation using, as simple cues, a feature vector with the set of color values contained around the pixel to be classified.

Besides, in order to further help this *K*-means clustering process to succeed in finding an accurate partition, a hard constraint enforcing the spatial contiguity of each (likely) textural region is

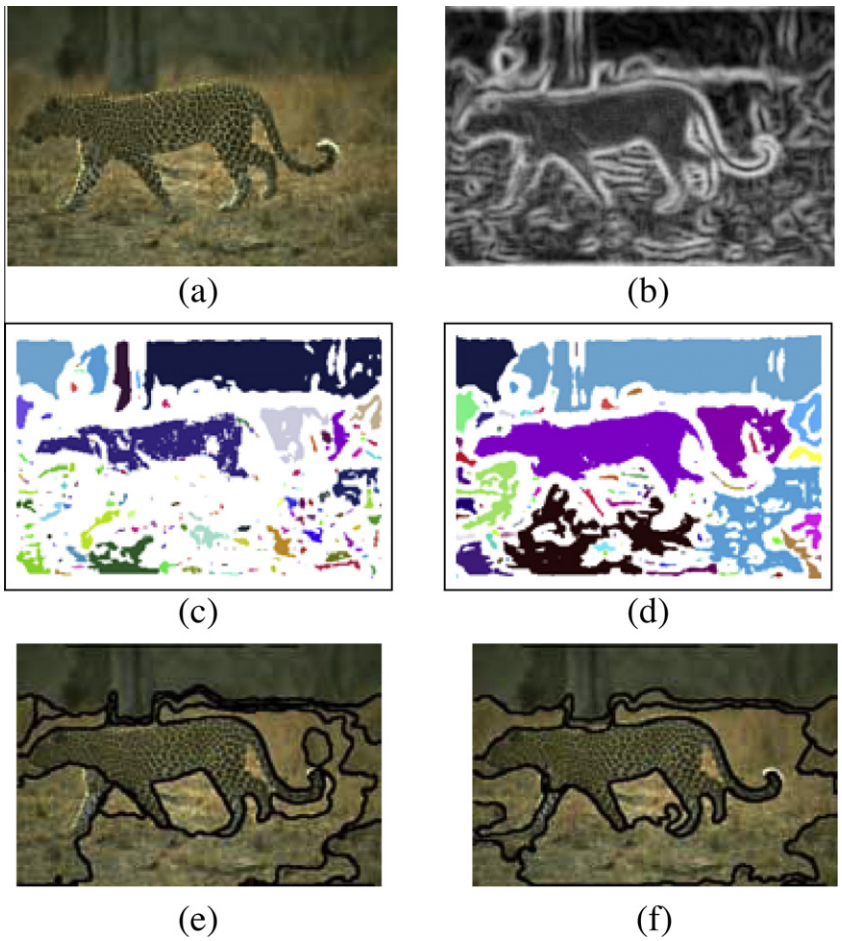

Fig. 4. From top to bottom and left to right; (a) a natural image from the Berkeley database (number 134,052), (b) its soft edge map and (c) and (d) the sets of connected pixels (i.e., regions) whose edge potential is respectively below $\xi = 0.25$ and $\xi = 0.35$. These regions are represented by colored regions. The white region corresponds to the sets of pixels whose edge potential is above the threshold (and thus corresponding to inhomogeneous regions in the textural sense). (e) and (f) results of the final segmentation using their respective map of homogeneous textured regions.

imposed during the iterative K-means labeling process. To this end, we first compute an edge map in the textural sense (see Fig. 4 and the estimation of this edge map will be explained in the following subSection 2.3). The most likely regions in this edge map are easily estimated by identifying the sets of connected pixels whose edge potential is below a given (and low) threshold ξ. In this way, we easily identify regions in which only low textural gradient magnitudes have been detected. This procedure allows us to define a map of (likely) homogeneous textural regions. The hard constraint enforcing the spatial contiguity of each of the K-means cluster is then simply done by assigning the majority class label in each (pre-estimated) homogeneous textural region, for each iteration of the K-means algorithm. This spatially constrained K-means is ensured after a few iterations (e.g. typically 10 iterations in our application) of the classical K-means clustering. Implicitly, this procedure allows to consider non-spherical clusters in the K-means clustering scheme since the distribution of each textural features are no more spherical after the spatial constraint.

2.3. Edge map

In order to estimate an edge map (in the textural sense), which will then be used to find the most likely textured regions and then to constrain the K-means procedure, we compute, at each pixel, the following distance

$$\mathcal{D}\left(h_{i-(N_0/2),j}, h_{i+(N_0/2),j}\right) + \mathcal{D}\left(h_{i,j-(N_0/2)}, h_{i,j+(N_0/2)}\right)$$

where h is the N_b-bin (descriptor) vector, i.e., the re-quantized local color histogram, located at row i and column j (see Section 2.1 and Algorithm 1), and $\mathcal{D}(h_{i-d,j}, h_{i+d,j})$ is the L_1 norm between vectors (or bin descriptors) $h_{i-d,j}$ and $h_{i+d,j}$ computed on a squared N_0-size window centered respectively at location $(i-d,j)$ and $(i+d,j)$. In order to propose a reliable edge map, we have averaged the different edge maps obtained for the same input image but expressed in different color spaces (and normalized this resulting and averaged potential map between the interval $[0-1]$). Fig. 4 shows an example of a soft edge map and two sets of regions obtained by two different threshold values ξ. The spatial constraints of the K-means procedure consists in assigning, at each iteration of the K-means segmentation map, the majority class label existing in each colored region. Let us point out that the white regions shown in Fig. 4(c)–(d) are not exploited in these spatial constraints since they represent inhomogeneous regions in the textural sense, thus corresponding to the sets of pixels whose edge potential is above the threshold value ξ. The spatially-constrained K-means segmentation algorithm is outlined below:

(1) Randomly choose K_1 initial cluster centers $c_1^{[1]}, \ldots, c_{K_1}^{[1]}$.
(2) At the kth step, assign sample $\boldsymbol{x}_m$ to the cluster with the nearest center, i.e., to the cluster i if: $\left\|\boldsymbol{x}_m - c_i^{[k]}\right\| < \left\|\boldsymbol{x}_m - c_j^{[k]}\right\|$, $\forall j \neq i$.
(3)
  (a) For each region $\mathcal{R}_i$ (defined by the set of connected pixels whose edge potential is below ξ) of the edge map:
  (b) Find the majority cluster of samples $\boldsymbol{x}_m \in \mathcal{R}_i$, i.e., the cluster l if: $c_l^{[k]} = \arg\max_{c_l^{[k]}} \sum_{\boldsymbol{x}_m \in \mathcal{R}_i} \sum_{1 \leqslant j \leqslant K_1,\ j \neq l} \mathcal{I}\left\{\left\|\boldsymbol{x}_m - c_l^{[k]}\right\| < \left\|\boldsymbol{x}_m - c_j^{[k]}\right\|\right\}$ where $\mathcal{I}$ is the indicator function.
  (c) Assign the majority cluster l to each sample $\boldsymbol{x}_m$ belonging to $\mathcal{R}_i$.
(4) Let $C_i^{[k]}$ denote the i-th cluster with n_i samples after step 3c. Determine new cluster centers by the mean of the samples in the cluster $c_i^{[k+1]} = (1/n_i) \sum_{\boldsymbol{x}_m \in C_i^{[k]}} \boldsymbol{x}_m$
(5) Repeat until convergence is achieved (i.e., until $c_i^{[k+1]} = c_i^{[k]}, \forall i$).
(6) Fuse each small region (i.e., set of connected pixels belonging to the same cluster whose size is below 300 pixels) with the neighboring region sharing its longest boundary.

3. Experimental results

3.1. Set up

In all the experiments, in order to promote diversity in the K-means segmentation results or in the different estimated edge maps (and also to somewhat increase the accuracy around the boundaries between distinct texture regions, after the averaging process), we have considered ten ($N_s = 10$) different color spaces (each color channel has been normalized between 0 and 255), namely RGB, HSV, YIQ, XYZ, LAB, LUV, $I_1I_2I_3$, $H_1H_2H_3$, YC_bC_r, TSL, (Martinkauppi et al., 2001; Banks, 1990; Braquelaire and Brun, 1997). For all K-means-based segmentations (first and second step), the L_1 norm is used as clustering distance measure (experiments have shown that the L_1 norm gives the same performance measure as the euclidean norm while consuming less computational resources in our application).

- For the K-means based-de-texturing approach and the estimation of the edge map, the number of bins for each local re-quantized color histogram, the number of classes, and the size of the overlapping squared window are respectively set to $q = 5$, $K_0 = 5$ and $N_0 = 7$.
- For the spatially-constrained K-means clustering used in the second step, on the de-textured color image, we have used as input multidimensional feature descriptor, the set of color values estimated on an overlapping squared fixed-size ($N_1 = 5$) neighborhood centered around each pixel to be classified and finally $K_1 = 9$ classes.

In order to obtain a more reliable color segmentation, we have noticed that better segmentation results are achieved if we consider, in the feature descriptor vector, all the color values expressed by each color space. The feature extraction step thus yields to a $[N_1 \times N_1 \times 3 \times 10]$-dimensional feature vector (i.e., number of pixels in the squared overlapping sliding window $\times$ 3 color channels $\times$ 10 color spaces). A final merging step is added to each segmentation map that simply consists of fusing each small region (i.e., regions whose size is below 300 pixels) with the region sharing its longest boundary.

In these experiments, we tested our segmentation algorithm on the Berkeley segmentation database (Martin et al., 2001) consisting of 300 color images of size 481×321 (beforehand divided, by their creators, into a disjoint training and test sets of respectively 200 and 100 images). For each color image, a set of benchmark segmentation results, provided by human observers is available and will be used to quantify the reliability of the proposed segmentation algorithm. In order to ensure the integrity of the evaluation, the internal parameters of the algorithm are tuned on the train image set. The algorithm is then bench-marked by using the optimal training parameters on the independent test set.

3.2. Training

Experiments have shown that our overall segmentation procedure is relatively sensitive to two parameters, namely, in order of importance, the threshold value ξ used to spatially constrain the K-means procedure used on the de-textured color image and, to a very lesser degree, the number of classes K_1 of the final clustering. These two parameters are thus tuned on the train image set

Table 1
Average performance, in term of PRI measure, of our algorithm for the optimal training set of its two internal parameters on the independent test set of the Berkeley image database (Martin et al., 2001) and comparison with other algorithms: (Unnikrishnan et al., 2005; Yang et al., 2008; Ilea and Whelan, 2008; Krinidis and Pitas, 2009; Mignotte, 2008; Felzenszwalb and Huttenlocher, 2004; Hedjam and Mignotte, 2009; Deng and Manjunath, 2001; Ma et al., 2007; Sfikas et al., 2008; Comaniciu and Meer, 2002; Shi and Malik, 2000). The first value is the performance metric on the entire database and values between square brackets correspond to performances on the train and test image sets.

ALGORITHMS	**PRI** (Unnikrishnan et al., 2005)
HUMANS in (Yang et al., 2008)	0.8754
SCKM$_{[K_1=9\|\xi=0.37]}$	**0.796** [train ▷ 0.803 test ◁ **0.782**]
CTex (Ilea and Whelan, 2008)	0.800
MIS$_{[\lambda=50]}$ (Krinidis and Pitas, 2009)	0.798
FCR[K_1=13|K_2=6|κ=0.135] (Mignotte, 2008)	0.788
FH (Felzenszwalb and Huttenlocher, 2004) in (Yang et al., 2008)	0.784
HMC (Hedjam and Mignotte, 2009)	0.783
JSEG (Deng and Manjunath, 2001) in (Ilea and Whelan, 2008)	0.770
CTM$_{\eta=0.15}$ (Yang et al., 2008; Ma et al., 2007)	0.762
St-SVGMM$_{[K=15]}$ (Sfikas et al., 2008)	0.759
Mean-Shift (Comaniciu and Meer, 2002) in (Yang et al., 2008)	0.755
NCuts (Shi and Malik, 2000) in (Yang et al., 2008)	0.722

by doing a local discrete grid search routine,[2] with a fixed step-size, on the parameter space and in the feasible ranges of parameter values (namely $\xi \in [0.0 - 1.0]$ [step-size: 0.01], $K_1 \in [4 - 10]$ [step-size: 1]). The best couple of parameter is obtained for [$K_1 = 9|\xi = 0.37$].

3.3. Comparison with state-of-the-art methods

We have compared our segmentation algorithm (called SCKM$_{[K_1|\xi]}$ for Spatially Constrained K-Means, K_1 and ξ being its two internal parameters) against several unsupervised algorithms. For each of these algorithms, the internal parameters are set to optimal values and/or correspond to the internal values suggested by the authors. All color images are normalized to have the longest side equals to 320 pixels. The segmentation results are then supersampled in order to obtain segmentation images with the original resolution (481×321) before the estimation of the performance metric.

The comparison is based on the PRI[3] performance measure (Unnikrishnan et al., 2005) which seems to be also highly correlated with human hand-segmentations (Yang et al., 2008) (a score equal to PRI = 0.796, for example, simply means that, on average, 79.6% of pairs of pixel labels are correctly classified in the segmentation results).

Table 1 shows the obtained PRI results for the different algorithms. In terms of PRI measure, we observe that the discussed fusion strategy gives competitive results over the set of images of the Berkeley image database. Without the spatial constraint (i.e., with $\xi = 0$, and $K_1 = 9$), we obtain about PRI = 0.754. We can thus notice than a good value of ξ allows to significantly improve the segmentation result. Consequently, it is really necessary to help the optimization process to succeed in finding an optimal solution. To this end, the hard constraint enforcing the spatial contiguity is useful in this very difficult segmentation method. Fig. 5 shows the distribution of the PRI measure over the 300 images of the Berkeley image database (for the algorithm SCKM$_{[K_1=9|\xi=0.37]}$).

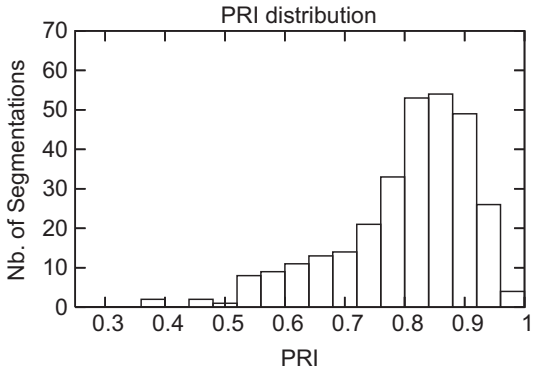

Fig. 5. Distribution of the PRI performance measure over the 300 images of the Berkeley database (for SCKM$_{[K_1=9|\xi=0.37]}$).

Figs. 6 and 7 display some examples of segmentations obtained by our algorithm. The results for the entire database are available on-line at http address www.iro.umontreal.ca/~mignotte/ResearchMaterial/sckm.html. While being simple, this segmentation procedure gives excellent segmentation results with a very competitive PRI score (which indicates that, on average, 79.6% of pairs of pixels are correctly classified on the entire Berkeley database and 78.2% on the test base).

The segmentation procedure takes, on average, less than 20 s per image (for an AMD Athlon 64 Processor 3500+, 2.2 GHz, 4435.67 bogomips and non-optimized code running on Linux) for the entire segmentation procedure. Table 2 compares the average computational time for an image segmentation and for different segmentation algorithms whose PRI is greater than 0.76.

3.4. Sensitivity to parameters

We have tested the influence of the final merging procedure and its threshold on the final PRI score. Without the final merging procedure (i.e., with a threshold equals to zero), we obtain a PRI equals to 0.789 (on the entire Berkeley database) and by considering a threshold twice greater, i.e., by merging each region whose size is below 600 pixels, with the region sharing its longest border, we obtain a PRI equals to 0.795. These experiments thus show that our segmentation model tends to slightly over-segment and that our model is not too sensitive to this parameter to the extent that it is around 300. We have also tested and quantified the influence of the variation of the parameters N_0, N_1, q, K_0, K_1 and ξ as respectively a function of the number of pixels, bins or classes. Fig. 8 shows the evolution of the PRI score for several discrete values of these parameters (holding all other parameters constant at their optimal values). Experiments show that our model is not too sensitive to parameters N_0, N_1, K_0 and q but sensitive to K_1 and ξ and this justifies the learning phase on these two parameters. Let us also note that for $N_1 = 1$, we obtain a PRI score equals to 0.792. This shows that this parameter is not essential (i.e., it is not essential to also take the color value of the neighbors of the pixel to be classified), although $N_1 = 5$ allows to "regularize" a bit the K-means procedure on the de-textured color image and to get a slightly better PRI score.

[2] The grid search routine is a standard optimization algorithm which consists, for several initial guesses of the parameter vector to be estimated, in employing a moving n-dimensional (n is the number of parameters and $n = 2$ in our application) grid containing three values per parameter low, medium, and high with spacing determined by step-size. The algorithm tries to center the grid around the best performance score for each dimension (parameter), moving in an appropriate direction during each iteration. The optimization is successful when the grid becomes centered on the best performance score across all dimensions.

[3] We have used the Matlab source code, proposed by Allen Y. Yang in order to estimate the PRI performance measure presented in the following Section. This code is kindly available on-line at address http://www.eecs.berkeley.edu/~yang/software/lossy_segmentation/.

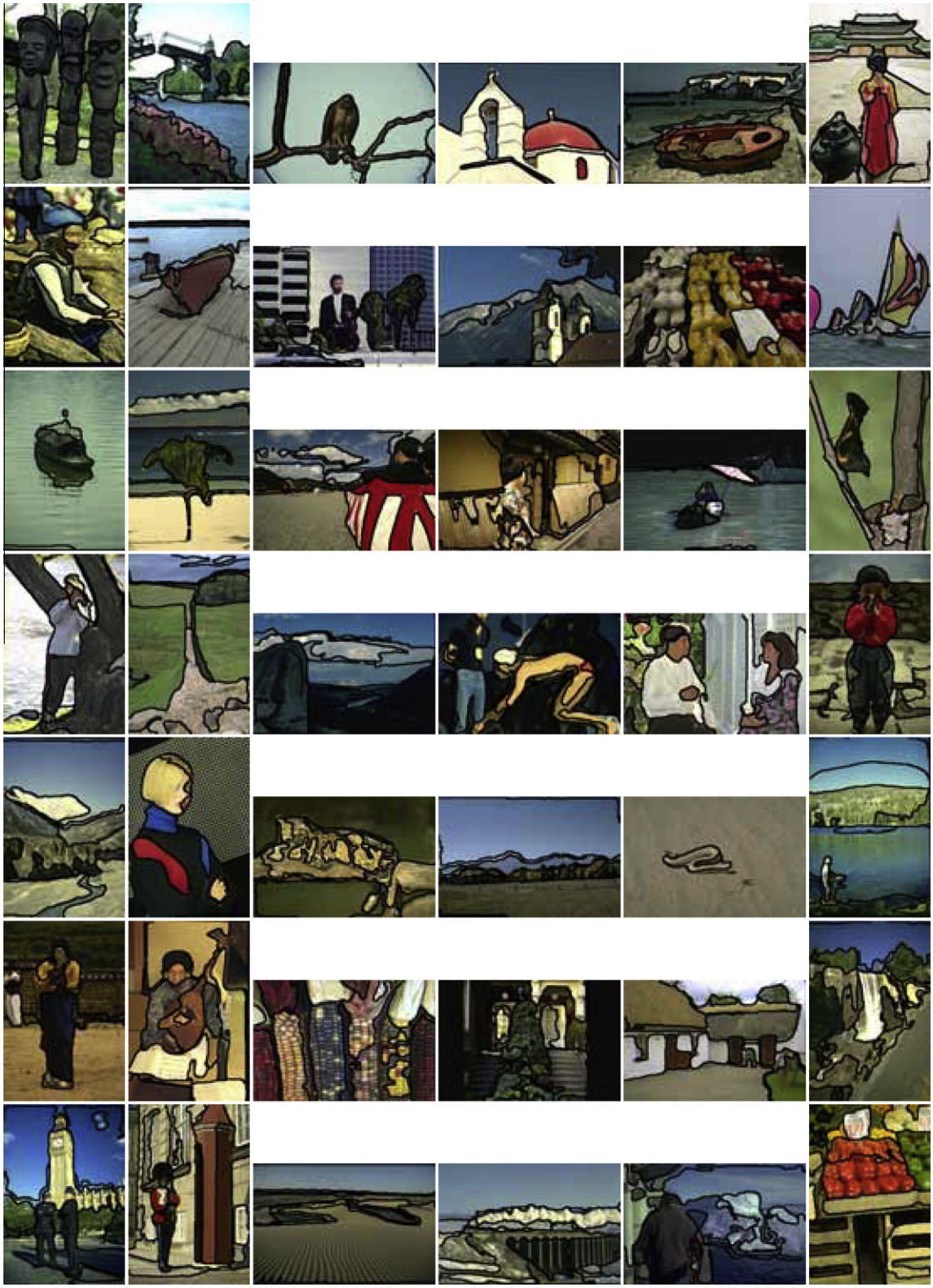

Fig. 6. Example of segmentations obtained by our algorithm $SCKM_{|K_1=9|\xi=0.37|}$ on several images of the Berkeley image database (see also Table 1 and Fig. 5 for quantitative performance measures and http://www.iro.umontreal.ca/~mignotte/ResearchMaterial/sckm.html for the segmentation results on the entire database).

We have also tested the influence and efficiency of our de-texturing procedure in our segmentation method. To this end, we have used the proposed spatially constrained K-means on the original input image, instead of the de-textured image (with K_1, ξ, q and N_0 keeping their optimal values). We obtain a PRI score equals to 0.773. This shows that our de-texturing helps the K-means clustering process to succeed in finding a better solution of segmentation, by *simplifying* the textural data to be segmented and/or by helping the K-means to converge toward a better local minima.

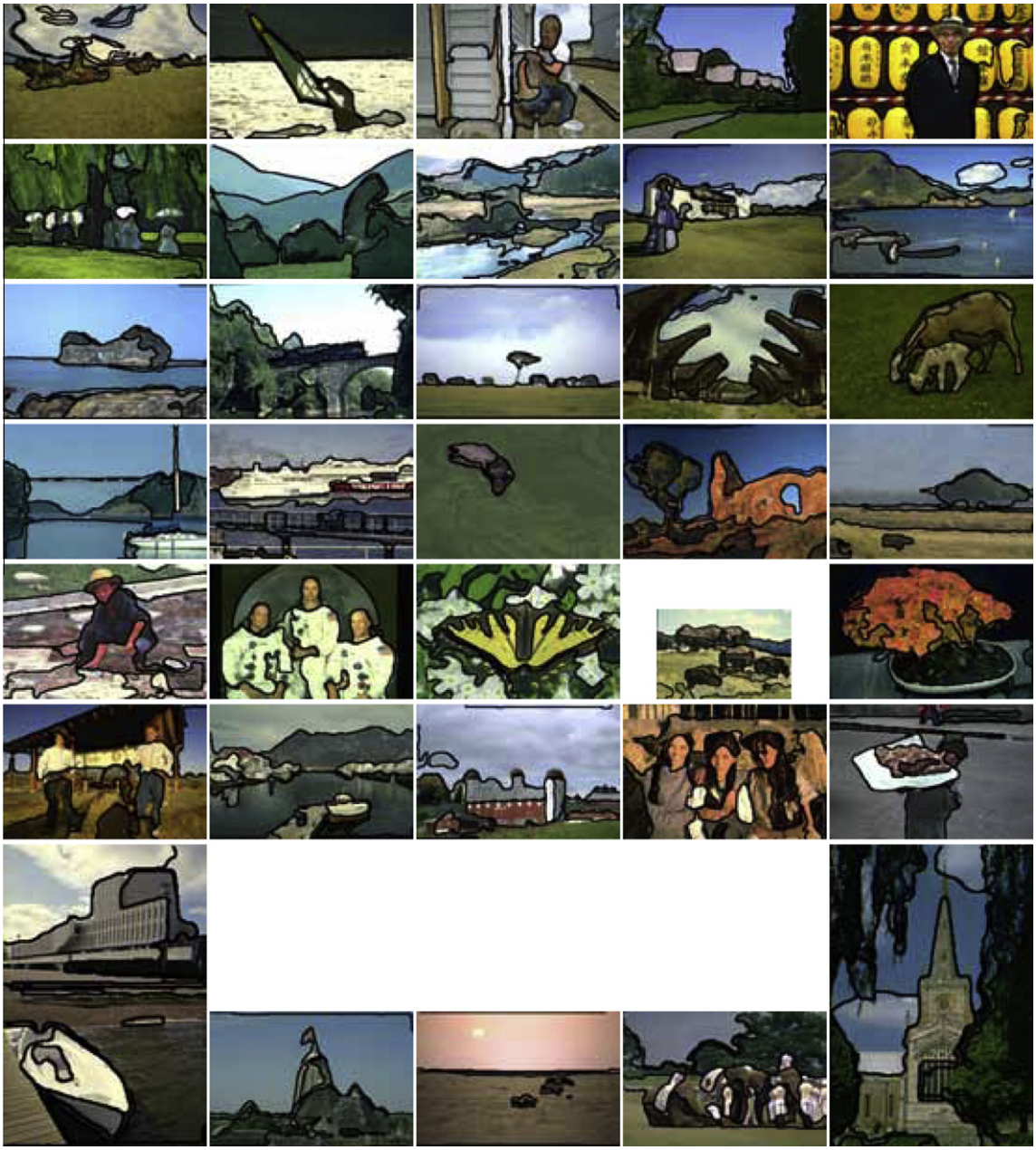

Fig. 7. Example of segmentations obtained by our algorithm $SCKM_{[K_1=9|\xi=0.37]}$ on several images of the Berkeley image database (see also Table 1 and Fig. 5 for quantitative performance measures and http://www.iro.umontreal.ca/~mignotte/ResearchMaterial/sckm.html for the segmentation results on the entire database).

Table 2
Average CPU time for different segmentation algorithms.

ALGORITHMS	PRI	CPU time (s)	On [image size]	Given in reference
CTex	0.800	≈85	[184 × 184]	Ilea and Whelan (2008)
$MIS_{[\lambda=50]}$	0.798	≈2.9	[320 × 200]	Krinidis and Pitas (2009)
SCKM$_{[K_1=9\|\xi=0.37]}$	0.796	≈20	[320 × 200]	
$FCR_{[K_1=13\|K_2=6\|\kappa=0.135]}$	0.788	≈60	[320 × 200]	Mignotte (2008)
FH	0.784	≈1	[320 × 200]	Felzenszwalb and Huttenlocher (2004)
HMC	0.783	≈80	[320 × 200]	Hedjam and Mignotte (2009)
JSEG	0.770	≈6	[184 × 184]	Ilea and Whelan (2008)
$CTM_{\eta=0.15}$	0.762	≈180	[320 × 200]	Yang et al. (2007)

We have shown, in Fig. 9, the four worst segmentations (in the PRI sense), obtained by our segmentation method. Experiments show that our method tends to over-segment some images containing regions with large texture elements or sometimes to merge a part of an animal (with its texture camouflage) with its natural environment (or more generally a textured region with the same average color of its background). In this latter case, this merging may be due to the fact that two different textured regions, but with

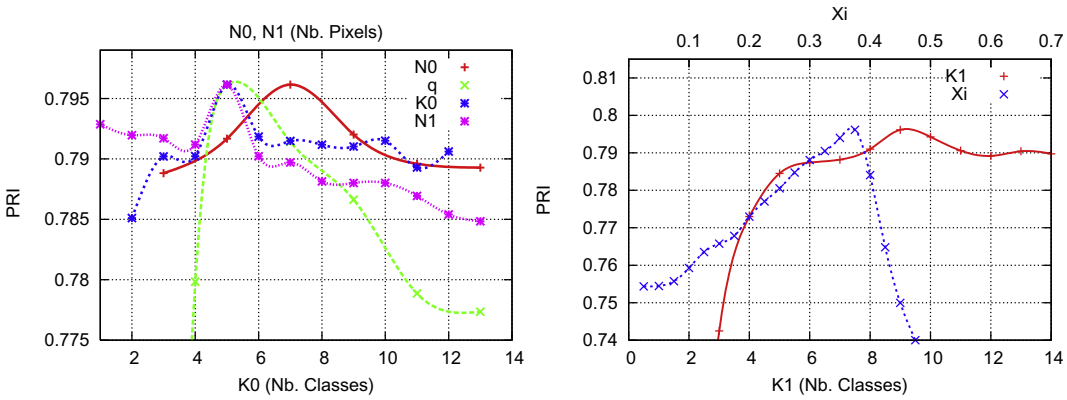

Fig. 8. Left: evolution of the PRI for respectively: [1] N_0 and N_1 as a function of the number of pixels, [2] q as a function of the number of bins, [3] K_0 as a function of the number of classes. Right: evolution of the PRI for respectively: K_1 and ξ.

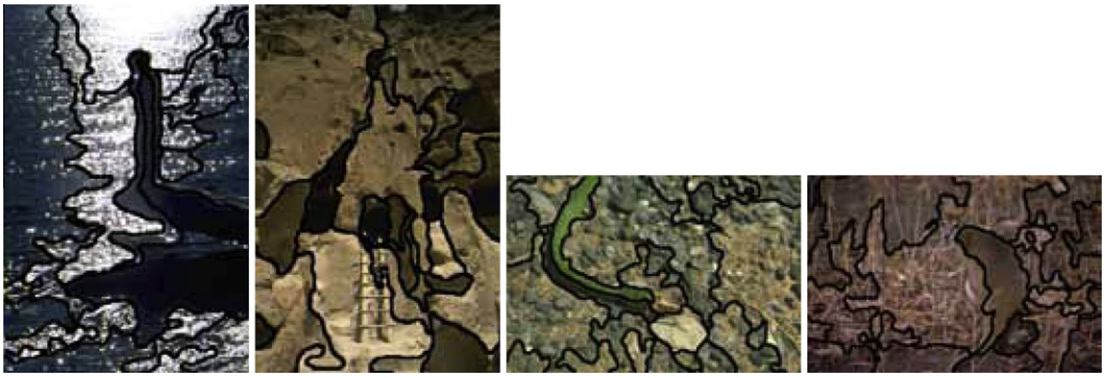

Fig. 9. The four Berkeley images associated with the four worst PRI scores obtained by our segmentation method $(\mathrm{SCKM}_{[K_1=9|\xi=0.37]})$, from left to right, image number 26,031 (PRI = 0.393), 54,005 (PRI = 0.459), 175,043 (PRI = 0.370), 69,040 (PRI = 0.438).

the same average color, will be assigned the same color value after the de-texturing step. To some extent, the spatial constraints of the final K-means procedure on the de-textured image, which is computed with the original textural cues (based on local histogram of each region) of the input image, is supposed to counterbalance a bit this. Nevertheless, two textured regions, with almost identical local histograms and two identical average colors will be merged by our segmentation method. It seems that this is the main drawback of our segmentation method which could be improved on this point.

We also present, in Fig. 10, some segmentation results for increasing values of N_0. Although we have previously shown that our segmentation model is less sensitive to N_0 (comparatively to ξ and K_1), this parameter, however, seems to be an important internal parameter of our segmentation algorithm because it is related to the accuracy of the detected boundary location. Indeed, experiments show that this parameter must be large enough to fully model the texture elements (i.e., the so-called *texton*) but should not be too large in order not to wrongly detect the boundaries, between distinct textured regions, as an existing region (and thus to affect the accuracy of the boundary location). A good compromise, between good classification of the segmentation and contour accuracy, seems to be the value that we gave, i.e., $N_0 = 7$. In order to quantify this contour accuracy, we have computed the Boundary Displacement Error (BDE) measure (lower distance is better) proposed by Freixenet et al. (2002) on the entire Berkeley database. The BDE measures the average displacement error of boundary pixels between two segmented images. Particularly, it defines the error of one boundary pixel as the distance between the pixel and the closest pixel in the other boundary image. For our algorithm, we obtain for $N_1 = 1$ and $N_1 = 5$, respectively, BDE = 9.8 and BDE = 10.0 compared to 9.9 for the algorithm called CTM and 10.0 for the algorithm called FH (results available in (Yang et al., 2008)), which, in terms of contour accuracy is comparable.

Fig. 10. Segmentation results for increasing values of N_0. From left to right, N_0 is equal to 3, 7, 15 and 3, 7 (holding all other parameters constant at their optimal values).

Let us finally add that, ideally, in order to get a more efficient segmentation algorithm, N_0, ξ and K_1 should be automatically estimated for each image.

3.5. Algorithm

The source code (in C++ language under Linux) of our algorithm (with the set of segmented images) are publicly available at the following http address www.iro.umontreal.ca/∼mignotte/ResearchMaterial/sckm.html in order to make possible eventual comparisons with future segmentation algorithms or different performance measures.

4. Conclusion

In this paper, we have presented an original, simple and efficient segmentation procedure based on the *K*-means procedure. The proposed method overcomes the inherent problems of this simple automatic clustering procedure using spherical distributions on the difficult problem of the textured color image segmentation. First, the clustering is done on a simplified (de-textured image), on which the search of well-separated clusters is easier. Besides, some spatial constraints help the iterative *K*-means labeling process to succeed in finding an accurate segmentation by taking into account the inherent spatial relationships and the presence of pre-estimated homogeneous textural regions in the input image. Implicitly, this procedure allows to consider non-spherical clusters in the *K*-means clustering scheme since the distribution of each textural feature vector is no more spherical after the spatial constraint. While being simple to implement or to understand and relatively fast compared to the best existing algorithms, the proposed segmentation procedure performs competitively among the recently reported state-of-the-art segmentation methods.

References

Banks, S., 1990. Signal Processing, Image Processing and Pattern Recognition. Prentice Hall.

Berkhin, P., 2002. Survey of Clustering Data Mining Techniques. Technical Report, Accrue Software, San Jose, CA.

Braquelaire, J.-P., Brun, L., 1997. Comparison and optimization of methods of color image quantization. IEEE Trans. Image Process. 6, 1048–1952.

Comaniciu, D., Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Machine Intell. 24 (5), 603–619.

Deng, Y., Manjunath, B.S., 2001. Unsupervised segmentation of color-texture regions in images and video. IEEE Trans. Pattern Anal. Machine Intell. 23 (8), 800–810.

Felzenszwalb, P., Huttenlocher, D., 2004. Efficient graph-based image segmentation. Internat. J. Comput. Vision 59, 167–181.

Freixenet, J., Munoz, X., Raba, D., Marti, J., Cufi, X., 2002. Yet another survey on image segmentation: Region and boundary information integration. In: ECCV02, vol. III, p. 408.

Ilea, D.E., Whelan, P.F., 2008. Ctex – An adaptive unsupervised segmentation algorithm on color-texture coherence. IEEE Trans. Image Process. 17 (10), 1926–1939.

Krinidis, M., Pitas, I., 2009. Color texture segmentation based on the modal energy of deformable surfaces. IEEE Trans. Image Process. 7 (18), 1613–1622.

Lloyd, S.P., 1982. Least squares quantization in PCM. IEEE Trans. Inf. Theory 28 (2), 129–136.

Ma, Y., Derksen, H., Hong, W., Wright, J., 2007. Segmentation of multivariate mixed data via lossy coding and compression. IEEE Trans. Pattern Anal. Machine Intell. 29 (9), 1546–1562.

Martin, D., Fowlkes, C., Tal, D., Malik, J., 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. Eighth Internat. Conf. on Computer Vision, vol. 2, pp. 416–423.

Martinkauppi, J.B., Soriano, M.N., Laaksonen, M.H., 2001. Behavior of skin color under varying illumination seen by different cameras at different color spaces. In: Proc. SPIE, Mach. Vision Appl. Ind. Inspection IX, San Jose, California, pp. 102–113.

Mignotte, M., 2007. Image denoising by averaging of piecewise constant simulations of image partitions. IEEE Trans. Image Process. 16 (2), 523–533.

Mignotte, M., 2008. Segmentation by fusion of histogram-based *k*-means clusters in different color spaces. IEEE Trans. Image Process. 17 (5), 780–787.

Hedjam, R., Mignotte, M., 2009. A hierarchical graph-based Markovian clustering approach for the unsupervised segmentation of textured color images. In: 16th IEEE Internat. Conf. on Image Process, ICIP'09, Cairo, Egypt, pp. 1365–1368.

Sfikas, G., Nikou, C., Galatsanos, N., 2008. Edge preserving spatially varying mixtures for image segmentation. In: Proc. IEEE Internat. Conf. on Computer Vision and Pattern Recognition, Anchorage, AK, USA, vol. 1, pp. 1–7.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Machine Intell. 22, 888–905.

Unnikrishnan, R., Pantofaru, C., Hebert, M., 2005. A measure for objective evaluation of image segmentation algorithms. In: Proc. 2005 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'05), Workshop on Empirical Evaluation Methods in Computer Vision, vol. 3, pp. 34–41.

Yang, A.Y., Wright, J., Sastry, S., Ma, Y., 2008. Unsupervised segmentation of natural images via lossy data compression. Comput. Vision and Image Understanding 110 (2), 212–225.