Color Image Segmentation

*Yining Deng, B. S. Manjunath and Hyundoo Shin**

Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560
*Samsung Electronics Inc.
{deng, manj, hdshin}@iplab.ece.ucsb.edu

Abstract

In this work, a new approach to fully automatic color image segmentation, called JSEG, is presented. First, colors in the image are quantized to several representing classes that can be used to differentiate regions in the image. Then, image pixel colors are replaced by their corresponding color class labels, thus forming a class-map of the image. A criterion for "good" segmentation using this class-map is proposed. Applying the criterion to local windows in the class-map results in the "J-image", in which high and low values correspond to possible region boundaries and region centers, respectively. A region growing method is then used to segment the image based on the multi-scale J-images. Experiments show that JSEG provides good segmentation results on a variety of images.

1. Introduction

Color image segmentation is useful in many applications. From the segmentation results, it is possible to identify regions of interest and objects in the scene, which is very beneficial to the subsequent image analysis or annotation. Recent work includes a variety of techniques: for example, stochastic model based approaches [1], [4], [8], [11], [12], morphological watershed based region growing [9], energy diffusion [7], and graph partitioning [10]. Quantitative evaluation methods have also been suggested [2]. However, due to the difficult nature of the problem, there are few automatic algorithms that can work well on a large variety of data.

The problem of segmentation is difficult because of image texture. If an image contains only homogeneous color regions, clustering methods in color space such as [3] are sufficient to handle the problem. In reality, natural scenes are rich in color and texture. It is difficult to identify image regions containing color-texture patterns. The approach taken in this work assumes the following:

- Each region in the image contains a uniformly distributed color-texture pattern.
- The color information in each image region can be represented by a few quantized colors, which is true for most color images of natural scenes.
- The colors between two neighboring regions are distinguishable - a basic assumption of any color image segmentation algorithm.

This work was supported in part by a grant from Samsung Electronics Inc.

The main contribution of this paper are the following:

- We introduce a new criterion for image segmentation (Section 3). This criterion involves minimizing a cost associated with the partitioning of the image based on pixel labels. The pixel labels are derived from color quantization, as explained in Section 2, and extensions to other image features are possible.
- A practical algorithm, called JSEG, is suggested towards achieving this segmentation objective. The notion of "*J*-images" is introduced in Section 3. *J*-images correspond to measurements of local image inhomogeneities at different scales. The *valleys* in the image correspond to homogeneous regions and the *peaks* correspond to potential boundary locations. A spatial segmentation algorithm is then described in Section 4, which grows regions from the valleys of the *J*-images to achieve segmentation.

Fig. 1 shows a schematic of the JSEG algorithm.

2. Criterion for Segmentation

First, colors in the image are coarsely quantized without significantly degrading the color quality. The purpose is to extract a few representing colors that can be used to differentiate neighboring regions in the image. Typically, 10-20 colors are needed in the images of natural scenes. A good color quantization is important to the segmentation process. A perceptual color quantization algorithm [5] is used in our implementation.

After quantization, the quantized colors are assigned labels. A color class is the set of image pixels quantized to the same color. The image pixel colors are replaced by their corresponding color class labels. The new constructed image of labels is called a class-map. Examples of class-

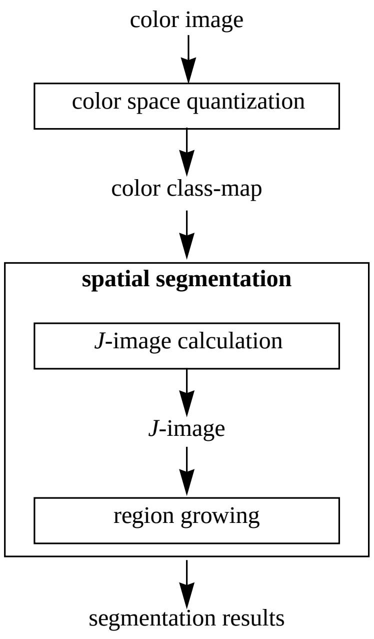


Fig 1. Schematic of the JSEG algorithm.

map are shown in Fig. 2, where label values are represented by three symbols, ‘*’, ‘+’, and ‘o’. The necessary color information for segmentation is extracted and stored in a simple class-map after color quantization. Usually, each image region contains pixels from a small subset of the color classes and each class is distributed in a few image regions.

The class-map can be viewed as a set of spatial data points located in a 2-D plane. The value of each point is the image pixel position, a 2-D vector *(x, y)*. These data points have been classified and each point is assigned a label, which is the value of the class-map at that image position. In the following, a criterion for “good” segmentation using these spatial data points is proposed.

Before proceeding further, let us first consider the measure *J* defined as follows. Let *Z* be the set of all *N* data points in the class-map. let $z = (x, y)$, $z \in Z$, and *m* be the mean,

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

Suppose *Z* is classified into *C* classes, Z_i, $i = 1, \dots, C$. Let m_i be the mean of the N_i data points of class Z_i,

$$m_i = \frac{1}{N_i}\sum_{z \in Z_i} z \tag{2}$$

Let

$$S_T = \sum_{z \in Z} \|z - m\|^2 \tag{3}$$

and

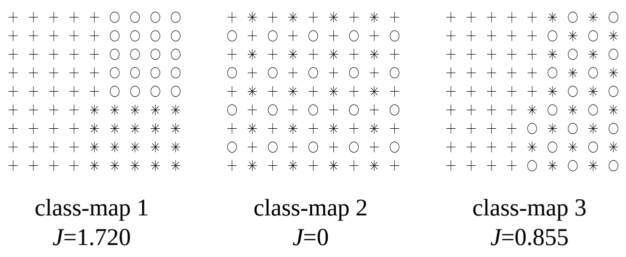


Fig 2. An example of different class-maps and their corresponding *J* values. ‘+’, ‘o’, and ‘*’ indicate three classes of data points.

$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C}\sum_{z \in Z_i} \|z - m_i\|^2 \tag{4}$$

The measure *J* is defined as

$$J = S_B / S_W = (S_T - S_W) / S_W \tag{5}$$

It essentially measures the distances between different classes S_B over the distances between the members within each class S_W, a similar idea as the Fisher’s multi-class linear discriminant [6], but for arbitrary nonlinear class distributions. A higher value of *J* indicates that the classes are more separated from each other and the members within each class are closer to each other, and vice versa.

For the case when an image consists of several homogeneous color regions, the color classes are more separated from each other and the value of *J* is large. On the other hand, if all color classes are uniformly distributed over the entire image, the value of *J* tends to be small. Most of the time, the value of *J* is somewhere in between. For example, in Fig. 2, three class-maps are shown, which correspond to the three cases mentioned above. There are three classes in each map and the number of points in each class is the same for all three maps. Notice that the *J* values are significantly different for these three cases.

Consider class-map 1 from Fig. 2, a “good” segmentation for this case would be three regions each containing a single class of data points. Class-map 2 is uniform by itself and no segmentation is needed. For class-map 3, a “good” segmentation would be two regions. One region contains class ‘+’ and the other one contains classes ‘*’ and ‘o’. The segmentation of class-map 1 and 3 is shown in Fig. 3.

Now let us recalculate *J* over each segmented region instead of the entire class-map and define the average $\bar{J}$ by

$$\bar{J} = \frac{1}{N}\sum_{k} M_k J_k \tag{6}$$

where J_k is *J* calculated over region *k*, M_k is the number of points in region *k*, *N* is the total number of points in the class-map, and the summation is over all the regions in the

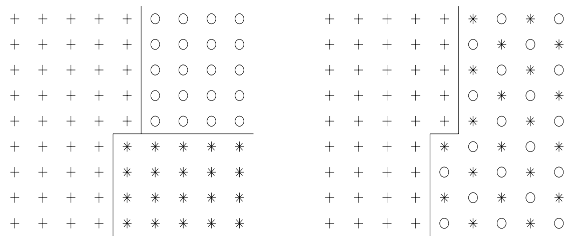


Fig 3. Segmented class-maps and their corresponding $\bar{J}$ values.

class-map. Note that J can be considered as a special case of $\bar{J}$ where there is only one segmented region.

We now propose $\bar{J}$ as the criterion to be minimized over all possible ways of segmenting the image. For a fixed number of regions, "better" segmentation tends to have a lower value of $\bar{J}$. If the segmentation is good, each segmented region contains a few uniformly distributed color classes and the resulting J value for that region is small. Therefore, the overall $\bar{J}$ is also small.

The values of $\bar{J}$ calculated for class-map 1 and 3 are shown in Fig. 3. It is clear in the case of class-map 1 that any other ways of segmenting the map into three regions will have a $\bar{J}$ value larger than the current one because $\bar{J}$ is non-negative. Same is true for the case of class-map 3 because if there are a large number of points in the map, $J_{\{*,o\}}$ is also approximately equal to 0. Intuitively, suppose the segmentation boundary is altered such that one region is added an extra area from another region. The J value for the region being subtracted remains 0, while the J value for the other region is increased due to the added impurity. Thus, the new segmentation will result in an increased value of $\bar{J}$.

3. *J*-images

The global minimization of $\bar{J}$ for the entire image is not practical. However, notice the fact that J, if applied to a local area of the class-map, is also a good indicator of whether that area is in the region center or near region boundaries. We now introduce the notion of a J-image:

***J*-image** The J-image is a gray-scale image whose pixel values are the J values calculated over local windows centered on these pixels.

In the rest of the paper, these J values will be referred as local J values. The higher the local J value is, the more likely that the pixel is near region boundaries. The J-image is like a 3-D terrain map containing valleys and mountains that actually represent the region centers and region boundaries, respectively.

The size of the local window determines the size of image regions that can be detected. Windows of small size are useful in localizing the intensity/color edges, while large windows are useful for detecting texture boundaries. Often, multiple scales are needed to segment an image. In our implementation, the basic window at the smallest scale is a 9 x 9 window without corners, as shown in Fig. 4 (a). The corners are removed to make the window more circular such that the choice of the window does not have any bias towards rectangular objects. The smallest scale is denoted scale 1. From scale 1, the window size is doubled each time to obtain the next larger scale as listed in Table 1.

For computational reasons, successive windows are downsampled appropriately. Fig. 4 (b) shows the window at twice the basic size or scale 2, where the sampling rate is 1 out of 2 pixels along both x and y directions. Fig. 6 (a) shows an original image from the "flower garden" video sequence (frame 0). The J-images at scale 3 and 2 are shown in (c) and (d).

Table 1: Window Sizes at Different Scales

scale	window (pixels)	sampling (1 / pixels)	region size (pixels)	min. valley (pixels
1	9 x 9	1 / (1 x 1)	64 x 64	32
2	17 x 17	1 / (2 x 2)	128 x 128	128
3	33 x 33	1 / (4 x 4)	256 x 256	512
4	65 x 65	1 / (8 x 8)	512 x 512	2048

4. Spatial Segmentation Algorithm

The characteristics of the J-images allow us to use a

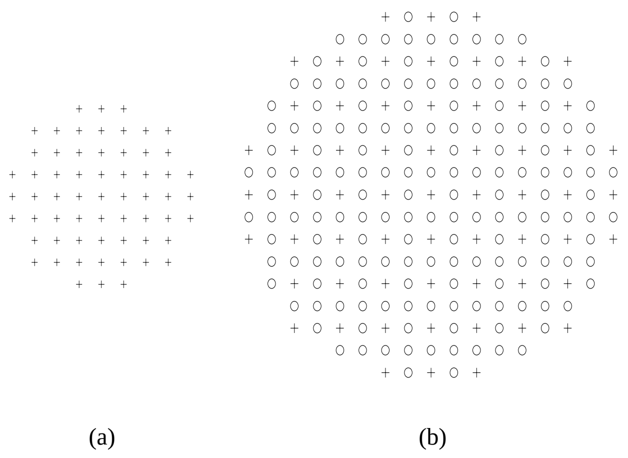


Fig 4. (a) the basic window for calculating local J values. (b) illustration of downsampling for the window at scale 2. Only '+' points are used for calculating local J values, which forms the same basic window as in (a).

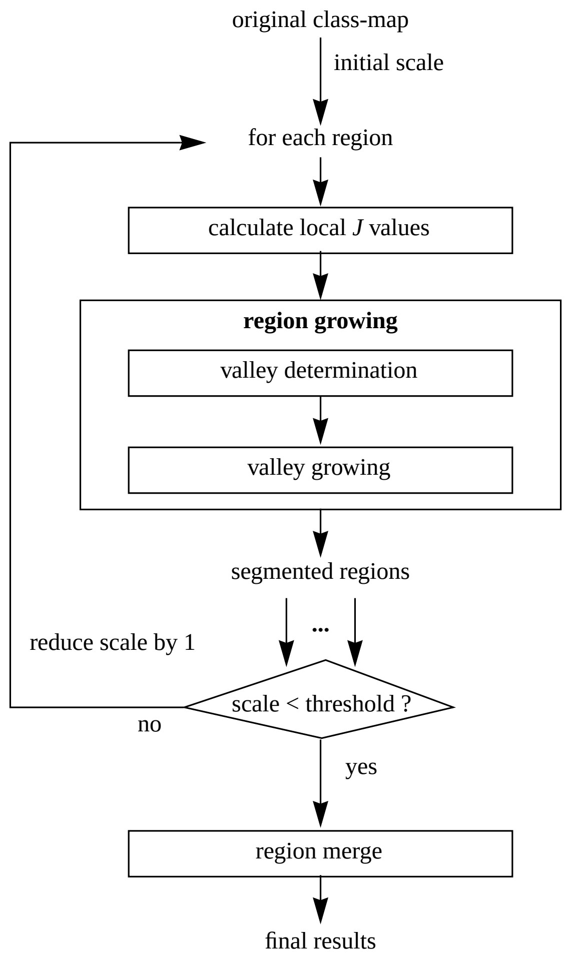


Fig 5. Flow-chart of the steps in spatial segmentation.

region-growing method to segment the image. Fig. 5 shows a flow-chart of the steps in our spatial segmentation algorithm. Consider the original image as one initial region. The algorithm starts segment all the regions in the image at an initial large scale. It then repeats the same process on the newly segmented regions at the next smaller scale until the minimum specified scale is reached.

Table 1 lists a set of scales and suitable region sizes for these scales to be used in the implementation. For example, if the image size is larger than 256 x 256, but smaller than 512 x 512, the starting scale is 3. The user specifies the number of scales needed for the image, which determines the minimum scale that ends the program.

In the actual implementation, local J values are calculated from for each individual region instead of the entire image. The difference between this and the J-image mentioned in Sec. 4 is that near the region boundaries the window is truncated according to the shape of the boundary to avoid the boundary artifacts from the neighboring regions.

4.1 Valley Determination

At the beginning, a set of small initial areas are determined to be the bases for region growing. These areas have the lowest local J values and are called valleys. In general, finding the best set of valleys in a region is a non-trivial problem. The following simple heuristics have provided good results in the experiments:

1. Calculate the average and the standard deviation of the local J values in the region, denoted by μ_J and σ_J respectively.
2. Set a threshold T_J at
$$T_J = \mu_J + a\sigma_J \tag{7}$$
Pixels with local J values less than T_J are considered as candidate valley points. Connect the candidate valley points based on the 4-connectivity and obtain candidate valleys.
3. If a candidate valley has a size larger than the minimum size listed in Table 1 at the corresponding scale, it is determined to be a valley.
4. a is chosen from the set of parameter values $[-0.6, -0.4, -0.2, 0, 0.2, 0.4]$ which gives the most number of valleys.

4.2 Valley Growing

The new regions are then grown from the valleys. It is slow to grow the valleys pixel by pixel. A faster approach is used in the implementation:

1. Remove “holes” in the valleys.
2. Average the local J values in the remaining unsegmented part of the region and connect pixels below the average to form growing areas. If a growing area is adjacent to one and only one valley, it is assigned to that valley.
3. Calculate local J values for the remaining pixels at the next smaller scale to more accurately locate the boundaries. Repeat steps 2.
4. Grow the remaining pixels one by one at the smallest scale. Unclassified pixels at the valley boundaries are stored in a buffer. Each time, the pixel with the minimum local J value is assigned to its adjacent “valley” and the buffer is updated till all the pixels are classified.

4.3 Region Merge

After region growing, an initial segmentation of the image is obtained. It often has over-segmented regions. These regions are merged based on their color similarity. The quantized colors are naturally color histogram bins. The color histogram features for each region are extracted and the distances between these features can be calculated. Since the colors are very coarsely quantized, in our algorithm it is assumed that there are no correlations between the quantized colors. Therefore, a Euclidean distance measure is applied directly.

An agglomerative method [6] is used to merge the regions. First, distances between two neighboring regions are calculated and stored in a distance table. The pair of regions with the minimum distance are merged together. The color feature vector for the new region is calculated and the distance table is updated. The process continues until a maximum threshold for the distance is reached. After merging, the final segmentation results are obtained.

5. Experimental Results

The JSEG algorithm is tested on a variety of images. Fig. 6 and Fig. 7 show the results of the segmentation on the "flower garden" and Corel photo images. Segmented images are dimmed to show boundaries. It can be seen that the results are quite good. Due to the lack of ground truth, however, we are unable to perform any objective evaluation or comparison with other segmentation methods.

Overall, the values of $\bar{J}$ confirm both the earlier discussions and subjective views. Notice that in Fig. 6, the $\bar{J}$ value after segmentation at scale 2 becomes larger that the $\bar{J}$ value at scale 3 because of over-segmentation in certain regions. The final results after merging, however, do achieve the lowest $\bar{J}$ value among all. There are a few exceptional cases where the $\bar{J}$ values are larger for segmented images than for original ones. Notice that the colors in these images are more or less center-symmetric, which results in small $\bar{J}$ values for the original images. Reformulating $\bar{J}$ using the polar coordinates will help in such cases. Despite the incorrect indication of $\bar{J}$ values, however, the segmentation results are still good because of the use of local J values instead.

The JSEG algorithm has 3 parameters that need to be specified by the user. The first one is a threshold for the color quantization process. It determines the minimum distance between two quantized colors [5]. The second one is the number of scales desired for the image as described in Section 4. The last one is a threshold for region merging. These parameters are necessary because of the varying image characteristics in different applications. For example, two scales are needed to give good results on the "flower garden" image in Fig. 6.

The algorithm works well on a variety of images using a fixed set of parameter values. Fig. 7 shows some examples of the 2,500 images processed without any parameter tuning on individual images. Color images of all the results are available on the web (http://vivaldi.ece.ucsb.edu/users/deng/seg).

JSEG also provides good results when applied to grayscale images where intensity values are quantized the same way as the colors. These results are posted on the web due to the page limit. Also shown on the web is an example of detecting illusionary contours by using the original black/white image to be processed directly as the class-map.

6. Conclusions

In this work, a new approach for fully automatic color image segmentation, called JSEG, is presented. The segmentation consists of color quantization and spatial segmentation. A criterion for "good" segmentation is proposed. Applying the criterion to local image windows results in J-images, which can be segmented using a multi-scale region growing method. Results show that JSEG provides good segmentation on a variety of color images.

In our experiments, several limitations are found for the algorithm. One case is when two neighbor regions do not have a clear boundary, for example, the small trees in Fig. 6. Another case is how to handle the varying shades of an object due to the illumination. For example, the big tree trunk in Fig. 6 is segmented into several parts due to the shades. Future research work is on how to solve these problems and improve the results.

7. References

[1] S. Belongie, et. al., "Color- and texture-based image segmentation using EM and its application to content-based image retrieval", *Proc. of ICCV*, p. 675-82, 1998.

[2] M. Borsotti, P. Campadelli, and R. Schettini, "Quantitative evaluation of color image segmentation results", *Pattern Recognition letters*, vol. 19, no. 8, p. 741-48, 1998.

[3] D. Comaniciu and P. Meer, "Robust analysis of feature spaces: color image segmentation", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp 750-755, 1997.

[4] Y. Delignon, et. al., "Estimation of generalized mixtures and its application in image segmentation", *IEEE Trans. on Image Processing*, vol. 6, no. 10, p. 1364-76, 1997.

[5] Y. Deng, C. Kenney, M.S. Moore, and B.S. Manjunath, "Peer group filtering and perceptual color image quantization", to appear in *Proc. of ISCAS*, 1999.

[6] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1970.

[7] W.Y. Ma and B.S. Manjunath, "Edge flow: a framework of boundary detection and image segmentation", *Proc. of CVPR*, pp 744-49, 1997.

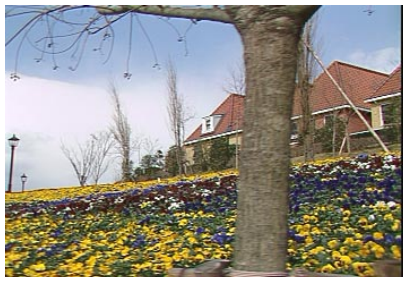
[8] D.K. Panjwani and G. Healey, "Markov random field models for unsupervised segmentation of textured color images", *PAMI*, vol. 17, no. 10, p. 939-54, 1995.

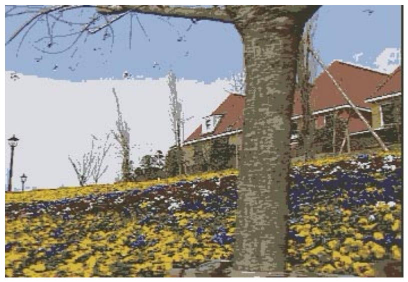
[9] L. Shafarenko, M. Petrou, and J. Kittler, "Automatic watershed segmentation of randomly textured color images", *IEEE Trans. on Image Processing*, vol. 6, no. 11, p. 1530-44, 1997.

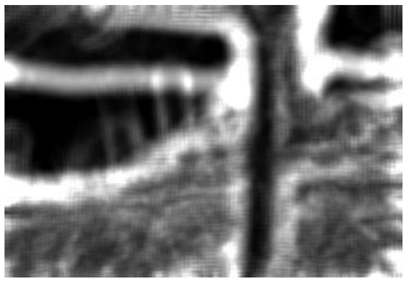
[10] J. Shi and J. Malik, "Normalized cuts and image segmentation", *Proc. of CVPR*, p. 731-37, 1997.

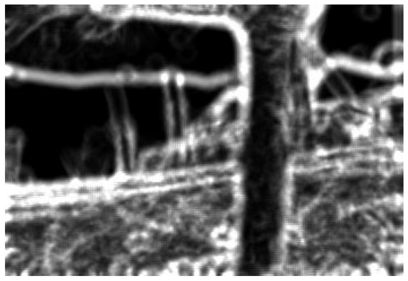
[11] J.-P. Wang, "Stochastic relaxation on partitions with connected components and its application to image segmentation", *PAMI*, vol. 20, no.6, p. 619-36, 1998.

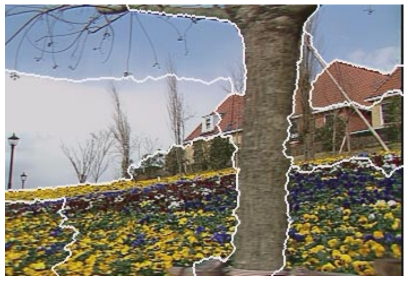
[12] S.C. Zhu and A. Yuille, "Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation", *PAMI*, vol. 18, no. 9, p. 884-900.

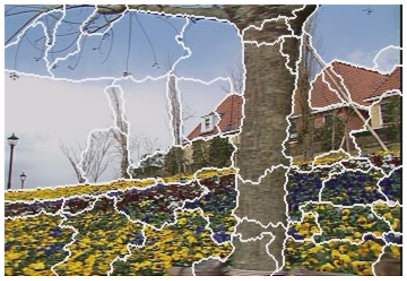

(a)

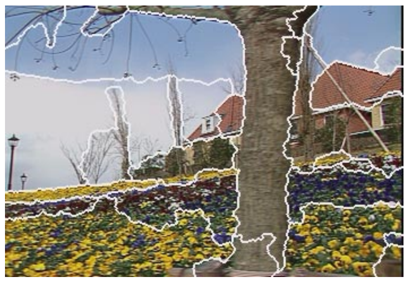

(b)


(c)


(d)


(e)

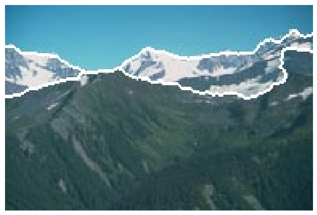

(f)

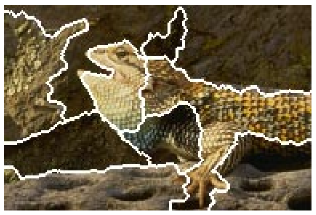

(g)

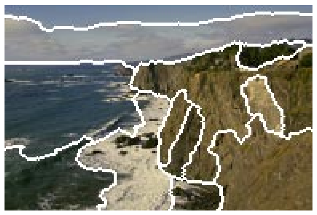
size: 352 x 240

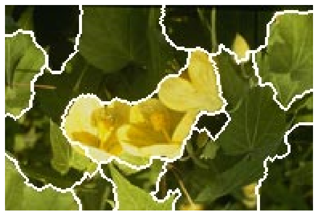
original (a): $\bar{J} = 0.435$
scale 3 (e): $\bar{J} = 0.103$
scale 2 (f): $\bar{J} = 0.125$
final (g): $\bar{J} = 0.088$

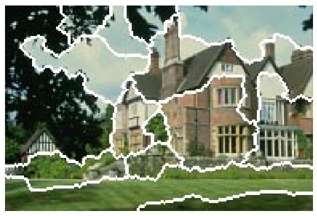
Fig 6. (a) original color image from the "flower garden" video sequence (frame 0), (b) result of color quantization with 13 colors, (c) *J*-image at scale 3, (d) *J*-image at scale 2, (e) result after segmentation at scale 3, 9 regions, (f) result after segmentation at scale 2, 49 regions, (g) final result of segmentation after merging, 27 regions.

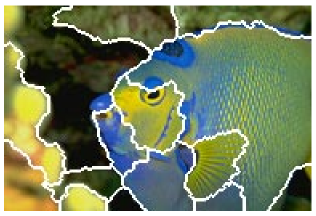

$\bar{J}_o = 0.296$, $\bar{J}_s = 0.021$

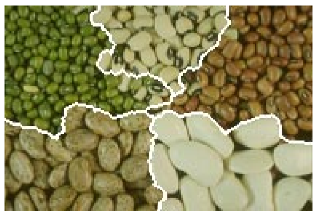

$\bar{J}_o = 0.093$, $\bar{J}_s = 0.081$


$\bar{J}_o = 1.247$, $\bar{J}_s = 0.071$


$\bar{J}_o = 0.062$, $\bar{J}_s = 0.140$


$\bar{J}_o = 0.600$, $\bar{J}_s = 0.217$


$\bar{J}_o = 0.651$, $\bar{J}_s = 0.327$


$\bar{J}_o = 0.445$, $\bar{J}_s = 0.040$

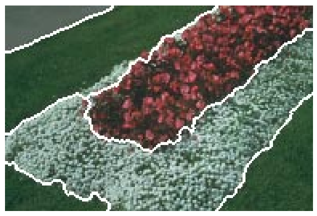

$\bar{J}_o = 0.098$, $\bar{J}_s = 0.029$

Fig 7. Example segmentation results on images from Corel photo CDs. 2,500 images are automatically processed without any parameter tuning on individual images. 1 scale, size 192 x 128, $\bar{J}_o$ is $\bar{J}$ calculated for the original image and $\bar{J}_s$ is for the segmented one.