Tutorial: Image Segmentation

Yu-Hsiang Wang (王昱翔)
E-mail: r98942059@ntu.edu.tw
Graduate Institute of Communication Engineering
National Taiwan University, Taipei, Taiwan, ROC

Abstract

For some applications, such as image recognition or compression, we cannot process the whole image directly for the reason that it is inefficient and unpractical. Therefore, several image segmentation algorithms were proposed to segment an image before recognition or compression. Image segmentation is to classify or cluster an image into several parts (regions) according to the feature of image, for example, the pixel value or the frequency response. Up to now, lots of image segmentation algorithms exist and be extensively applied in science and daily life. According to their segmentation method, we can approximately categorize them into region-based segmentation, data clustering, and edge-base segmentation. In this tutorial, we survey several popular image segmentation algorithms, discuss their specialties, and show their segmentation results. Moreover, some segmentation applications are described in the end.

1. Introduction

Image segmentation is useful in many applications. It can identify the regions of interest in a scene or annotate the data. We categorize the existing segmentation algorithm into region-based segmentation, data clustering, and edge-base segmentation. Region-based segmentation includes the seeded and unseeded region growing algorithms, the JSEG, and the fast scanning algorithm. All of them expand each region pixel by pixel based on their pixel value or quantized value so that each cluster has high positional relation. For data clustering, the concept of them is based on the whole image and considers the distance between each data. The characteristic of data clustering is that each pixel of a cluster does not certainly connective. The basis method of data clustering can be divided into hierarchical and partitional clustering. Furthermore, we show the extension of data clustering called mean shift algorithm, although this algorithm much belonging to density estimation. The last classification of segmentation is edge-based segmentation. This type of the segmentations generally applies edge detection or the concept of edge. The typical one is the watershed algorithm, but it always has the over-segmentation problem, so that the use of markers was proposed to improve the watershed algorithm by smoothing and selecting markers. Finally, we show some applications applying segmentation technique in the preprocessing.

2. Region-Based Segmentation Methods

Region-based methods mainly rely on the assumption that the neighboring pixels within one region have similar value. The common procedure is to compare one pixel with its neighbors. If a similarity criterion is satisfied, the pixel can be set belong to the cluster as one or more of its neighbors. The selection of the similarity criterion is significant and the results are influenced by noise in all instances. In this chapter, we discuss four algorithms: the Seeded region growing, the Unseeded region growing, the Region splitting and merging, and the Fast scanning algorithm.

2.1. Seeded Region Growing

2.1.1. The Concept of Seeded Region Growing

The seeded region growing (SRG) algorithm is one of the simplest region-based segmentation methods. It performs a segmentation of an image with examine the neighboring pixels of a set of points, known as seed points, and determine whether the pixels could be classified to the cluster of seed point or not [3]. The algorithm procedure is as follows.

Step1. We start with a number of seed points which have been clustered into n clusters, called C_1, C_2, …, C_n. And the positions of initial seed points is set as p_1, p_2, …, p_3.

Step2. To compute the difference of pixel value of the initial seed point p_i and its neighboring points, if the difference is smaller than the threshold (criterion) we define, the neighboring point could be classified into C_i, where $i = 1, 2, …,n$.

Step3. Recompute the boundary of C_i and set those boundary points as new seed points p_i (s). In addition, the mean pixel values of C_i have to be recomputed, respectively.

Step4. Repeat Step2 and 3 until all pixels in image have been allocated to a suitable cluster.

The threshold is made by user and it usually based on intensity, gray level, or color values. The regions are chosen to be as uniform as possible.

There is no doubt that each of the segmentation regions of SRG has high color similarity and no fragmentary problem. However, it still has two drawbacks, initial seed-points and time-consuming problems. The initial seed-points problem means the different sets of initial seed points cause different segmentation results. This problem reduces the stability of segmentation results from the same image. Furthermore, how many seed points should be initially decided is an important issue because various images have individually suitable segmentation number. The other problem is time-consuming because SRG requires lots of computation time, and it is the most serious problem of SRG.

2.1.2. Experimental Results and Discussion

We show the segmentation result of Lena image by using SRG. The sub-images are sorted according to the size of cluster from large to small. The display order is from left to right and from up to down. We only show the first 16 large clusters in Fig. 1.

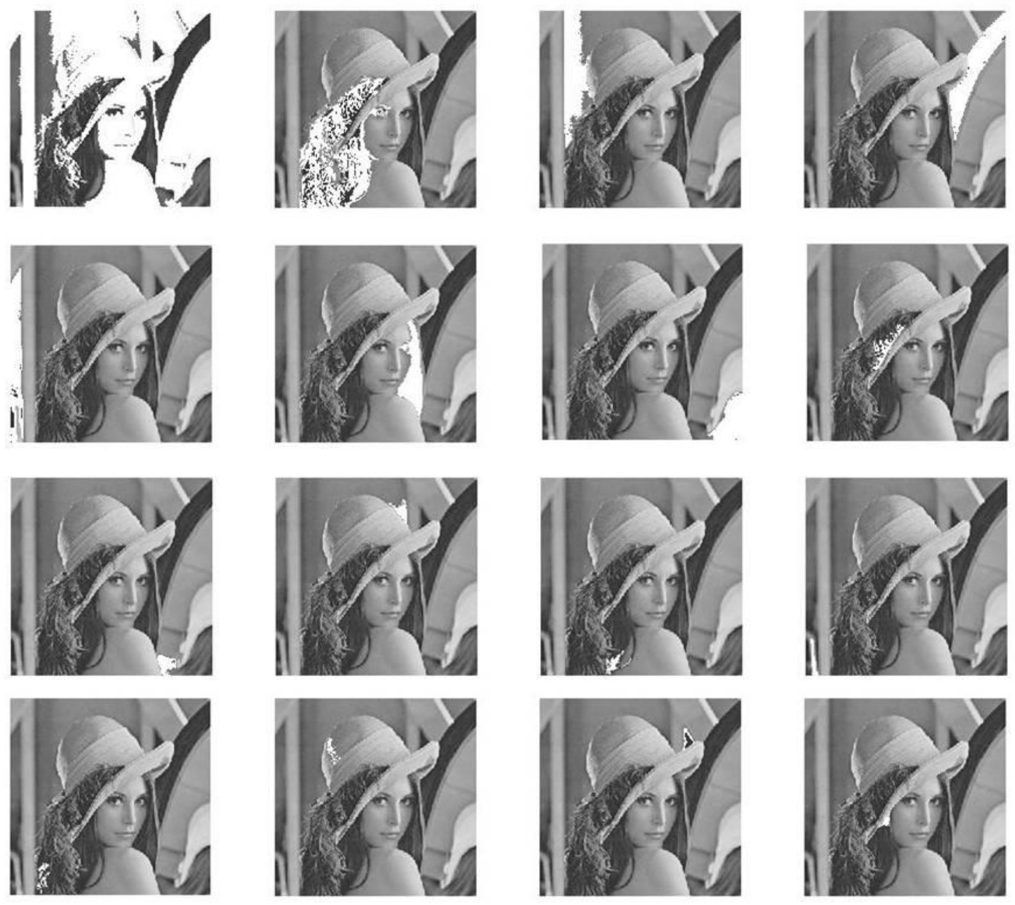


Fig. 1 The segmentation result of Lena image using SRG.

2.2. Unseed Region Growing

2.2.1. The Concept of Unseed Region Growing

The unseeded region growing (URG) algorithm is a derivative of seeded region growing proposed by Lin et al. [4]. Their distinction is that no explicit seed selection is necessary. In the segmentation procedure, the seeds could be generated automatically. So this method can perform fully automatic segmentation with the added benefit of robustness from being a region-based segmentation. The steps of URG are as below.

Step1. The process initializes with cluster C_1 containing a single image pixel, and the running state of the process compose of a set of identified clusters, C_1, C_2, …, C_n.

Step2. We define the set of all unsigned pixels which borders at least one of those clusters as:

$$S = \left\{ x \notin \bigcup_{i=1}^{n} C_i \wedge \exists k : N(x) \bigcap C_k \neq \varnothing \right\}, \tag{2.1}$$

where $N(x)$ are current neighboring pixels of point x. Moreover, let δ be a difference measure

$$\delta(x, C_i) = \left| g(x) - \underset{y \in C_i}{\operatorname{mean}} \left[g(y) \right] \right|, \tag{2.2}$$

where $g(x)$ denotes the pixel value of point x, and i is an index of the cluster such that $N(x)$ intersect C_i.

Step3. To choose a point $z \in S$ and cluster C_j where $j \in [1, n]$ such that

$$\delta(z, C_j) = \min_{x \in S, k \in [1,n]} \left\{ \delta(x, C_k) \right\}. \tag{2.3}$$

If $\delta(z, C_j)$ is less than the predefined threshold t, the pixel is clustered to C_j. Else, we must select the most considerable similar cluster C such that

$$C = \arg \min_{C_k} \left\{ \delta(z, C_k) \right\}. \tag{2.4}$$

If $\delta(z, C) < t$, then we can allocate the pixel to C. If neither of two conditions conform, it is obvious that the pixel is substantially from all the clusters found so far, so that a new cluster C_{n+1} would be generated and initialized with point z.

Step4. After the pixel has been allocated to the cluster, the mean pixel value of the cluster must be updated.

Step5. Iterate Step2 to 4 until all pixels have been assigned to a cluster.

2.2.2. Experimental Results and Discussion

From Fig. 2, the segmentation result of unseed region growing seems a little over-segmentation. We speculate the result will be better after the threshold is adjusted higher.

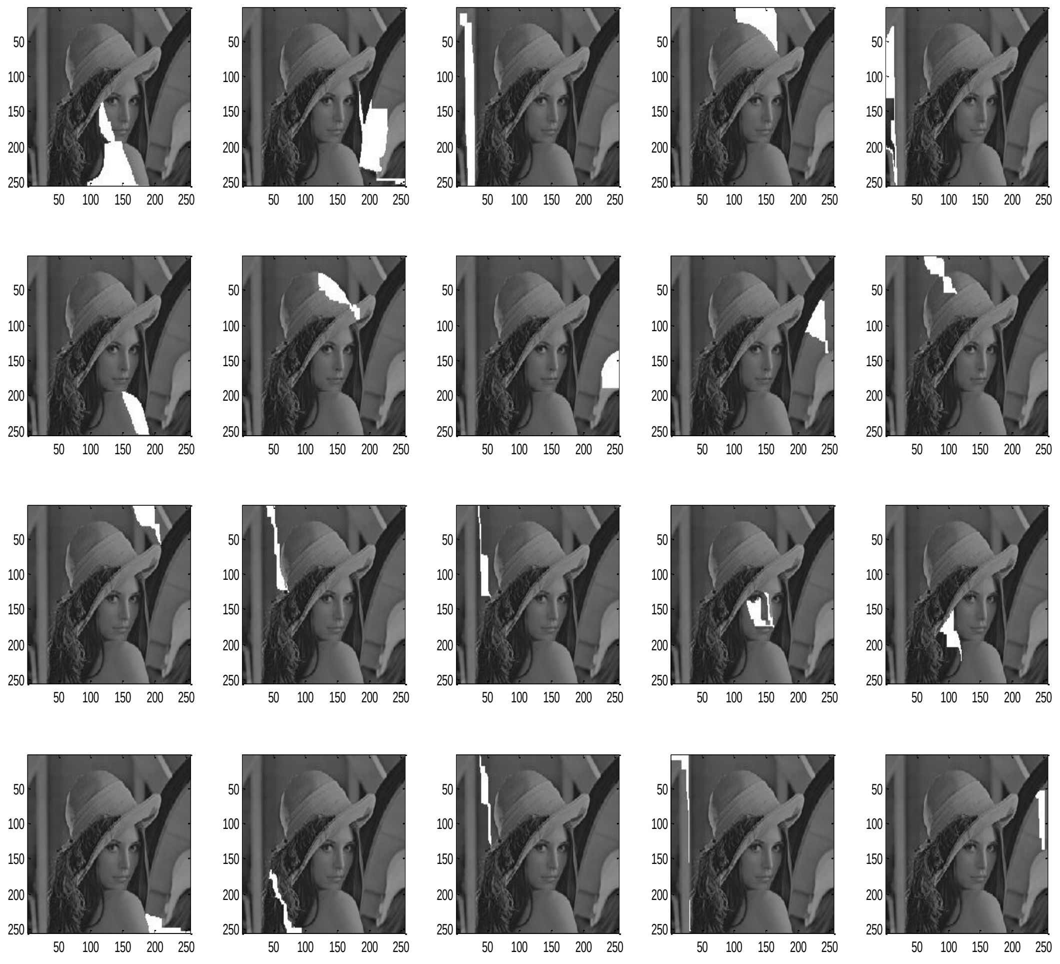


Fig. 2 The segmentation result of Lena image using URG.

2.3. Region Splitting and Merging

The main goal of region splitting and merging is to distinguish the homogeneity of the image [5]. Its concept is based on quadtrees, which means each node of trees has four descendants and the root of the tree corresponds to the entire image. Besides, each node represents the subdivision of a node into four descendant nodes. The instance is shown in Fig. 3(a), and in the case of Fig. 3(b), only R_4 was subdivided further. The basics of splitting and merging are discussed below.

Let R represent the entire image region and decide a predicate P. The purpose is that if $P(R) = \text{FALSE}$, we divide the image R into quadrants. If P is FALSE for any quadrant, we subdivide that quadrant into subquadrants, and so on. Until that, for any region R_i, $P(R_i) = \text{TRUE}$. After the process of splitting, merging process is to merge two adjacent regions R_j and R_k if $P(R_j \cup R_k) = \text{TRUE}$. The summarized procedure is described as follows:

Step1. Splitting steps: For any region R_i, which $P(R_i) = \text{FALSE}$, we split it into four disjoint quadrants.

Step2. Merging steps: When no further splitting is possible, merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k) = \text{TRUE}$.

Step3. Stop only if no further merging is possible.

We conclude the advantages and disadvantages of region splitting and merging:

Advantages:

a. The image could be split progressively according to our demanded resolution because the number of splitting level is determined by us.

b. We could split the image using the criteria we decide, such as mean or variance of segment pixel value. In addition, the merging criteria could be different to the splitting criteria.

Disadvantages:

a. It may produce the blocky segments.

The blocky segment problem could be reduced by splitting in higher level, but the trade off is that the computation time will arise.

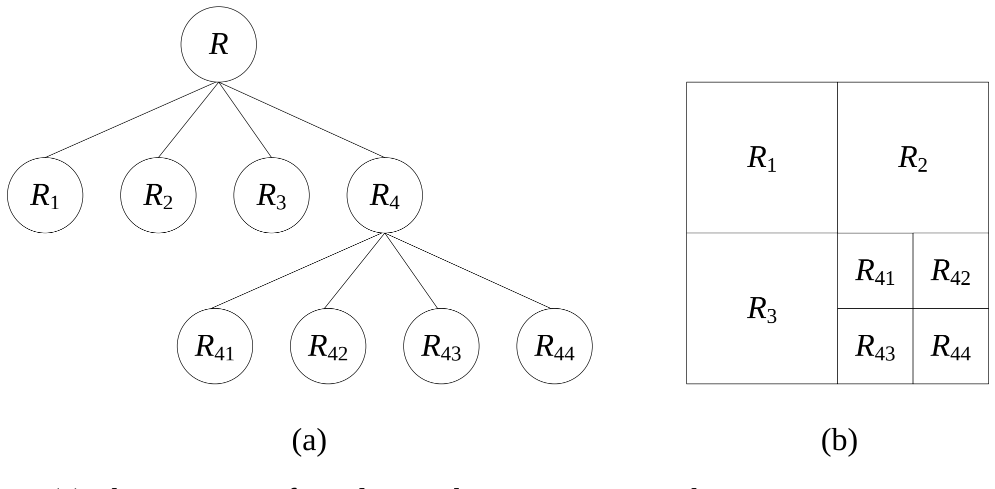


Fig. 3 (a) The structure of quadtree, where R represents the entire image region. (b) Corresponding partitioned image. [1]

2.4. Unsupervised Segmentation of Color-Texture Regions in Images and Video (JSEG)

The drawback of unsupervised segmentation is ill-defined because the segmented objects do not usually conform to homogeneous spatiotemporal regions in color, texture, or motion. Hence, in 2001, Deng et al. present the method for unsupervised segmentation of color-texture regions in images and video, called as JSEG [6]. The goal of this algorithm is to segment images and video into homogeneous color-texture regions. In this thesis, we only describe the image segmentation part of JSEG.

The concept of the JSEG algorithm is to separate the segmentation process into two portions, color quantization and spatial segmentation. The color quantization quantizes colors in image into several representative classes that can differentiate regions in the image. The process of quantization is implemented in the color space without considering the spatial distribution of the colors. The corresponding color class labels replace the original pixel values and then create a class-map of the image. The CIE LUV color space is used for the color space in JSEG. In second portion, spatial segmentation executes on the class-map instead of regarding the corresponding pixel color similarity. The benefit of this separation is that respectively analyzing the similarity of the colors and their distribution is more tractable than complete them at the same time.

2.4.1. Criterion for Image Segmentation

Before introduce the JSEG algorithm, we discuss the new criterion for segmentation applied in JSEG. The preceding process of the criterion is an unsupervised color quantization algorithm based on human perception [7]. This quantization method quantizes the set of image pixels to the same color, called color class. Then the image pixel colors are replaced by their corresponding color class label and the newly established image of labels is called a class-map. The class-map can be regarded as a special kind of texture composition. In the following, the details of this criterion are described.

Let Z be the set of all N data points in a class-map and $z = (x, y)$, where $z \in Z$ and (x, y) is the image pixel position. The mean m is

$$m = \frac{1}{N}\sum_{z \in Z} z. \tag{2.5}$$

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

$$m_i = \frac{1}{N_i}\sum_{z \in Z_i} z. \tag{2.6}$$

Let

$$S_T = \sum_{z \in Z} \|z - m\|^2. \tag{2.7}$$

and

$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} \sum_{z \in Z_i} \| z - m_i \|^2. \tag{2.8}$$

S_w is the total variance of points belonging to the same class. Define

$$J = \left(S_T - S_W \right) / S_W. \tag{2.9}$$

The value of *J* relates with the degree of the color classes' distribution. More uniform distribution of the color classes is, smaller the value of *J* is. An example of such a class-map is shown in Fig. 4(a) for which *J* equals 0. On the other hand, if an image consists of several homogeneous color regions and the color classes are more detached from each other, the corresponding value *J* is large. This is illustrated by class-map 3 in Fig. 4(c) and the corresponding value *J* is 1.720. There exists another example in Fig. 4(b) for which *J* is 0.855 that signifies more homogeneous distribution than class-map 1. The motivation for the definition of J derives from the Fisher's multiclass linear discriminant [8], but for arbitrary nonlinear class distributions.

+	#	+	#	+	#	+	#	+
o	+	o	+	o	+	o	+	o
+	#	+	#	+	#	+	#	+
o	+	o	+	o	+	o	+	o
+	#	+	#	+	#	+	#	+
o	+	o	+	o	+	o	+	o
+	#	+	#	+	#	+	#	+
o	+	o	+	o	+	o	+	o
+	#	+	#	+	#	+	#	+

(a) class-map 1
J = 0

+	+	+	+	+	#	o	#	o
+	+	+	+	+	o	#	o	#
+	+	+	+	+	#	o	#	o
+	+	+	+	+	o	#	o	#
+	+	+	+	+	#	o	#	o
+	+	+	+	#	o	#	o	#
+	+	+	+	o	#	o	#	o
+	+	+	+	#	o	#	o	#
+	+	+	+	o	#	o	#	o

(b) class-map 2
J = 0.855

+	+	+	+	+	o	o	o	o
+	+	+	+	+	o	o	o	o
+	+	+	+	+	o	o	o	o
+	+	+	+	+	o	o	o	o
+	+	+	+	+	o	o	o	o
+	+	+	+	#	#	#	#	#
+	+	+	+	#	#	#	#	#
+	+	+	+	#	#	#	#	#
+	+	+	+	#	#	#	#	#

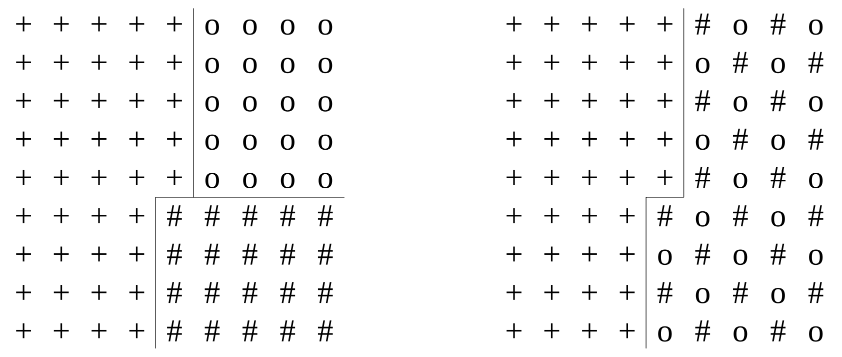
(c) class-map 3
J = 1.720

Fig. 4 An example of different class-maps and their corresponding *J* values. "+", "o", and "#" represent three classes of data points [6].

After realizing the operational method of *J*, in the algorithm of JSEG, we need to apply two values: average $\bar{J}$ and local *J* value. The average $\bar{J}$ is used as the criterion to estimate the performance of segmentation region. The average $\bar{J}$ is defined as

$$\bar{J} = \frac{1}{N} \sum_{k} M_k J_k, \tag{2.10}$$

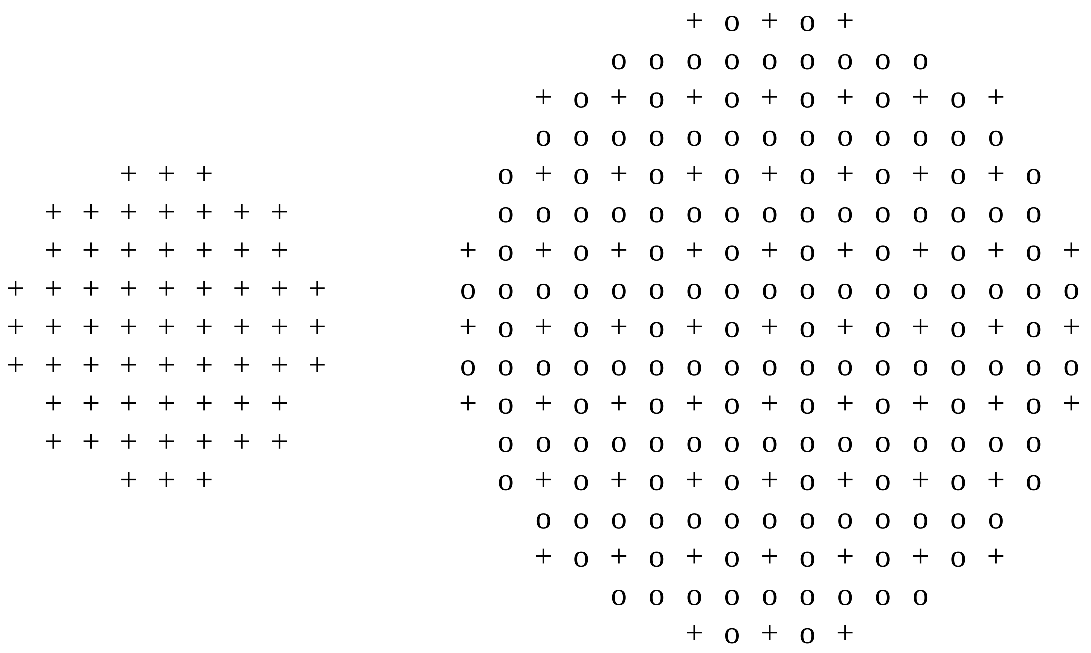
where J_k is *J* computed over region *k*, M_k is the number of points (pixels) in region *k*, *N* is the total number of points in the class-map, and the summation is over all the regions in the class-map. For a fixed number of regions, a "better" segmentation normally has a lower value of $\bar{J}$. A low value of $\bar{J}$ represents each segmented region contains a few uniformly distributed color class labels. shows two examples of segmented class-map and their $\bar{J}$ values.



(a) segmented class-map 1
$J_+ = 0$, $J_o = 0$, $J_\# = 0$
$\bar{J} = 0$

(b) segmented class-map 2
$J_+ = 0$, $J_{\{o, \#\}} = 0.011$
$\bar{J} = 0.05$

Fig. 5 Two examples of segmented class-map and their corresponding $\bar{J}$ values [6].



(a)

(b)

Fig. 6 (a) The basic window for computing local *J* values. (b) The downsampling for the window at scale 2. Only "+" pixels are used for computing local *J* values [6].

The operation of local *J* value bases on the local window. The size of the local window determines the size of image regions that can be detected. Small windows are useful for localizing the intensity / color edges, while windows of large size are useful in detecting texture boundaries. In addition, the higher the local J value is, the more likely that the corresponding pixel is near a region [6]. In JSEG algorithm, multiple scales are applied to segment an image. The basic window at the smallest scale is a 9×9 window without corners, as shown in Fig. 6(a). The larger window size (larger scale) is obtained by doubling the size of the forward scale. The window at scale 2 is shown in Fig. 6(b), where the sampling rate is 1 out of 2 pixels along both x and y directions. Note that user determines the number of scales needed for the image, which affects how detail the segmentation will be.

2.4.2. Algorithm of JSEG

According to the characteristics of the *J* values, the modified region growing method can be applied to segment an image. The algorithm starts the segmentation at the largest scale. Then it repeats the same process on the newly segmented regions at the next lower scale. After finishing the final segmentation at the smallest scale, the region merging operation follows region growing to derive the final segmentation result. The flow-chart of the steps in JSEG is presented in Fig. 7. In the following sections, we describe three steps: seed determination, seed growing, and region merge, respectively.

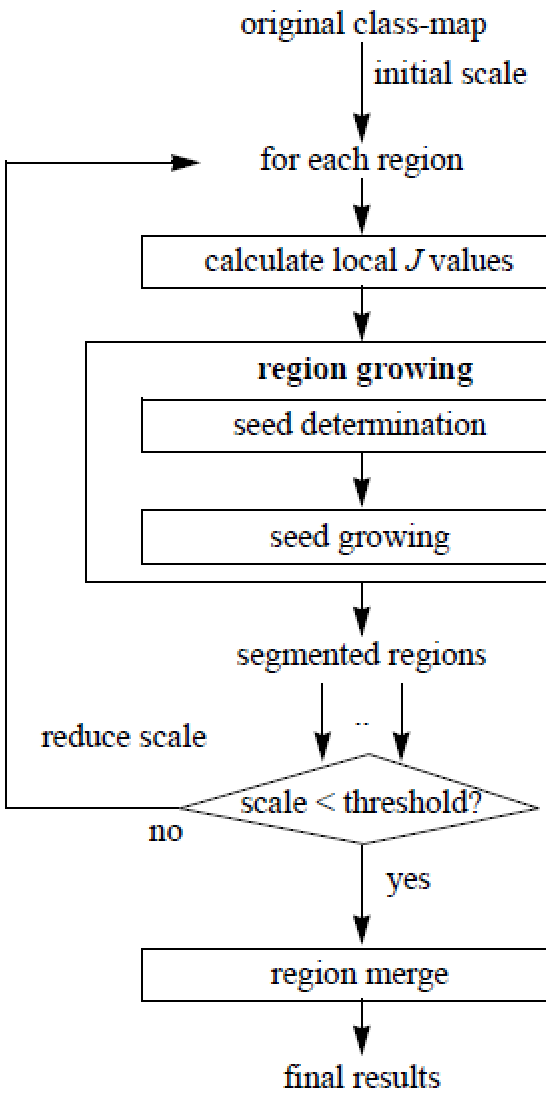


Fig. 7 Flow-chart of the steps in JSEG [6].

2.4.2.1.Seed Determination

Before operating seed growing, we must determine the initial seed areas. These areas correspond to minima of local *J* values. The steps of seed determination method are shown as below.

Step1. Compute the average and the standard deviation of the local J values in the region, denoted as μ_J and T_J, respectively.

Step2. Define a threshold T_J

$$T_J = \mu_J + \alpha\sigma_J, \tag{2.11}$$

where α is selected from several preset values that will result in the most number of seeds. Then we set the pixels with local *J* values less than T_J as candidate seed points and connect the candidate seed points based on the 4-connectivity and obtain the candidate seed areas.

Step3. If a size of the candidate seed area is larger than the minimum size listed in Table 1 at the corresponding scale, then it is determined as a seed.

Table 1 Window size at different scales.

scale	window (pixels)	sampling (1 / pixels)	region size (pixels)	min. seed (pixels)
1	9×9	$1/(1\times1)$	64×64	32
2	17×17	$1/(2\times2)$	128×128	128
3	33×33	$1/(4\times4)$	256×256	512
4	65×65	$1/(8\times8)$	512×512	2048

2.4.2.2.Seed Growing

The seed growing is not the same as the region growing described before. This seed growing is presented for accelerating the computation time. The procedure is shown as below.

Step1. Remove “holes” in the seeds.

Step2. Compute the average of the local *J* values in the remaining unsegmented part of the region and connect pixels below the average to compose growing areas. If a growing area is adjacent to one and only one seed, we merge this growing area into that seed.

Step3. Compute local *J* values of the remaining unsegmented pixels at the next smaller scale and repeat Step 2 (This motion is to more precisely locate the boundaries). When the operation reaches the smallest scale, continue to Step 4.

Step4. At the smallest scale, the remaining pixels are grown one by one. The remaining pixels are sorted by their local *J* values. The pixel is assigned to its adjacent seed in order from the minimum local J value to maximum.

2.4.2.3.Region Merge

The segmentation from seed growing has the problem of over-segmented regions. Region merge is in order to solve over-segmented regions. These regions are merged based on their color similarity. The color information of region is characterized by its color histogram and the color histogram bins are according to the quantized colors from the color quantization process. Then we calculate the Euclidean distance between two color histograms i and j shown as

$$D_h(i, j) = \left\| P_i - P_j \right\|, \tag{2.12}$$

where P denotes the color histogram vector. The method of region merge is based on the agglomerative method in [8]. We first construct a distance table containing the distances between the color histogram of any two neighboring regions. The pair of regions with the minimum distance is merged together. Then the new color feature vector of the new region is computed and the distance table is updated along with the neighboring relationships. The region merge process continues until a maximum threshold for the distance is reached.

2.4.2.4.Experimental Results and Discussion

Fig. 8 shows several results of the segmentation from [6]. Generally, these results match well with the sensed color-texture boundaries and the $\bar{J}$ values for those segmentation results mainly larger than for the original ones. Only an example in Fig. 8 where the $\bar{J}$ values are poor for the segmented images.

JSEG can also be applied in gray level images where the intensity values are quantized the same as the colors. Two instances are shown in Fig. 9. The results are reasonably not as good as the color image ones because that intensity alone is not as discriminative as color.

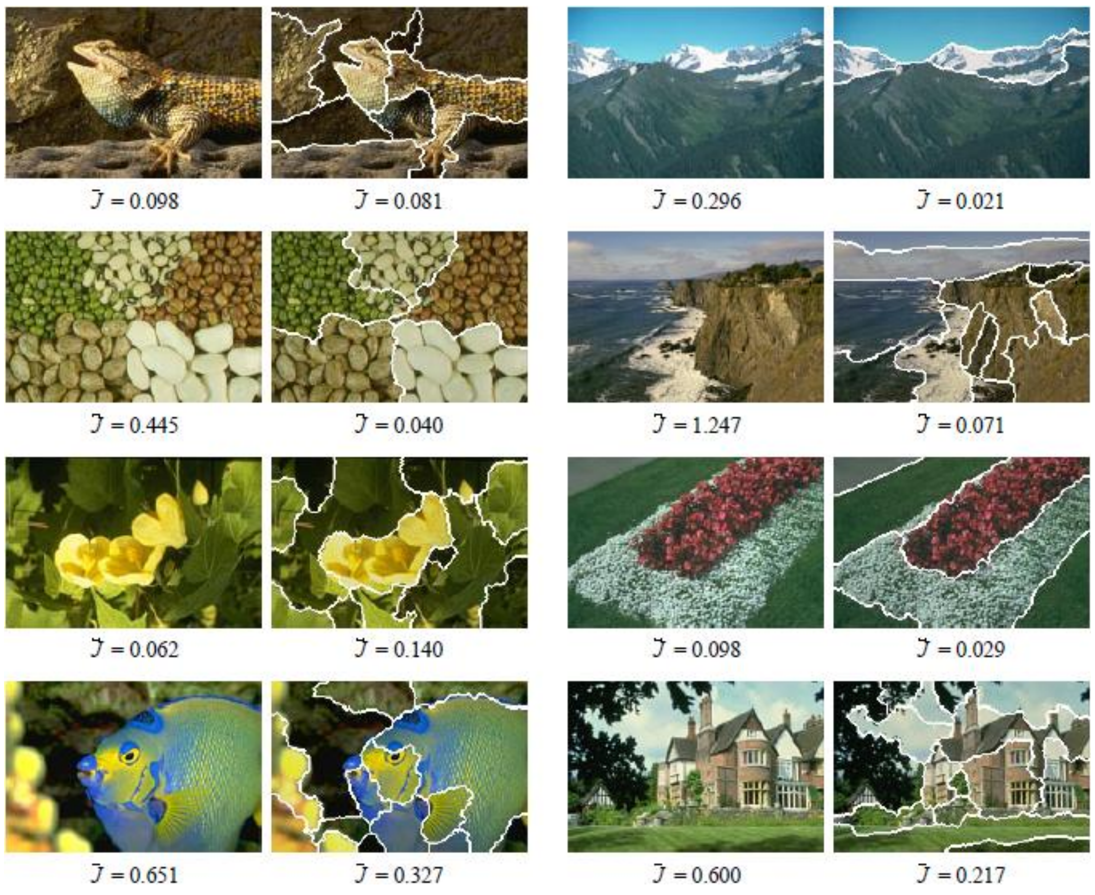


Fig. 8 Segmentation results of some examples using JSEG (one scale, size 192×128 pixels) [6].

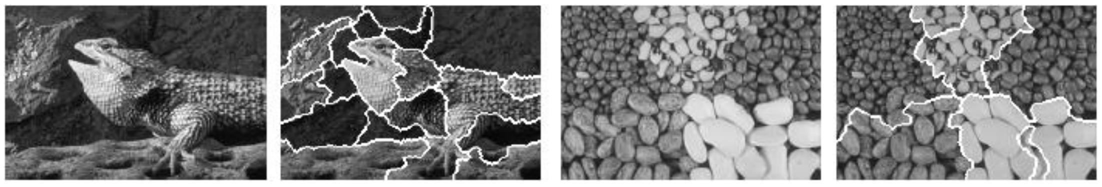


Fig. 9 Segmentation of gray level images using JSEG. The corresponding color image results are in Fig. 8. [6]

2.5. Fast Scanning Algorithm

2.5.1. The Concept of Fast Scanning Algorithm

Unlike region growing, fast scanning algorithm do not need seed point. The concept of fast scanning algorithm [9] is to scan from the upper-left corner to lower-right corner of the whole image and determine if we can merge the pixel into an existed clustering. The merged criterion is based on our assigned threshold. If the difference between the pixel value and the average pixel value of the adjacent cluster is smaller than the threshold, then this pixel can be merged into the cluster. The threshold usually chooses 45. We describe the steps of the fast scanning algorithm as below.

Step1. Let the upper left pixel as the first cluster. Set the pixel (1, 1) in the image as one cluster C_i and the pixel which we are scanning as C_j. We give an example in Fig. 10 and assume that the *threshold* is 45 here.

Step2. In the first row, we scan the next pixel (1, 1+1) and determine if it can be merged into the first cluster or become a new cluster according to the threshold. The judgments are in the following, where *mean* represents the average pixel value of cluster C_i.

- If $\left|C_j - mean(C_i)\right| \le threshold$ then we merge C_j into C_i and recalculate the *mean* of C_i. Fig. 10(b) shows this case.
- If $\left|C_j - mean(C_i)\right| > threshold$ then we set C_j as a new cluster C_{i+1}. This case is shown in Fig. 10(c).

Step3. Repeat Step 2 until all the pixels in the first row have been scanned.

Step4. To scan the pixel (x+1, 1) in the next row and compare this pixel with the cluster C_u which is in the upside of it. And determine if we can merge the pixel (x+1, 1) into the cluster C_u. (In the 2nd row, x is 1 and it increases with iteration)

- If $\left|C_j - mean(C_u)\right| \le threshold$ then we merge C_j into C_u and recalculate the *mean* of C_u.
- If $\left|C_j - mean(C_u)\right| > threshold$ then we set C_j as a new cluster C_n, where n is the cluster number so far. Fig. 10(e) shows above two situations.

Step5. Scan the next pixel (x+1, 1+1) and compare this pixel with the cluster C_u and C_l, which is in the upside of it and in the left side of it, respectively. And decide if we can merge the pixel (x+1, 1+1) into anyone of two clusters.

- If $\left|C_j - mean(C_u)\right| \le threshold$ & $\left|C_j - mean(C_l)\right| \le threshold$,

  (1) We merge C_j into C_u or merge C_j into C_l.
  (2) Merge the cluster C_u and C_l into cluster C_n, where n is the cluster number so far.
  (3) Recompute the *mean* of C_n.

  This case is shown in Fig. 10(f).
- If $\left|C_j - mean(C_u)\right| \le threshold$ & $\left|C_j - mean(C_l)\right| > threshold$,

we merge C_j into C_u and recalculate the *mean* of C_u.

- If $\left|C_j - mean\left(C_u\right)\right| > threshold$ and $\left|C_j - mean\left(C_l\right)\right| \le threshold$, we merge C_j into C_l and recalculate the *mean* of C_l.
- Otherwise, set C_j as a new cluster C_n, where n is the cluster number so far.

Step6. Repeat Step 4 to 5 until all the pixels in the image have been scanned.

Step7. Remove small clusters. If the number of $C_m < \Delta$, we remove cluster m and assign the pixels in cluster m into adjacent clusters. The assignment is according to the smallest differences between the pixel and its mean of adjacent clusters. Fig. 10(g)(h) shows the small cluster case.

We conclude the advantages and disadvantages of the fast scanning algorithm:

Advantages:

a. The pixels of each cluster are connected and have similar pixel value, i.e. it has good shape connectivity.

b. The computation time is faster than both region growing algorithm and region splitting and merging algorithm.

c. The segmentation results exactly match the shape of real objects, i.e. it has good shape matching.

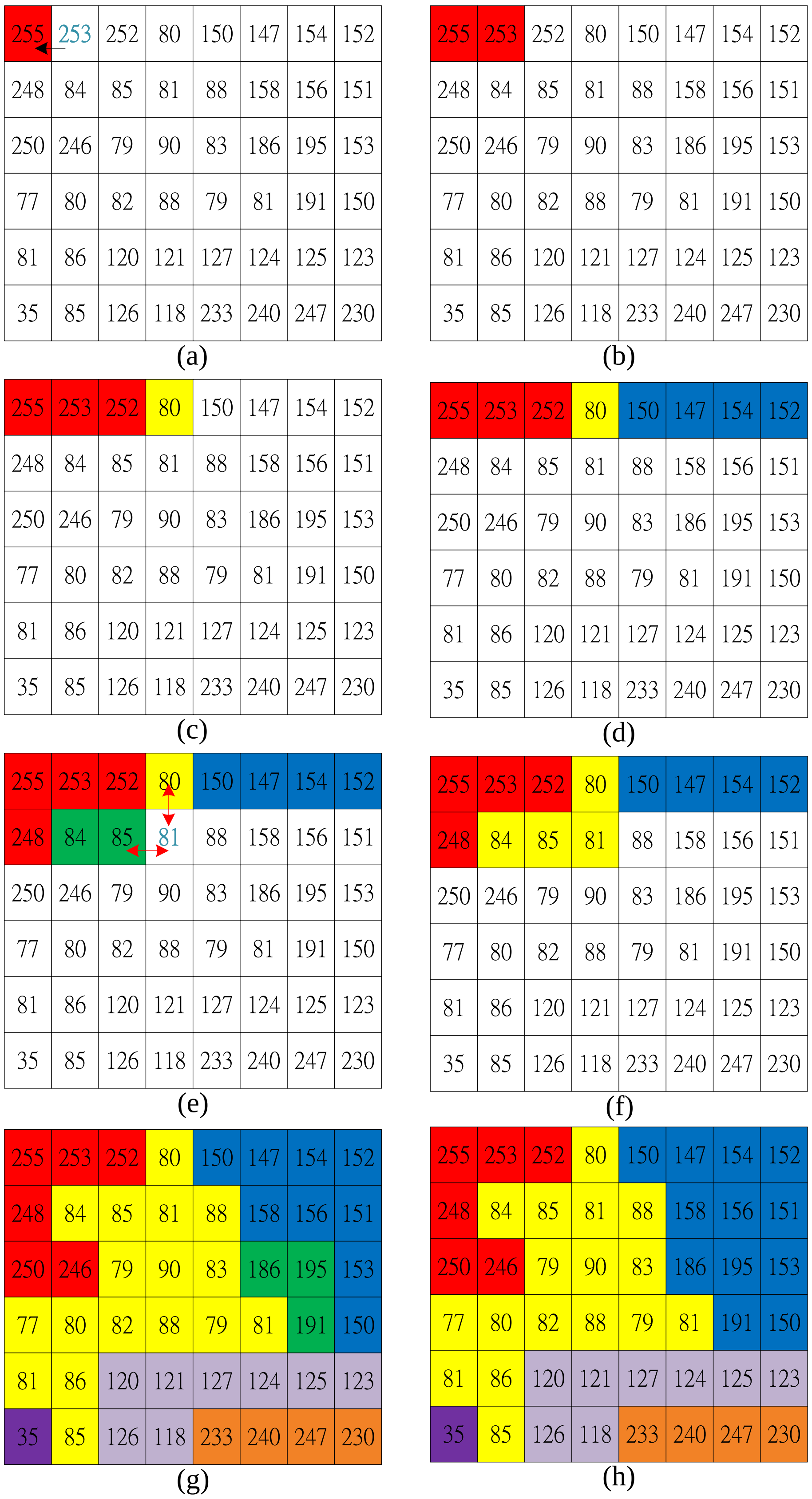


Fig. 10 Applying the fast scanning algorithm to an example image.

2.5.2. Experimental Results and Discussion

The regions of result are quite complete except some regions are not good in connectivity of shape. For example, the cloche in first region should be separated with the background.

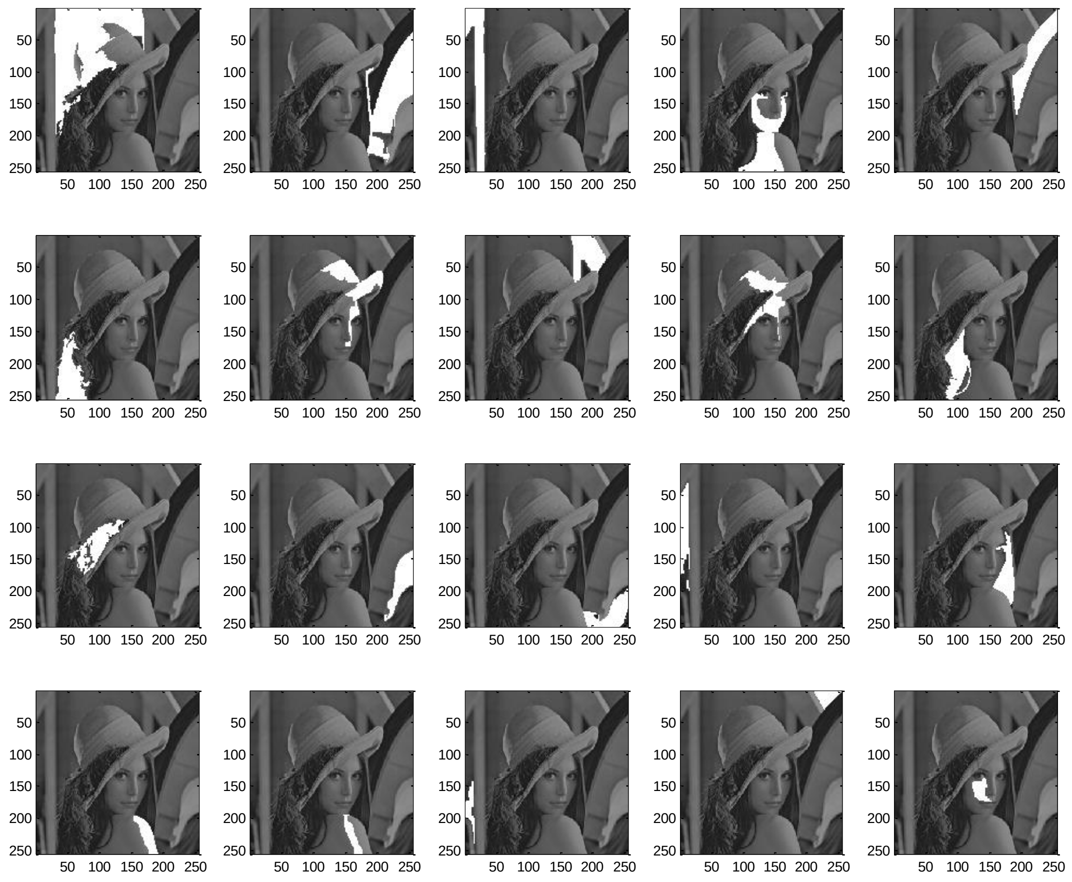


Fig. 11 The segmentation result of Lena image using the fast scanning algorithm.

3. Data clustering

Data clustering is one of methods widely applied in image segmentation and statistic. The main concept of data clustering is to use the centroid to represent each cluster and base on the similarity with the centroid of cluster to classify. According to the characteristics of clustering algorithm, we can roughly divide into “hierarchical” and “partitional” clustering. Except for this two classes, mean shift algorithm is part of data clustering, too, and its concept is based on density estimation.

3.1. Hierarchical Clustering

The concept of hierarchical clustering is to construct a dendrogram representing the nested grouping of patterns (for image, known as pixels) and the similarity levels at which groupings change. We can apply the two-dimensional data set to interpret the operation of the hierarchical clustering algorithm. For example, the eight patterns labeled A, B, C, D, E, F, G, and H in three clusters shown in Fig. 12 and Fig. 13 shows the dendrogram corresponding to the eight patterns in Fig. 12.

The hierarchical clustering can be divided into two kinds of algorithm: the hierarchical agglomerative algorithm and the hierarchical divisive algorithm. We describe the detail of two algorithms as below, respectively.

Hierarchical agglomerative algorithm:

Step1. Set each pattern in the database as a cluster C_i and compute the proximity matrix including the distance between each pair of patterns.

Step2. Use the proximity matrix to find out the most similar pair of clusters and then merge these two clusters into one cluster. After that, update the proximity matrix.

Step3. Repeat Step 1 and 2 until all patterns in one cluster or just achieve the similarity we demand, e.g. the dash line in Fig. 13.

Notice that there are several ways to define the "distance" between each pair of patterns. The most popular definitions are the single-link and complete-link algorithms. If we assume D(C_i, C_j) as the distance between cluster C_i and C_j, and assume $d(a, b)$ as the distance between pattern a and b. Then the distance definition of single-link method is

$$D\left(C_i, C_j\right) = \min\left(d\left(a, b\right)\right), \quad \text{for } \forall a \in C_i, \forall b \in C_j, \tag{3.1}$$

which means the distance between two clusters is the minimum of all pairwise distances between patterns in the two clusters. The distance definition of complete-link method is

$$D\left(C_i, C_j\right) = \max\left(d\left(a, b\right)\right), \quad \text{for } \forall a \in C_i, \forall b \in C_j, \tag{3.2}$$

which means the distance between two clusters is the maximum of the distances between all pairs of patterns in the two clusters.

The advantage of complete-link method is that it can produce more compact clusters than the single-link method [11]. Although the single-link method is more versatile than the complete-link method [12], the complete-link method generates more practical hierarchies in many applications than the single-link method [13].

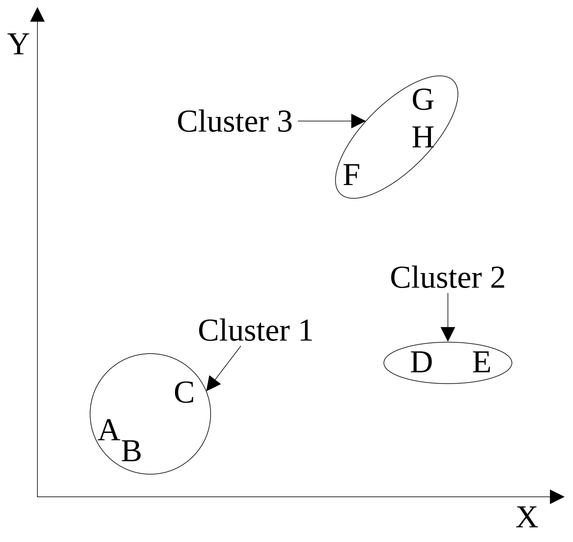


Fig. 12 Grouping eight patterns into three clusters.

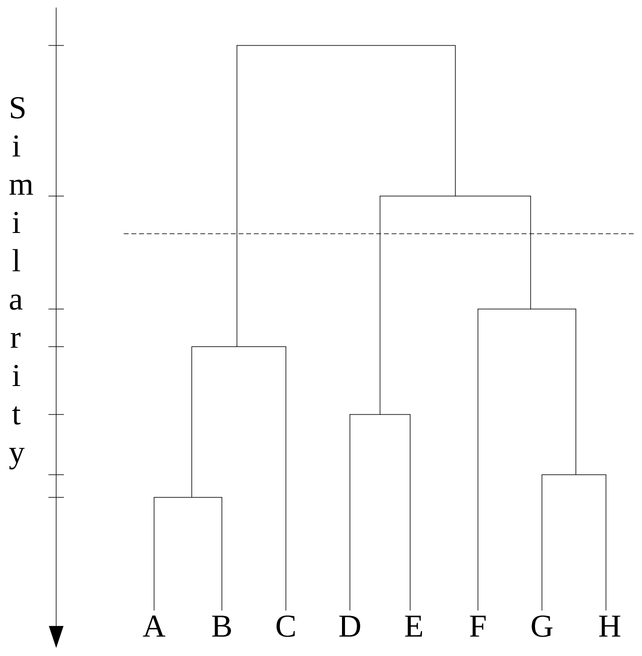


Fig. 13 The dendrogram corresponds to the eight patterns in Fig. 12 using the single-link algorithm.

Hierarchical divisive algorithm:

Before we introduce the algorithm of hierarchical division, it has to define the distance between pattern *x* (as for image, the pixel) and cluster *C* as *d*(*x*, *C*) = the mean of the distance between *x* and each pattern in the cluster *C*. The steps of hierarchical divisive algorithm are as below.

Step1. Start with one cluster of the whole database (as for image, the whole image).

Step2. Find the pattern x_i in cluster C_i satisfied $d(x, C_i) = \max(d(y, C_i))$, for $\forall y \in C_i$, where $i = 1, 2, \ldots, N$ and N is the current number of clusters in the whole database.

Step3. Split x_i out as a new cluster C_{i+N}, and then compute $d(y, C_i)$ and $d(y, C_{i+N})$, for $\forall y \in C_i$. If $d(y, C_i) > d(y, C_{i+N})$, then split y out of C_i and merge it into C_{i+N}.

Step4. Repeat to Step 2 until all of the clusters are not change anymore.

The advantages and disadvantages of the hierarchical algorithm are concluded as below.

Advantages:

a. The process and relationships of hierarchical clustering can just be realized by checking the dendrogram.
b. The result of hierarchical clustering presents high correlation with the characteristics of original database.
c. We only need to compute the distances between each pattern, instead of calculating the centroid of clusters.

Disadvantages:

a. For the reason that hierarchical clustering involves in detailed level, the fatal problem is the computation time.

3.2. Partitional Clustering

In contrast with the hierarchical clustering constructing a clustering structure, the partitional clustering algorithm obtains a single partition of the data. It is useful to implement in large data sets, but for hierarchical clustering, the construction of dendrogram needs lots of computation time. The problem of partitional clustering is that we have to select the number of desired output clusters before we start to classify data. Some papers provide the guidance of this problem which we mention later.

3.2.1. Squared Error algorithm

Before we describe the steps of partitional clustering, the convergence criterion should be mentioned. The concept of partitional clustering is to start with random initial data points and keep reassigning the patterns to clusters based on the similarity between the pattern and the centroid of clusters until a convergence criterion is encountered. One of convergence criterion frequently applied is squared error algorithm [14]. The benefit of squared error is that it works well with isolated and compact clusters. The squared error for a clustering L of a pattern set R (containing K clusters) is

$$e^2(R,L)=\sum_{j=1}^{K}\sum_{i=1}^{n_j}\left\|\mathbf{x}_i^{(j)}-\mathbf{c}_j\right\|^2, \tag{3.3}$$

where $\mathbf{x}_i^{(j)}$ is the *i*th pattern belonging to the *j*th cluster and $\mathbf{c}_j$ is the centroid of the *j*th cluster.

3.2.2. K-means Clustering Algorithm

The most famous partitional clustering algorithm is k-means clustering. The steps of k-means clustering are as below.

Step1. Determine the number of clusters we want in the final classified result and set the number as *N*. Randomly select *N* patterns in the whole data bases as the *N* centroids of *N* clusters.

Step2. Classify each pattern to the closest cluster centroid. The closest usually represent the pixel value is similarity, but it still can consider other features.

Step3. Recompute the cluster centroids and then there have *N* centroids of *N* clusters as we do after Step1.

Step4. Repeat the iteration of Step 2 to 3 until a convergence criterion is met. The typical convergence criteria are: no reassignment of any pattern from one cluster to another, or the minimal decrease in squared error.

We conclude the advantages and disadvantages of the k-means clustering algorithm as follows:

Advantages:

a. K-means algorithm is easy to implement.
b. Its time complexity is *O*(*n*), where *n* is the number of patterns. It is faster than the hierarchical clustering.

Disadvantages:

a. The result is sensitive to the selection of the initial random centroids.
b. We cannot show the clustering details as hierarchical clustering does.

Here we give an example of seven two-dimensional patterns shown in Fig. 14, the problem of being sensitive to the initial partition. If we start with patterns A, B, and C as the initial three centroids, then the end partition will be {{A}, {B, C}, {D, E, F, G}} shown by ellipses. The squared error criterion value for this partition is much larger than for the best partition {{A, B, C}, {D, E}, {F, G}} shown by rectangles. The solution of this problem is summarized in the next section.

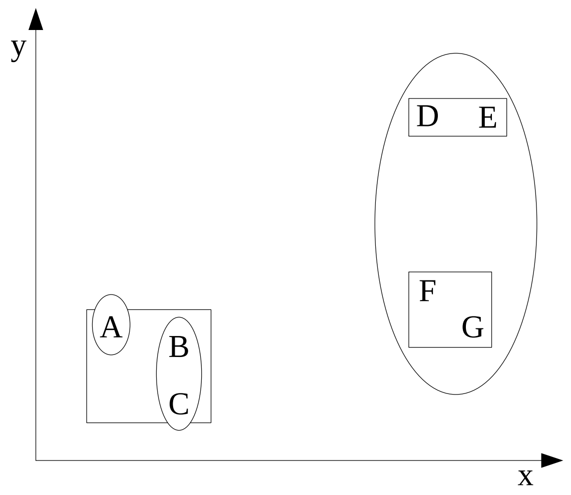


Fig. 14 The example of the k-means algorithm, which is sensitive to initial partition. [10]

3.2.3. Improvement of K-means

There exist several variants of the k-means algorithm in the literature [15]. Some of them present the guidance on good initial centroids so that the algorithm is more likely to find the optimal result [16]. Another variation of the k-means algorithm is to select a different criterion function altogether. Diday [17] and Symon [18] propose the dynamic clustering algorithms which replace the centroid for each cluster by the framework of maximum-likelihood estimation.

Another variation involves the idea of splitting and merging after completing the k-means algorithm. The method is that a cluster is split when its variance is above a pre-defined threshold and two clusters are merged when the distance between their centroids is below another pre-defined threshold. This variant is possible to achieve the optimal partition starting from any arbitrary initial centroids. The condition is that the proper threshold values are demanded. The well known ISODATA algorithm [19] utilizes this technique of merging and splitting clusters. Therefore, if ISODATA is given the “ellipse” partition shown in Fig. 14 as an initial partition (just like the example we mention in previous section), ISODATA will first merge the clusters {A} and {B, C} into one cluster because their distance is small and then split {D, E, F, G} into two clusters {D, E} and {F, G} for the reason that {D, E, F, G} has large variance.

3.3. Mean Shift

3.3.1. The Concept of Mean Shift

Numerous nonparametric clustering methods can be separated into two parts: hierarchical clustering and density estimation. Hierarchical clustering composes either aggregation or division based on some proximate measure. The concept of the density estimation-based nonparametric clustering method is that the feature space can be considered as the experiential probability density function (p.d.f.) of the represented parameter. The mean shift algorithm [20], [21] can be classified as density estimation. It adequately analyzes feature space to cluster them and can provide reliable solutions for many vision tasks. The basics of mean shift are discussed as below.

Given n data points $\mathbf{x}_i$, $i = 1,\dots, n$ in the d-dimensional space R^d, the multivariate kernel density estimator with kernel $K(\mathbf{x})$ is

$$f(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right), \tag{3.4}$$

where h is one bandwidth parameter satisfying $h > 0$ and K is the radially symmetric kernels satisfying

$$K(\mathbf{x}) = c_{k,d}\,k\left(\|\mathbf{x}\|^2\right), \tag{3.5}$$

where $c_{k,d}$ is a normalization constant which induces $K(\mathbf{x})$ integrate to one. The function $k(x)$ is the profile of the kernel, only for $x \geq 0$.

Applying the profile notation, the density estimator (3.4) can be written as

$$f_{h,K}(\mathbf{x}) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right). \tag{3.6}$$

For analyzing a feature space with the density $f(x)$, we have to find the modes of this density. The modes are located among the zeros of the gradient $\nabla f(x) = 0$. The gradient of the density estimator (3.6) is

$$\nabla f_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}-\mathbf{x}_i)\,k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right). \tag{3.7}$$

We define the function $g(x) = -k'(x)$ and introduce $g(x)$ into (3.7) generates,

$$\begin{aligned}\nabla f_{h,K}(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x})\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\\ &= \frac{2c_{k,d}}{nh^{d+2}}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}-\mathbf{x}\right].\end{aligned} \tag{3.8}$$

The first term of (3.8) is proportional to the density estimate at x computed with the kernel $G(\mathbf{x}) = c_{g,d}\, g\left(\|\mathbf{x}\|^2\right)$ shown as

$$f_{h,G}(\mathbf{x}) = \frac{c_{g,d}}{nh^d} \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right). \tag{3.9}$$

The second term is the mean shift

$$\mathrm{m}_{h,G}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}, \tag{3.10}$$

Introducing (3.9) and (3.10) into (3.8) yields

$$\nabla f_{h,K}(\mathbf{x}) = f_{h,G}(\mathbf{x}) \frac{2c_{k,d}}{h^2 c_{g,d}} \mathrm{m}_{h,G}(\mathbf{x}), \tag{3.11}$$

Rewriting as

$$\mathrm{m}_{h,G}(\mathbf{x}) = \frac{1}{2} h^2 c \frac{\nabla f_{h,K}(\mathbf{x})}{f_{h,G}(\mathbf{x})}. \tag{3.12}$$

Therefore, it is obvious that the mean shift vector $\mathrm{m}_{h,G}(\mathbf{x})$ computed with kernel *G* is proportional to the normalized density gradient estimate obtained with kernel *K*. The mean shift vector always directs to the direction of maximum increase in the density and the procedure is guaranteed to converge at a nearby point where the estimate (3.6) has zero gradient.

The preceding discussion can be summarized by the following procedure.

Step1. Decide what feature space we want mean shift to consider and let every features be a vector. Then we construct *d* dimensions matrix. For example,

$$dataPts = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 4 & 1 & 7 & 9 \\ 4 & 5 & 1 & 2 & 6 & 7 \end{bmatrix}.$$

Step2. Randomly select a column to be an initial mean. For example,

$$\begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}.$$

Step3. Construct a matrix, which is the repeat of an initial mean and use this matrix to minus "*dataPts*". Then calculate the square of every components of the new matrix and individually sum every column to get a vector "*SqDistToAll*". For example,

$$SqDistToAll = \left\{ \begin{bmatrix} 4 & 4 & 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix} - dataPts \right\} .\wedge 2$$

$$= \begin{bmatrix} 9 & 4 & 1 & 0 & 1 & 4 \\ 4 & 16 & 9 & 0 & 36 & 64 \\ 4 & 9 & 1 & 0 & 16 & 25 \end{bmatrix} \xrightarrow{\text{Sum each column}} \begin{bmatrix} 17 & 29 & 11 & 0 & 53 & 93 \end{bmatrix}$$

Step4. From "*SqDistToAll*", find out the positions that their value are smaller than (bandwidth)^2. Store these positions in "*inInds*" and label these positions in "*beenVisitedFlag*" which represents the positions have been clustered.

Step5. Recompute the new mean among the value of "*inInds*".

Step6. Repeat Step3 to 5 until the mean is convergence. The convergence means the distance between previous mean and present mean is smaller than the threshold that we decide. Distance represents their mean square or the sum of their difference's square.

Step7. After convergence, we can cluster those labeled positions into the same cluster. But before clustering, we have to examine whether the distance between the new found mean and those old means is too close. If it happens, we should merge those labeled positions into the old mean's cluster.

Step8. Afterward eliminate those clustered data from "*dataPts*" and repeat Step2 to 7 until all of "*dataPts*" are clustered. Then the mean shift's clustering is finished.

We summarize the advantages and disadvantages of the mean shift algorithm as follows:

Advantages:

a. An extremely versatile tool for feature space analysis.

b. Suitable for arbitrary feature spaces.

Disadvantages:

a. The kernel bandwidth is the only factor can control the output.

b. The computation time is quite long.

3.3.2. Experimental Results and Discussion

The segmentation result of mean shift algorithm in Fig. 15 seems really match human sense except some fragmental regions. However, to consider the computation time, it is a time-consuming method.

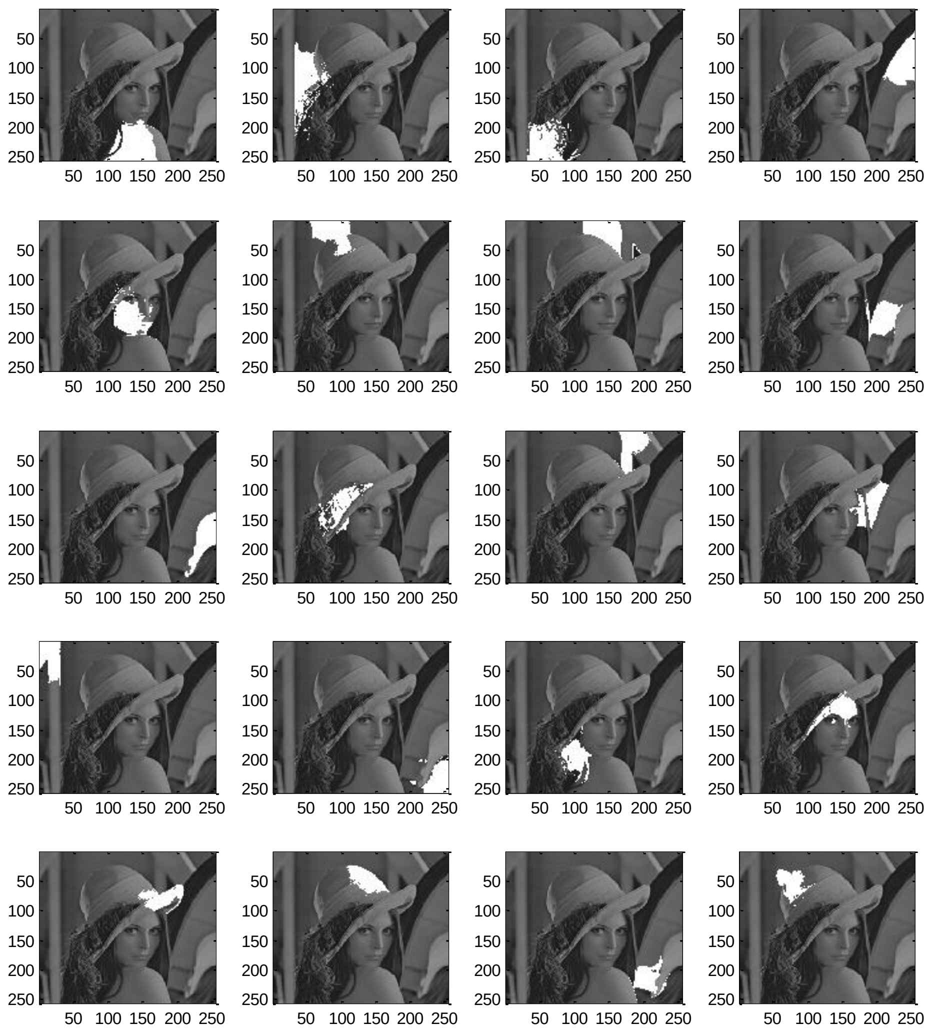


Fig. 15 The segmentation result of Lena image using the mean shift algorithm.

4. Edge-Based Segmentation Method

Edge-base segmentation generally indicates the segmentation method based on the edge in an image. The simple methods apply some edge detection methods before segmentation. Some edge detection methods are gradient operators [2] and Hilbert transform [22]. Then the other methods only base on the concept of edge instead of using edge detection methods, for instance, watershed segmentation algorithm.

4.1. Watershed Segmentation Algorithm

The main goal of watershed segmentation algorithm is to find the "watershed lines" in an image in order to separate the distinct regions. To imagine the pixel values of an image is a 3D topographic chart, where *x* and *y* denote the coordinate of plane, and *z* denotes the pixel value. The algorithm starts to pour water in the topographic chart from the lowest basin to the highest peak. In the process, we may detect some peaks disjoined the catchment basins, called as "dam". The diagram shows in Fig. 16.

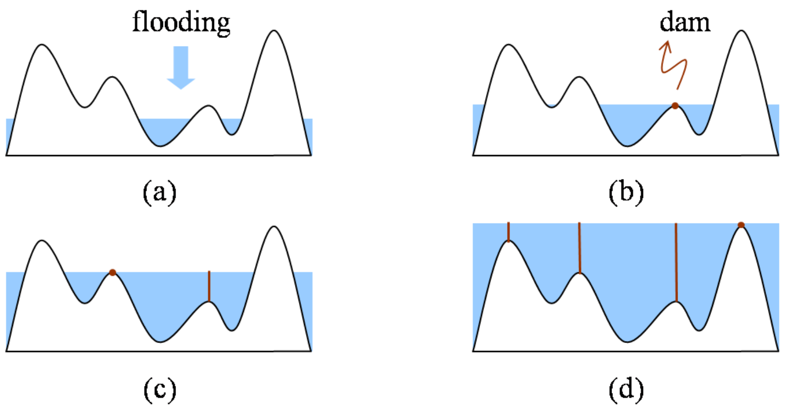


Fig. 16 The concept of watershed.

Before describing the steps of watershed, we previously define some parameters. Let M_1, M_2, …, M_R sets denoting the coordinates in the regional minima of an image $g(x, y)$, where $g(x, y)$ is the pixel value of coordinate (x, y). Denote $C(M_i)$ as the coordinates in the catchment basin associated with regional minimum M_i. Finally, let $T[n]$ be the set of coordinates (s, t) for which $g(s, t) <$ n and show as

$$T[n]=\{(s,t)\,|\,g(s,t)<n\}. \tag{4.1}$$

Then the process of watershed algorithm is discussed as below [1].

Step1. Find the minimum and maximum pixel value of $g(x, y)$ as min and max. Assign the coordinate of min into M_i. The topography will be flooded in integer flood increments from n = min +1. Let $C_n(M_i)$ as the coordinates in the catchment basin associated with minimum M_i that are flooded at stage *n*.

Step2. Compute

$$C_n(M_i)=C(M_i)\cap T[n]. \tag{4.2}$$

If $(x,y)\in C(M_i)$ and $(x,y)\in T[n]$, $C_n(M_i) = 1$ at location (x, y); otherwise $C_n(M_i) = 0$. Then let $C[n]$ denote the union of the flooded catchment basins at stage n:

$$C[n]=\bigcup_{i=1}^{R} C_n(M_i). \tag{4.3}$$

Set $n = n + 1$.

Step3. Derive the set of connected components in $T[n]$ denoting as Q. For each connected component $q \in Q[n]$, there are three conditions:

a. If $q \cap C[n-1]$ is empty, connected component q is incorporated into $C[n - 1]$ to form $C[n]$ because it represents a new minimum is encountered.

b. If $q \cap C[n-1]$ contains one connected component of $C[n - 1]$, connected component q is incorporated into $C[n - 1]$ to form $C[n]$ because it means q lies within the catchment basin of some regional minimum.

c. If $q \cap C[n-1]$ contains more than one connected component of $C[n - 1]$, it represents all or part of a ridge separating two or more catchment basins is encountered so that we have to find the points of ridge(s) and set them as "dam".

Step4. Construct $C[n]$ according to (4.2) and (4.3). Set $n = n + 1$.

Step5. Repeat Step 3 and 4 until n reaches max + 1.

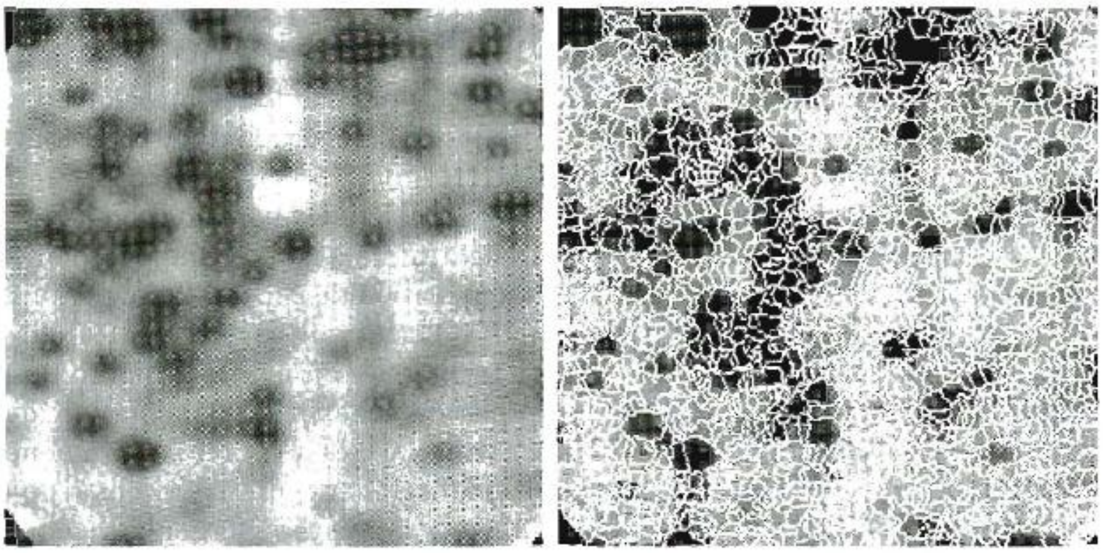


Fig. 17 (a) Electrophoresis image and (b) result of applying the watershed algorithm. [1]

We show the segmentation result of the watershed algorithm in Fig. 17 and list the advantages and disadvantages of the watershed algorithm as follows:

Advantages:

a. The boundaries of each region are continuous.

Disadvantages:

a. The segmentation result has over-segmentation problem, shown in Fig. 17(b).

b. The algorithm is time-consuming.

4.2. Markers

For resolving the over-segmentation problem in the watershed algorithm, an approach based on the concept of marker is described in [1]. A marker is a connected component belonging to an image. The markers include the internal markers, associated with objects of interest, and the external markers, associated with the background. The marker selection typically consists of two steps: preprocessing and definition of a set of criteria that markers must satisfy. The preprocessing scheme is to filter an image with a smoothing filter. This step can minimize the effect of small spatial detail, in other words, this step is to reduce the large number of potential minima (irrelevant detail), which is the reason of over-segmentation.

The definitions of an internal marker is

a. A region that is surrounded by points of higher “altitude”.

b. The points in the region form a connected component.

c. All the points in the connected component have the same intensity value.

After the image is smoothed, the internal markers can be defined by these definitions, shown as light gray, blob like regions in Fig. 18(a). Consequently, the watershed algorithm is applied to the smoothed image, under the restriction that these internal markers be the only allowed regional minima. Fig. 18(a) shows the watershed lines, defined as the external markers. The points of the watershed line are along the highest points between neighboring markers.

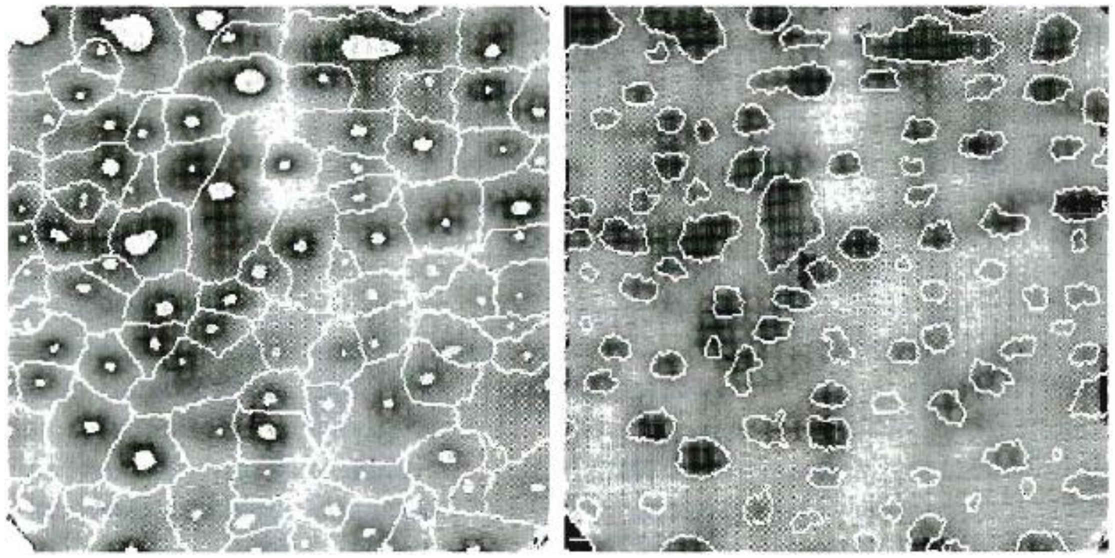


Fig. 18 (a) Image showing internal markers (light gray regions) and external markers (watershed lines). (b) Segmentation result of (a). [1]

The external markers effectively segment the image into several regions with each region composed by a single internal marker and part of the background. Then the goal is to reduce each of these regions into two: a single object and its background. The segmentation techniques discussed earlier can be applied to each individual region. Fig. 18(b) shows the segmentation result of applying the watershed algorithm to each individual region.

5. Segmentation Application

Image segmentation can be applied in variety of different fields, e.g. pattern recognition, image compression, and image retrieval. We introduce an example in muscle injury determination in Section 5.1. A modified JPEG image compression is described in Section 5.2.

5.1. Muscle Injury Determination by Segmentation

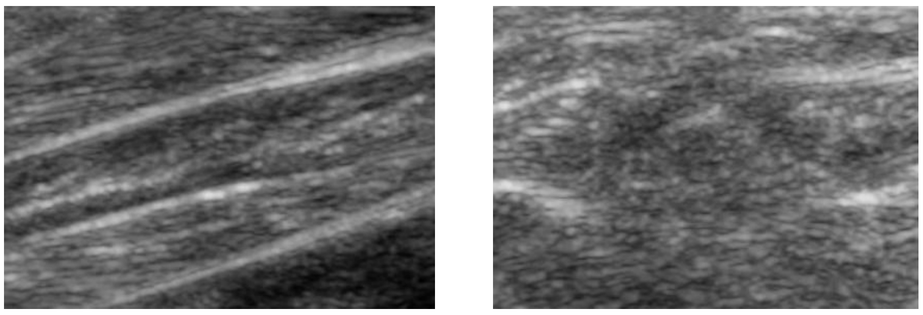

(a) (b)

Fig. 19 The ultrasonic images of (a) healthy muscle fibers and (b) unhealthy muscle fibers.

In this application, the ultrasonic image of the healthy muscle and unhealthy muscle are shown in Fig. 19, the health of muscle is defined by the fibrosis degree, the lower fibrosis degree muscle has, the healthier muscle will be. We use the fast scanning algorithm to segment the ultrasonic image of muscle fiber because the expectation of muscle injury determination is to obtain the shape of fiber. According to the segmentation result, the muscle injury determination methods can conveniently discriminate the health of muscle [24]. We describe the procedure of determination as follows.

Step1. Enhance the intensity contrast of the muscle image, shown in Fig. 20.

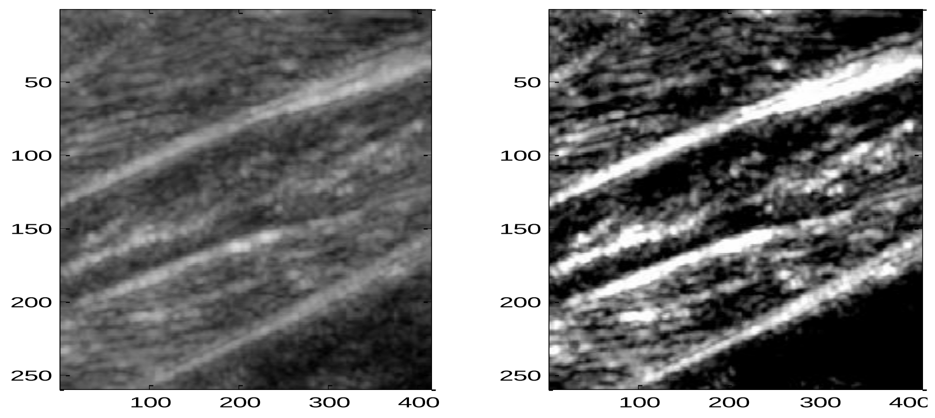


Fig. 20 The original image of healthy muscle and its enhanced gray level image.

Step2. Apply the fast scanning algorithm to segment the enhanced muscle image. The segmentation results are shown in .

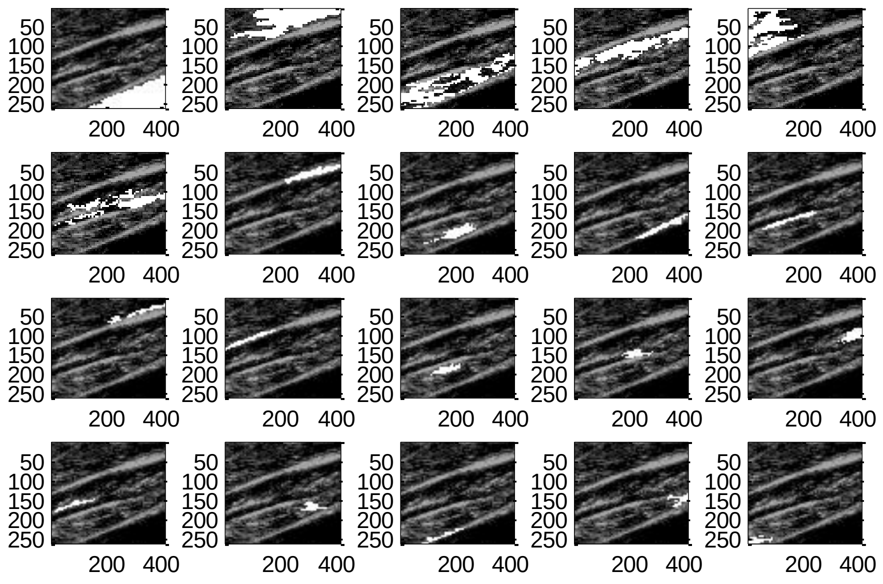


Fig. 21 The segmentation results of Fig. 20.

Step3. Sift the fiber-like clusters from the segmentation results of Step 2, shown in Fig. 22 and find out the healthy muscle fibers. The healthy muscle fiber is decided by the length of fiber, which must beyond to 80% of the width of the image.

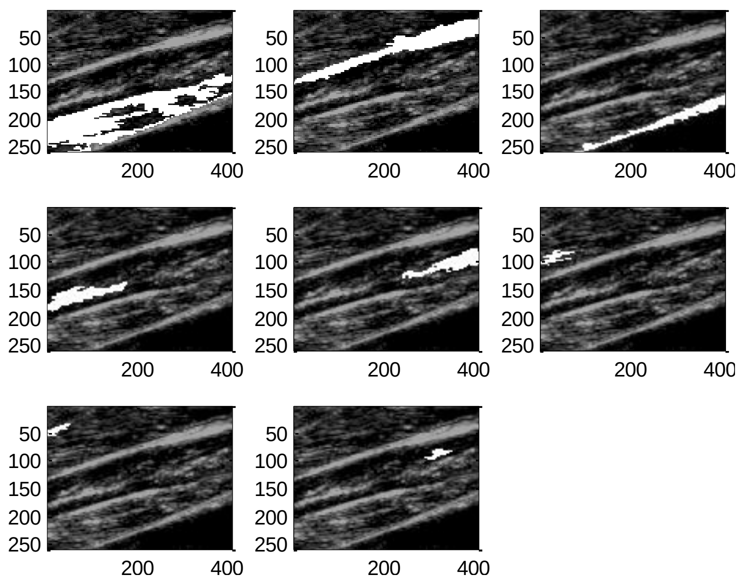


Fig. 22 Sifting the fiber regions from Fig. 21.

Step4. Find the unhealthy muscle fibers based on the distance between any two fibers. The distance between any two fibers considers both the horizontal distance and the vertical distance. The healthy and unhealthy muscle fibers are shown in Fig. 23.

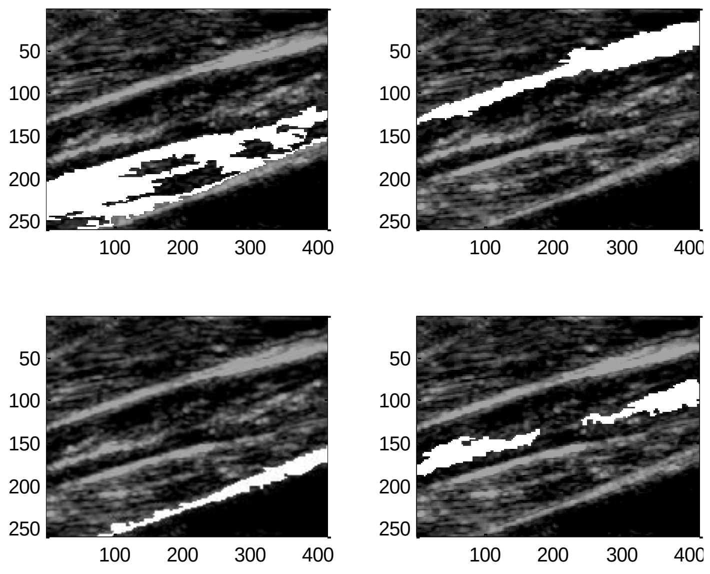


Fig. 23 Finding the healthy or broken fibers from Fig. 22.

Step5. Compute the injury score by the total number of fibers, summation of the broken length, and summation of the length of fibers. According to the injury score, we can estimate the degree of fibrosis and realize the health of muscle.

5.2. Modified JPEG Image Compression

Traditionally, JPEG uses the block-based image coding for compression, which may produce significantly blocking artifacts and distortions. In recent years, coding of arbitrarily shaped image region played an important role in many visual coding applications. The reason is that it employs the information of arbitrarily-shaped region to exploit the high correlation of the color values within the same image segment in order to achieve a higher compression ratio. However, the shape adaptive image coding traditionally relies on the Gram-Schmidt process the obtained orthonormal bases for an arbitrarily-shaped image segment, which takes high computation complexity. So an approach of object coding based on two dimensional orthogonal DCT expansion in triangular and trapezoid regions is proposed [25].

The preprocessing of this modified JPEG image compression is to apply the fast scanning algorithm to segment an image (certainly, other region-based segmentation algorithms can be applied here, it is up to the preference of user) and derive several high autocorrelation of the color values within the same segment. Then these segments are segmented to several triangular and trapezoid regions by the triangular and trapezoid dividing method proposed in [25]. Fig. 24 shows the trapezoid dividing result of an homogenous region. For each homogeneous triangular and trapezoid region, we find out its complete and orthonormal DCT basis. This basis can be applied to accomplish DCT coding and achieve the purpose of reducing spatial redundancy.

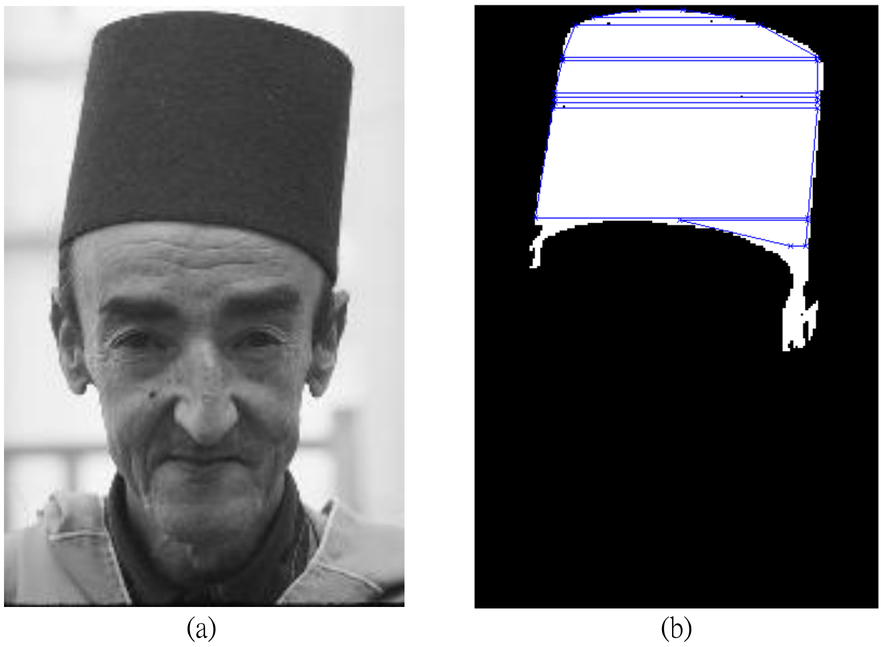


Fig. 24 (a) An example of a man and (b) the trapezoid dividing result of his hat.

6. Conclusion

In this tutorial, we survey several segmentation methods. For various applications, there are suitable segmentation methods that can be applied. If the requirement is that the pixels of each cluster should be linked, then region-based segmentation algorithms, especially, the JSEG and the fast scanning algorithms, are preferred, because of their better performances. If the requirement is to classify the whole image pixels but not consider the connection of cluster then data clustering, especially the k-means algorithm, is the better choice. The clustering result of mean shift is fine, but it costs much computation time. The edge-based segmentation method, especially watershed, has the over-segmentation problem, so we normally combine the marker tool and watershed for overcoming the over-segmentation problem.

Reference:

A. Digital Image Processing

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed., Prentice Hall, New Jersey 2008.

[2] W. K. Pratt, *Digital Image Processing*, 3th ed., John Wiley & Sons, Inc., Los Altos, California, 2007

B. Region-Based Segmentation Method

[3] R. Adams, and L. Bischof, "Seeded region growing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 6, pp. 641-647, June, 1994.

[4] Z. Lin, J. Jin and H. Talbot, "Unseeded region growing for 3D image segmentation," *ACM International Conference Proceeding Series*, vol. 9, pp. 31-37, 2000.

[5] S. L. Horowitz and T. Pavlidis, "Picture segmentation by a tree traversal algorithm," *JACM*, vol. 23, pp. 368-388, April, 1976.

[6] Y. Deng, and B.S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 8, pp. 800-810, Aug. 2001.

[7] Y. Deng, C. Kenney, M.S. Moore, and B.S. Manjunath, "Peer group filtering and perceptual color image quantization," *Proc. IEEE Int'l Symp. Circuits and Systems*, vol. 4, pp. 21-24, Jul. 1999.

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

[9] J. J. Ding, C. J. Kuo, and W. C. Hong, "An efficient image segmentation technique by fast scanning and adaptive merging," *CVGIP*, Aug. 2009.

C. Data Clustering

[10] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM Computing Surveys*, vol. 31, issue 3, pp. 264-323, Sep. 1999.

[11] W. B. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, Upper Saddle River, NJ, 13–27.

[12] G. Nagy, “State of the art in pattern recognition,” *Proc. IEEE*, vol. 56, issue 5, pp. 836–863, May 1968.

[13] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[14] J. MacQueen, *Some methods for classification and analysis of multivariate observations,* In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 281–297.

[15] M. R. Anderberg, *Cluster Analysis for Applications,* Academic Press, Inc., New York.

[16] S. Ray and R. H. Turi., “Determination of number of clusters in k-means clustering and application in colour image segmentation,” Presented at *4th Inter-national Conference on Advances in Pattern Recognition and Digital Techniques*(*ICAPRDT’99*), Dec 1999.

[17] E. Diday, “The dynamic cluster method in non-hierarchical clustering,” *J. Comput. Inf. Sci.*, vol. 2, pp. 61-88, 1973.

[18] M.J. Symons, “Clustering criteria and multivariate normal mixtures,” *Biometrics*, vol. 37, no. 1, pp. 35-43, Mar. 1981.

[19] G. H. Ball, and D. J. Hall, ISODATA, *A Novel Method of Data Analysis and Pattern Classification*, Menlo Park, CA: Stanford Res. Inst. 1965.

[20] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space Analysis,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.

[21] Y. Cheng, “Mean shift, mode seeking, and clustering,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790-799, Aug. 1995

D. Edge-Based Segmentation Method

[22] S. C. Pei and J. J. Ding, “The generalized radial Hilbert transform and its applications to 2-D edge detection (any direction or specified directions),” ICASSP 2003, vol. 3, pp. 357-360, Apr. 2003.

[23] L. Vincent, P. Soille, “Watersheds in digital spaces: an efficient algorithm based on immersion simulations,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, issue 6, pp. 583-598, Jun. 1991.

E. Application

[24] J. J. Ding, Y. H. Wang, L. L. Hu, W. L. Chao, and Y. W. Shau, “Muscle injury

determination by image segmentation," *CVGIP*, Aug. 2010.

[25] S. C. Pei, J. J. Ding, P. Y. Lin and T. H. H. Lee, "Two-dimensional orthogonal DCT expansion in triangular and trapezoid regions," *CVGIP*, Aug. 2009

F. Other Segmentation

[26] K. S. Fu, "A survey on image segmentation," *Pattern Recognition*, vol. 13, pp. 3–16, 1981.

[27] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer Vision Graphics Image Process.*, vol. 29, pp. 100–132, 1985.

[28] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277-1294, 1993.