NEW TECHNIQUES FOR DATA CLUSTERING

AND COLOR IMAGE SEGMENTATION

By

TRUNG HUY DUONG

Bachelor of Technology in Mechatronics
Hanoi University of Technology
Hanoi, Vietnam
2004

Master of Science in Mechanical and Aerospace Engineering
Oklahoma State University
Stillwater, Oklahoma
2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2013

NEW TECHNIQUES FOR DATA CLUSTERING

AND COLOR IMAGE SEGMENTATION

Dissertation Approved:

Dr. Lawrence L. Hoberock

Dissertation Adviser

Dr. Jay C. Hanan

Dr. Gary E. Young

Dr. Guoliang Fan

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor, Dr. Lawrence L. Hoberock, Professor and Head, School of Mechanical and Aerospace Engineering, Oklahoma State University, who has inspired and directed me in the fascinating research area of machine vision, always encouraging me to do my best, giving me a lot of flexibility, and generously providing me financial support. I would like to express my sincere thanks to Dr. Jay C. Hanan, Associate Professor, Dr. Gary E. Young, Professor, School of Mechanical and Aerospace Engineering, Oklahoma State University, and Dr. Guoliang Fan, Professor, School of Electrical and Computer Engineering, Oklahoma State University, for serving on my Graduate Committee.

I wish to express my special thanks to School of Mechanical and Aerospace Engineering, and New Product Development Center, Oklahoma State University for supporting a research assistantship to complete this work. I would also thank to University of Communication and Transport for providing me an opportunity to study in USA.

I would like to extend my thanks to my colleagues at Robotics Research Lab in Oklahoma State University, for their valuable ideas and discussions. Finally, I would like to thank my parents Mr. Lung H. Duong and Ms. Luc T. Vu and my relatives for their love and encouragement. I specially express my appreciation for my wife, Ms. Ly H. Doan, for her love, understanding, and helping me overcome difficulties. I would like to dedicate my work to my daughters, Windy Giang Duong and Sunny Giang Duong, who are my joy, inspiration, and future.

Name: TRUNG HUY DUONG

Date of Degree: DECEMBER, 2012

Title of Study: NEW TECHNIQUES FOR DATA CLUSTERING AND COLOR IMAGE SEGMENTATION

Major Field: MECHANICAL AND AEROSPACE ENGINEERING

Abstract: The objectives of this work are twofold: (1) to create an improved automatic clustering procedure that produces results consistent with manual clustering of data points by humans; and (2) to find an improved technique for automatic segmentation of images. First, we developed a clustering technique using an M-ART (Mahalanobis distance-based Adaptive Resonance Theory) neural network. The "vigilance" ρ in the M-ART network affects the maximum size of clusters, and consequently affects the number of clusters. Normally the "optimal" value of ρ is heavily data dependent and therefore can only be chosen by users and adjusted by trial-and-error. We propose a procedure to automatically adjust the value of ρ based on a pre-defined required separation between clusters, which is data independent and can be determined beforehand. Experiments conducted on synthetic multidimensional and texture datasets demonstrate the effectiveness and reliability of the proposed method. Segmentation is the process of partitioning a digital image into multiple segments or non-overlapping regions. Partitioning an image into non-overlapping regions assures that pixels in each region share the same visual properties, such as color or texture, while pixels in different regions exhibit significant differences in these features. We found that M-ART works well only with convex-shaped clusters (segments) that are sufficiently separated, which is not the case for typical real-scene images. Accordingly, we investigated and presented developing a more advanced general purpose image segmentation method, called the DUHO method. This DUHO algorithm contains two main steps. First, the superpixel generating algorithm is applied to a given image to build K superpixels. Then a new region growing algorithm iteratively groups these superpixels into appropriate regions and forms

the final image segmentation result. The proposed method is a type of unseeded region-based segmentation that preserves the spatial relationship between pixels in the image, and hence preserves the detailed edges and the image spatial structure. A quantitative evaluation method based on square color error is introduced, and experiments with real datasets, consisting of 300 color images of natural scenes from the available data, show very good results from our DUHO method when compared with results from the well-known segmentation methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF NOTATIONS

Bold	Indicates a vector or a matrix or a set
Normal	Indicates a scale number
$\mathbf{X}_k$	The k^{th} input vector, d-dimensional: $\mathbf{X}_k=[x_{k1},x_{k2},\ldots, x_{kd}]^T$
x_{ki}	The i^{th} element of the input vector $\mathbf{X}_k$
d	dimension of input vector space
i,k	The index for an input vector, $1\leq i,k\leq L$
L	The number of input vectors in a data set
S	The data set $\mathbf{S} = \{\mathbf{X}_k,k=1..L\}$
(**C**,**Q**,R,N)	An ellipsoid cluster with center vector **C**, covariance matrix **Q**, cluster size R, number of members N.
$\mathbf{C}_j$	The center of the cluster ($\mathbf{C}_j$, $\mathbf{Q}_j$, R_j,N_j), d-dimensional $\mathbf{C}_j=[c_{j1},c_{j2},\ldots, c_{jd}]^T$
$\mathbf{Q}_j$	The covariance matrix of the cluster ($\mathbf{C}_j$, $\mathbf{Q}_j$, R_j,N_j), symmetric and semi-definite positive of size d-by-d.
R_j	The radius of the cluster($\mathbf{C}_j$,R_j,N_j)
N_j	The number of members of the cluster ($\mathbf{C}_j$,R_j,N_j)
j,k	The index for a cluster, $1\leq j,k\leq M$
J	The index for the resonated clusters (closest to the current input), $1\leq J\leq M$
M	The number of current clusters in M-ART/E-ART/ART1network
M(), T()	The Match function and the Activation function

$\|.\|$	The Euclidean norm of a vector: $\|\mathbf{X}\|=\sqrt{\mathbf{X}^T\mathbf{X}}$
$\|.\|_1$	The 1-norm of a binary vector: $\|\mathbf{X}\|_1=\sum_{i=1}^{d}\|x_i\|$
$\|.\|_{\mathbf{A}}$	The **A**-norm of a vector: $\|\mathbf{X}\|_{\mathbf{A}}=\sqrt{\mathbf{X}^T\mathbf{A}\mathbf{X}}$
β	The learning rate
ρ	The vigilance of the M-ART/E-ART/ART1 network
SV	The desired separation ratio
SV_{jk}	The separation ratio between cluster j^{th} and cluster k^{th}

PART I: DATA CLUSTERING

Chapter 1

INTRODUCTION TO DATA CLUSTERING

1.1 Introduction

Clustering is a principal tool for data analysis that aims to produce natural groupings, or structure, in a given data set. Its wide application can be found in data mining, customer recommendation system, text document, image segmentation, sequence analysis, medical imaging, and crime analysis [2-9]. Interestingly, even though clusters in say a set of 2-D points laid out on an x-y axis system might be intuitively identified by most observers without pre-instruction , defining formally what constitutes a cluster is not only difficult, but may also be inaccurate. The most acceptable definitions typically arize from examples. According to Frank at el. [3], a partition resulting from clustering should have two properties: homogeneity within clusters (data belong to the same cluster should be as similar as possible) and heterogeneity between clusters (data belong to different clusters should be as different as possible).

1.2 Previous Work

A primary concern, and perhaps the most difficult, for a given data set is determining how many clusters are present. A second concern is to determine to which cluster a given data point belongs.

This second question could be relatively easily answered, once the correct number of clusters is known. Some frequently used clustering methods such as K-mean and Fuzzy c-means shown in Bezdek and Pal [10] require the number of clusters to be given a priori, but this is often not known. The "optimal" number of clusters could be chosen according to some criteria, such as cluster compactness or variation within a cluster and/or separation or isolation between clusters [11]. Cluster compactness (variation within cluster) and/or separation (isolation between clusters) are normally considered as major factors in forming validation indexes [10, 11]. Almost all clustering algorithms are not parameter-free and require user supplied values for input parameters. Determining these values is difficult, and is usually guided by trial-and-error. Moreover, the results produced could be very sensitive to these values, producing significantly different partition results with only slightly different parameter values [6], rendering them unusable.

Density based clustering method, such as DBSCAN [5], generally can handle arbitrary cluster shapes. However, there are two parameters that users must feed to DBSCAN: a maximum distance between points for which two points can be considered as neighbors and the minimum number of points required to form a cluster, which are difficult to choose a prior. "Optimal" values of these parameters are problem dependent, and can only be obtained by trail-and-error. In addition, the computational time required of DBSCAN is large without an indexing structure. The worst case time complexity of DBSCAN is $O(n^2)$ without indexing, and is O(nlogn) with spatial indexing [12], where n is the number of data point.

1.3 Proposed Approaches

In this paper, a clustering technique is introduced using an M-ART (Mahalanobis-based Adaptive Resonance Theory) neural network, in which Mahalanobis distance between data points is used as a metric. Similar to Kohonen's Learning Vector Quantization network (LVQ) and Reilly and Cooper's Restricted Coulomb Energy network (RCE), as in [13], M-ART uses hyper-ellipsoids to form training patterns into classes or clusters. During training, M-ART fixes the size (maximum size) of the hyper-

ellipsoid, while RCE fixes the position and LVQ fixes the number of clusters [8]. The control parameter called vigilance, ρ, in the M-ART network affects the maximum size of clusters, and consequently affects the number of clusters. Conventionally, the "optimal" value of ρ is heavily data dependent and therefore can only be chosen by users after trial-and-error. To overcome this shortcoming, we propose a procedure to auto-adjust the value of ρ based on a pre-defined allowed separation between clusters. This separation factor is data independent and can be determined beforehand. To assist M-ART in producing improved partitions, density-based removal pre-processing is introduced to remove noise and produce improved data separation.

In what follows, Chapter 2 presents our Mahalanobis distance-based ART algorithm (M-ART), applied to the clustering problem, and the procedure for auto-adjustment of ρ. Chapter 3 introduces the density-based removal pre-processing and our overall automatic clustering procedure. Chapter 5 and 6 present experimental results and conclusions, respectively.

Chapter 2

ART NEURAL NETWORK AND VARIATIONS

Adaptive Resonance Theory (ART), first introduced by Grossberg [14], is well known as an unsupervised neural network for self-organized stable, fast, incremental learning to recognize categories in response to arbitrary sequences of binary input vectors in real time. There are many variations and extensions of the ART network, such as ART2, ARTMAP, Fuzzy ART, and FARTMAP to deal with continuous inputs or extensions to supervised learning models [8, 14-16]. Essentially, any network based on ART forms input vectors (patterns) into separate categories (clusters) based on the similarity between them. The key idea is checking for similarity between the new input vector and the representatives of categories already learned. If there is a close enough match, the new vector is incorporated to the associated existing category. Otherwise, the ART network creates a new category to store this new pattern. In this way, previously learned memories are not eroded by new learning. ART directly addresses the Stability-Plasticity dilemma: “How can a system be receptive to significant new patterns and yet remain stable in response to irrelevant patterns?” [17].

2.1 ART1 Network:

The operation of the ART1 network originally introduced in [14] can be characterized by three steps: searching, vigilance testing, and learning. ART1 works only with binary input vectors $\mathbf{X}_k$ (containing element values of either 0 or 1): $\mathbf{X}_k=[x_{k1},x_{k2},\ldots, x_{kd}]^T$, $x_{ki} \in [2]$, i=1..d, where k is the index of the k^{th} input.

In the searching step, the existing cluster in the network that is most similar to the input pattern is found. The function that measures the similarity between two vectors (the input $\mathbf{X}_k$ and an existing j^{th} cluster center $\mathbf{C}_j$) is called the activation function $T(\mathbf{X}_k,\mathbf{C}_j)$ [18]:

$$T(\mathbf{X}_k, \mathbf{C}_j) = \frac{\|\mathbf{X}_k \cap \mathbf{C}_j\|_1}{\alpha + \|\mathbf{C}_j\|_1} \tag{2.1}$$

where $\cap$ is the bitwise AND operation, such that $\mathbf{X}\cap\mathbf{Y}=(x_1$ AND y_1, x_2 AND $y_2,\ldots, x_d$ AND $y_d)$, $\|\mathbf{X}\|_1 = \sum_{i=1}^{d}|x_i|$ is the number of ones in the vector $\mathbf{X}$ (or so called Manhattan norm, or 1-norm), and α is a small positive constant to avoid dividing by zero.

Let J denote the index that represents the cluster for which the activation is highest (highest similarity), given by:

$$J = \arg\max_j\left(T(\mathbf{X}_k, \mathbf{C}_j)\right) \tag{2.2}$$

A match function $M(\mathbf{X}_k,\mathbf{C}_J)$ measure the likeness of input $\mathbf{X}_k$ to this cluster J^{th}, given by:

$$M(\mathbf{X}_k, \mathbf{C}_J) = \frac{\|\mathbf{X} \cap \mathbf{C}_J\|_1}{\|\mathbf{X}\|_1} \tag{2.3}$$

In the vigilance testing step, the match function is compared with a dimensionless parameter called vigilance ρ to verify the match between the input and the most similar cluster. The condition for a good match, called the "resonance state", is:

$$M(\mathbf{X}_k,\mathbf{C}_J) > \rho \tag{2.4}$$

If (2.4) satisfied, then the input is incorporated into the J^{th} cluster (with highest activation). Otherwise, a new cluster is formed as the input itself. According to (2.3), $M(\mathbf{X}_k,\mathbf{C}_J)$ always lies

between 0 and 1, so we choose $0<\rho\leq 1$. The value of vigilance ρ determines the "coarseness" of the clusters created by the input vectors. With this ART1 network, a small value for ρ means more input vectors are classified into the same cluster, resulting in a small number of clusters. On the other hand, a large value for ρ yields a large number of clusters.

In the learning step, the J^{th} cluster that resonates with an input $\mathbf{X}_k$ is updated by:

$$\mathbf{C}_J^{new} = \beta(\mathbf{X}_k \cap \mathbf{C}_J^{old}) + (1-\beta)\mathbf{C}_J^{old} \quad (2.5)$$

where β is the learning rate, $0\leq\beta\leq 1$, and $\mathbf{C}_J^{old}$ and $\mathbf{C}_J^{new}$ are respectively the center of J^{th} cluster before and after adding the input $\mathbf{X}_k$. The learning process with $\beta = 1$ is called "fast learning", which minimizes the training time, but could lead to unstable results [15, 17].

2.2 Euclidean Distance-based ART (E-ART) Network:

A Euclidean distance-based ART network (E-ART) [16, 19, 20], designed to cluster analog pattern inputs, has several differences from ART1 discussed above. First, this network works with continuous, rather than binary, inputs. Before being fed individually to the network, each d-dimensional input vector $\mathbf{X}_k=[x_{k1},x_{k2},\ldots, x_{kd}]^T$ is normalized, producing $\mathbf{X}_n$, such that each element x_{ki}, i=1..d, is in the range [0,1]:

$$\mathbf{X}_n = (\mathbf{X}_k - \mathbf{X}_{min}) ./ (\mathbf{X}_{max} - \mathbf{X}_{min}) \quad (2.6.a)$$

Where ./ means an element-by-element division of two vectors, $\mathbf{X}_n$ is the normalized vector, and $\mathbf{X}_{min}$ ($\mathbf{X}_{max}$) is a new vector in which each element is the minimum (maximum) over all corresponding elements of all L input vectors, namely [17]:

$$\mathbf{X}_{min} = \left[\min_{1\leq k\leq L}(x_{k1}), \min_{1\leq k\leq L}(x_{k2}), \ldots, \min_{1\leq k\leq L}(x_{kd})\right]^T \quad (2.6.b)$$

$$\mathbf{X}_{max} = \left[\max_{1\leq k\leq L}(x_{k1}), \max_{1\leq k\leq L}(x_{k2}), \ldots, \max_{1\leq k\leq L}(x_{kd})\right]^T \quad (2.6.c)$$

Second, similar to [20], we set both the activation function and the match function as the Euclidean distances between the normalized input vector $\mathbf{X}_n$ and the center of each cluster $\mathbf{C}_j$, namely:

$$T(\mathbf{X}_n, \mathbf{C}_j) = M(\mathbf{X}_n, \mathbf{C}_j) = \|\mathbf{X}_n - \mathbf{C}_j\| = \sqrt{(\mathbf{X}_n - \mathbf{C}_j)^T(\mathbf{X}_n - \mathbf{C}_j)} \quad (2.7)$$

where $\|\mathbf{a}\|$ is the Euclidean norm of the vector $\mathbf{a}$.

Third, we define the cluster J that is most similar to the normalized input $\mathbf{X}_n$ as the one with the smallest Euclidean distance, given by:

$$J = \arg\min_j(\|\mathbf{X}_n - \mathbf{C}_j\|) \quad (2.8)$$

Fourth, define the resonance state by:

$$M(\mathbf{X}_n, \mathbf{C}_J) < \rho \quad (2.9)$$

where vigilance ρ is a pre-defined dimensionless real number in range $[0,\sqrt{d}]$. ($\rho \leq \sqrt{d}$ because each element of $\mathbf{X}_n$ and $\mathbf{C}_J$ is in the range [0,1] and therefore $\|\mathbf{X}_n - \mathbf{C}_j\| \leq \sqrt{d}$).

Notice that (2.8) and (2.9) are used differently from (2.2) and (2.4). In (2.8) the highest similarity is defined as the minimized activation function, rather than maximized as in (2.2). The resonance state in (2.9) occurs when the match function is small enough (less than ρ), rather than when it is large enough (greater than ρ) in (2.4). Accordingly, an input will be incorporated into the nearest cluster only if the Euclidean distance from it to this cluster center is small enough. Otherwise, this input forms a new cluster in the E-ART network. It is thus straightforward that the E-ART network classifies input vectors into clusters with a hyper-sphere shape. Furthermore, if the Euclidean distance from a new input vector to the nearest cluster center is larger than ρ, this input does not belong to this cluster. In E-ART, the vigilance ρ can be considered as the maximum allowable hyper radius of the cluster.

Because inputs are normalized before being fed to E-ART, in the rest of this study, we assume that any vector input $\mathbf{X}_k$ already has its elements in the range of [0,1].

The learning rule for E-ART is the same as for ART1 as described in (2.5). As shown in Fig.2.1, if (2.9) is satisfied (resonance occurs), the new input $\mathbf{X}_k$ is incorporated into the nearest cluster

(the J[th] cluster); and the cluster center $\mathbf{C}_J$ is updated by moving it toward $\mathbf{X}_k$. Other existing clusters are unchanged.

With $\mathbf{C}_J^{old}$ is the center of the J[th] cluster consisting of N previously submitted patterns, then:

$$\mathbf{C}_J^{old} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{X}_k \tag{2.10}$$

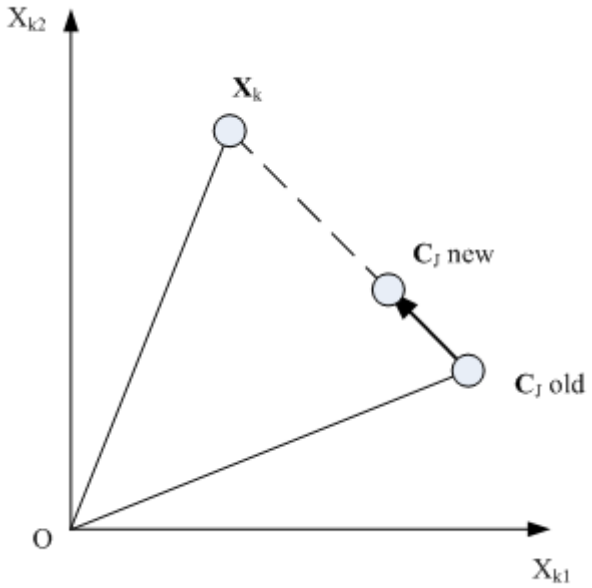


Figure 2.1: Graphical Representation of Learning Rule for 2-Dimendional Vectors

We update the center when adding a new pattern $\mathbf{X}_{N+1}$ such that the $\mathbf{C}_J^{new}$ becomes the center of the new cluster containing (N+1) patterns by using:

$$\mathbf{C}_J^{new} = \frac{1}{N+1}\sum_{k=1}^{N+1} \mathbf{X}_k = \frac{\mathbf{X}_{N+1} + N.\mathbf{C}_J^{old}}{N+1} \tag{2.11}$$

Therefore from (2.5) and (2.11), the learning rate should be:

$$\beta = \frac{1}{N+1} \tag{2.12}$$

2.3 Mahalanobis Distance-based ART (M-ART) Network

As described in Section 2.2, E-ART limits output clusters to hyper-sphere shapes, which is not sufficient to handle most real data. We propose a new Mahalanobis distance-based ART network,

which is an extension of the E-ART network to handle clustering with hyper-ellipsoid shaped clusters. M-ART operates very similarly to E-ART, except that the distance metric used in M-ART is the Mahalanobis distance [21].

The M-ART network also works with normalized continuous inputs as in (2.6). As suggested in [22], we set both the activation function and the match function as the Mahalanobis distances, defined in [21], between the normalized input vector $\mathbf{X}_n$ and the center of each cluster $\mathbf{C}_j$, employing the covariance matrix $\mathbf{Q}_j$ of the corresponding cluster, which produces activation $T(\mathbf{X}_n,\mathbf{C}_j)$ and match $M(\mathbf{X}_n,\mathbf{C}_j)$ functions given by: $\|\mathbf{X}_k - \mathbf{C}\|_{\mathbf{Q}_j^{-1}} = \sqrt{(\mathbf{X}_k - \mathbf{C})^T \mathbf{Q}_j^{-1} (\mathbf{X}_k - \mathbf{C})}$

$$T(\mathbf{X}_n, \mathbf{C}_j) = M(\mathbf{X}_n, \mathbf{C}_j) = \|\mathbf{X}_n - \mathbf{C}_j\|_{\mathbf{Q}_j^{-1}} = \sqrt{(\mathbf{X}_n - \mathbf{C}_j)^T \mathbf{Q}_j^{-1} (\mathbf{X}_n - \mathbf{C}_j)} \tag{2.13}$$

The covariance matrix $\mathbf{Q}_j$ is a semi-definite positive matrix, which can be estimated as shown in Chapter 3, Eqn (3.2). Notion $\|.\|_{\mathbf{A}}$ is a norm related to a matrix **A**, as in [23].

According to [23, 24] a covariance matrix $\mathbf{Q} \in \Re^{dxd}$, which is real, symmetric, and non-singular, can be decomposed as $\mathbf{Q} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \sum_{i=1}^{d} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$, where **U** is orthornomal, ($\mathbf{U}^{-1} = \mathbf{U}^T$), whose each column $\mathbf{u}_i$ is an eigenvector of **Q**, and **Λ** is diagonal matrix contained eigenvalues of **Q**, **Λ=** diag($\lambda_1, \lambda_1, \ldots \lambda_d$). The inverse of the covariance matrix can then be computed as:

$$\mathbf{Q}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T = \sum_{i=1}^{d} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T \tag{2.14}$$

The square of the Mahalanobis distance from **X** to the cluster center **C** employing the cluster covariance matrix **Q** can be computed as:

$$\|\mathbf{X} - \mathbf{C}\|_{\mathbf{Q}^{-1}}^2 = (\mathbf{X} - \mathbf{C})^T \mathbf{Q}^{-1} (\mathbf{X} - \mathbf{C}) = (\mathbf{X} - \mathbf{C})^T \left(\sum_{i=1}^{d} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T \right) (\mathbf{X} - \mathbf{C}) = \sum_{i=1}^{d} \frac{y_i^2}{\lambda_i} \tag{2.15}$$

where $y_i^2 = \mathbf{u}_i^T (\mathbf{X} - \mathbf{C})$

We can interpret y_i as a new coordinate system defined by the orthonormal vectors $\mathbf{u}_i$. As shown in Fig.2.2, the Mahalanobis distance from point **X** to a cluster center **C** is equivalent to the

normalized (or weighted) Euclidean distance from this point to the center in the new coordinate system defined by the orthonormal eigenvectors of the cluster covariance matrix. Along each eigenvector direction, the distance is weighted by the inverse square root of the corresponding eigenvalues.

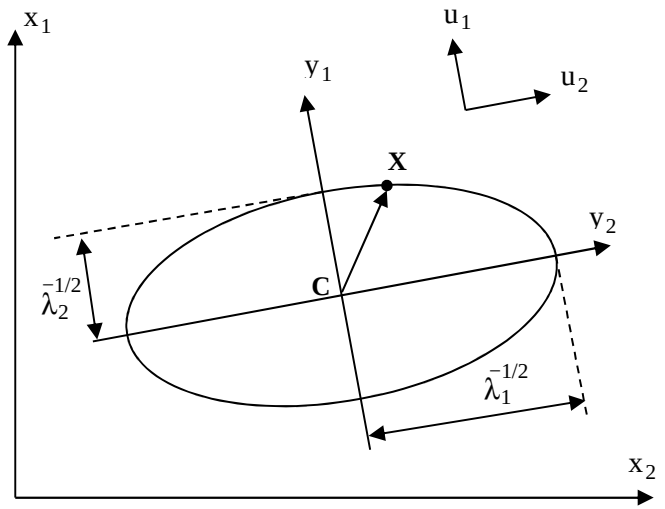
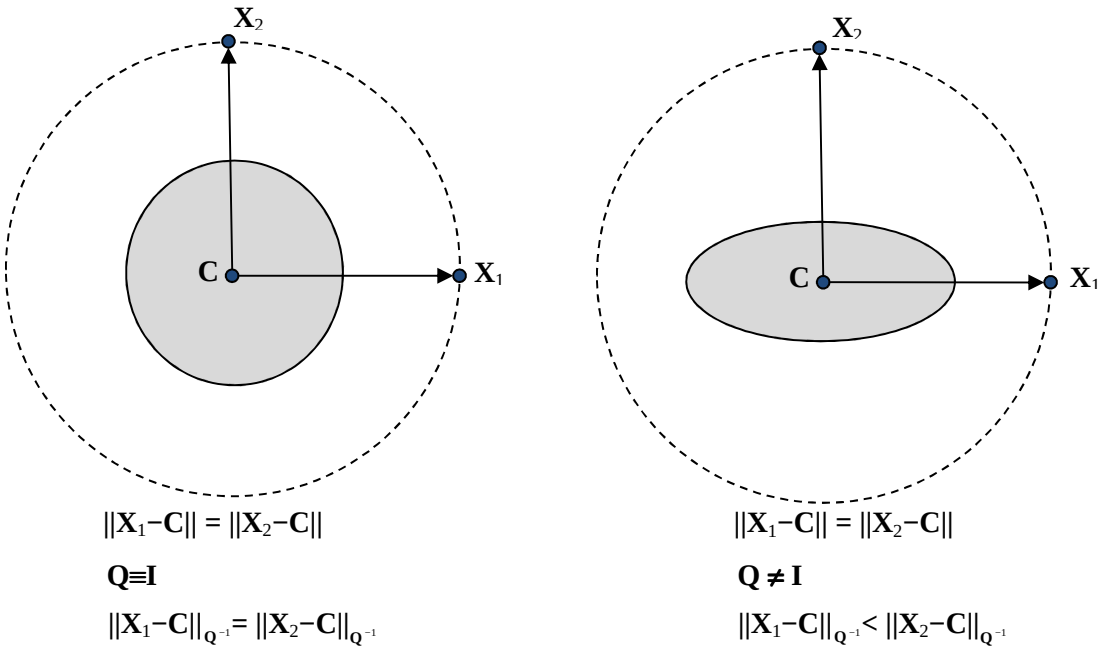


Figure 2.2: Mahalanobis Distance from the Point **X** and the Cluster Center **C**.

The Mahalanobis distance is used as a similarity measurement between the point **X** and the center of the cluster **C**, which indicates how likely this new point should belong to this cluster, a set of known points. The Mahalanobis distance differs from Euclidean distance, which is isotropic and does not depend on the distribution of the cluster data points. The Mahalanobis distance puts high weights along axes with high variance (major axes) of the cluster data points, while lower weights are placed along axes with low variance (minor axes). In other words, the Mahalanobis distance takes into account the correlations of the data set. If the covariance matrix of the cluster data points is the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. If the covariance matrix is diagonal, then the resulting distance measure is called the normalized Euclidean distance [18, 21]. Figure 2.3 illustrates the difference between the Mahalanobis and Euclidean distances from points $\mathbf{X}_1$ and $\mathbf{X}_2$ to the center **C** of a cluster. The Euclidean distances from $\mathbf{X}_1$ to **C** and from $\mathbf{X}_2$ to **C** are equal and do not depend on the shape of the cluster. However,

the Mahalanobis distances from these points to **C** depend on the shape of cluster (or its covariance matrix **Q**). If the cluster has elliptical shape, which means its covariance matrix **Q** is not the identity matrix, the Mahalanobis distance from $\mathbf{X}_1$ to **C** is less than the Mahalanobis distance from $\mathbf{X}_2$ to **C** (see Fig.2.3 b). If the cluster has a circular shape, which means its covariance matrix **Q** is the identity matrix, the Mahalanobis distance is the same as the Euclidean distance (see Fig.2.3 a).



(a): Euclidian Distance (b): Mahalanobis Distance

Figure 2.3: Difference between Mahalanobis Distance and Euclidean Distance.

Define the cluster J that is most similar to the normalized input $\mathbf{X}_n$ as the one with the smallest activation function, or smallest Mahalanobis distance, given by:

$$J = \arg\min_j \left(\left\| \mathbf{X}_n - \mathbf{C}_j \right\|_{\mathbf{Q}_j^{-1}} \right) \tag{2.16}$$

Finally, define the resonance state by:

$$M(\mathbf{X}_n, \mathbf{C}_J) < \rho \tag{2.17}$$

where vigilance ρ is a pre-defined, dimensionless, non-negative real number.

The resonance state in (2.17) occurs when the match function is small enough (less than ρ). Accordingly an input will be incorporated into the nearest cluster only if the Mahalanobis distance from it to this cluster center is small enough. Otherwise, this input forms a new cluster in the M-ART network. By the nature of the Mahalanobis distance, the M-ART network classifies input vectors into clusters with a hyper-ellipsoid shape. Furthermore, if the Mahalanobis distance from an input vector to the nearest cluster center is larger than ρ, this input does not belong to this cluster. Therefore, in M-ART, the vigilance ρ can be considered as the maximum allowable size of the cluster.

The update rule for the J^{th} center of M-ART is the same as for E-ART as described in (2.11). Other parameters of the J^{th} cluster, such as the covariance matrix and the size of this cluster, must also be updated, and will be discussed in Chapter 3.

Chapter 3

NEW DATA CLUSTERING METHOD

In this chapter we introduce our formal definition of clusters as hyper-ellipsoid shapes, define the M-ART clustering procedure, and introduce lemmas on results from applying M-ART.

3.1 Definitions of Hyper-ellipsoid Shaped Clusters and the M-ART Clustering Procedure

In what follows, we consider a cluster as a hyper-ellipsoid shape. Hyper-spheroid clusters are treated as special cases of hyper-ellipsoid clusters.

Definition 3.1: A Hyper-ellipsoid Shaped Cluster

Let **S** *be a data set consisting of a number* L *of* d*-dimensional vectors* $\mathbf{X}_k$. $\mathbf{X}_k \in \mathbf{S} \subset \Re^d$, k=1..L. *A hyper ellipsoidal shaped cluster with its center* $\mathbf{C} \in \Re^d$, *covariance matrix* $\mathbf{Q} \in \Re^{dxd}$, *maximum Mahalanobis distance, or cluster size,* $R \in \Re^+$, *and number of members* $N \leq L$, *denoted by* (**C**,**Q**,R,N), *is defined as:*

$$(\mathbf{C}, \mathbf{Q}, R, N) = \{\mathbf{X}_k \in \mathbf{S} : \|\mathbf{X}_k - \mathbf{C}\|_{\mathbf{Q}^{-1}} \leq R\} \quad (3.1)$$

where the center $\mathbf{C} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{X}_k$, $\mathbf{X}_k \in (\mathbf{C}, \mathbf{Q}, R, N), k = 1..N$, *is the mean of all vectors in* **S**, *the covariance matrix* **Q** *is a semi-definite positive matrix computed by:* (3.2)

$$\mathbf{Q} = \frac{1}{N-1}\sum_{k=1}^{N}(\mathbf{X}_k - \mathbf{C})(\mathbf{X}_k - \mathbf{C})^T$$ *,(unbiased estimation [23]),*

.

and $\|\mathbf{X}_k - \mathbf{C}\|_{\mathbf{Q}^{-1}} = \sqrt{(\mathbf{X}_k - \mathbf{C})^T \mathbf{Q}^{-1} (\mathbf{X}_k - \mathbf{C})}$ *is the Mahalanobis distance from the vector* $\mathbf{X}_k$ *to the vector (cluster center)* $\mathbf{C}$ *employing the covariance matrix* $\mathbf{Q}$*, where* $(.)^{-1}$ *and* $(.)^T$ *are the matrix inverse and transpose operations, respectively.*

Hence, it is evident that if $N > N_0$ *then* $\exists\, \mathbf{X}_k \in (\mathbf{C},\mathbf{Q},R,N)$: $\|\mathbf{X}_k - \mathbf{C}\|_{\mathbf{Q}^{-1}} = R$ (3.3)

The algorithm for M-ART is described by the following.

Definition 3.2: M-ART Clustering

1. *If* $\mathbf{X}_k$ *is the first input vector to M-ART , then the network produces one cluster* $(\mathbf{X}_k,\mathbf{I},0,1)$*, where* $\mathbf{I}$ *is a d-by-d identity matrix.*
2. *Assume there exists* $M \geq 1$ *clusters* $(\mathbf{C}_j, \mathbf{Q}_j, R_j, N_j)$, $j = 1..M$*, in an M- ART system. A new input vector* $\mathbf{X}_k$ *as an input to this system will be classified as a member of either cluster described in a. or b. bellow:*
   a. *an existing cluster* $(\mathbf{C}_J, \mathbf{Q}_J, R_J, N_J)$ *if and only if the two conditions (3.4) and (3.5) are satisfied:*

$$J = \arg\min_{1 \leq j \leq M} \left(\|\mathbf{X}_n - \mathbf{C}_j\|_{\mathbf{Q}_j^{-1}} \right), \tag{3.4}$$

$$\|\mathbf{X}_n - \mathbf{C}_J\|_{\mathbf{Q}_J^{-1}} \leq \rho, \tag{3.5}$$

where ρ *is the vigilance parameter.*

If (3.4) and (3.5) are satisfied, then that J^{th} *cluster will be updated by:*

$$N_J^{new} = N_J^{old} + 1, \tag{3.6}$$

$$\mathbf{C}_J^{new} = \frac{\mathbf{X}_k + N_J^{old} . \mathbf{C}_J^{old}}{N_J^{old} + 1}, \tag{3.7}$$

$$\mathbf{Q}_J^{new} = \frac{N_J^{old} - 1}{N_J^{old}} \mathbf{Q}_J^{old} + \frac{1}{N_J^{old} + 1} (\mathbf{X}k - \mathbf{C}_J^{old})(\mathbf{X}k - \mathbf{C}_J^{old})^T, \tag{3.8}$$

$$R_J^{new} = \max\left(R_J^{old}, \ \|\mathbf{X}_k - \mathbf{C}_J^{new}\|_{(\mathbf{Q}_J^{new})^{-1}} \right). \tag{3.9}$$

   b. *otherwise, a new cluster* $(\mathbf{X}_k,\mathbf{I},0,1)$.

Iterative updating of cluster center, covariance matrix, and inverse of covariance matrix:

Assume a cluster consists of N points: $\mathbf{X}_1,\mathbf{X}_2,\ldots,\mathbf{X}_N$, whose center, $\mathbf{C}_N$, and covariance matrix, $\mathbf{Q}_N$, are defined in (3.2) and will be rewritten, with extra subscripts indicating the current number of points, as follows:

$$\mathbf{C}_N = \frac{1}{N}\sum_{k=1}^{N}\mathbf{X}_k, \tag{3.10}$$

$$\mathbf{Q}_N = \frac{1}{N-1}\sum_{k=1}^{N}(\mathbf{X}_k - \mathbf{C}_N)(\mathbf{X}_k - \mathbf{C}_N)^T. \tag{3.11}$$

Update the cluster center:

When a new point $\mathbf{X}_{N+1}$ is added to a cluster, the new cluster center will be given by:

$$\boxed{\mathbf{C}_{N+1} = \frac{1}{N+1}\sum_{k=1}^{N+1}\mathbf{X}_k = \frac{1}{N+1}\left(\mathbf{X}_{N+1} + \sum_{k=1}^{N}\mathbf{X}_k\right) = \frac{\mathbf{X}_{N+1} + N\mathbf{C}_N}{N+1},} \tag{3.12}$$

which is equivalent to (3.7).

Update the cluster covariance matrix:

The new cluster covariance matrix is defined by:

$$\mathbf{Q}_{N+1} = \frac{1}{N}\sum_{k=1}^{N+1}(\mathbf{X}_k - \mathbf{C}_{N+1})(\mathbf{X}_k - \mathbf{C}_{N+1})^T \tag{3.13a}$$

$$= \frac{1}{N}\sum_{k=1}^{N}(\mathbf{X}_k - \mathbf{C}_{N+1})(\mathbf{X}_k - \mathbf{C}_{N+1})^T + \frac{1}{N}(\mathbf{X}_N + 1 - \mathbf{C}_{N+1})(\mathbf{X}_N + 1 - \mathbf{C}_{N+1})^T \tag{3.13b}$$

We would like to express the new covariance matrix, $\mathbf{Q}_{N+1}$, in term of its predecessor, $\mathbf{Q}_N$. Substituting the right side of (3.12) into the first term on the right of (3.13a) and (3.13b), we have:

$$\frac{1}{N}\sum_{k=1}^{N}(\mathbf{X}_k - \mathbf{C}_{N+1})(\mathbf{X}_k - \mathbf{C}_{N+1})^T = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{X}_k - \frac{\mathbf{X}_{N+1} + N\mathbf{C}_N}{N+1})(\mathbf{X}k - \frac{\mathbf{X}_{N+1} + N\mathbf{C}_N}{N+1})^T \tag{3.14a}$$

$$= \frac{1}{N}\sum_{k=1}^{N}\left[(\mathbf{X}_k - \mathbf{C}_N) + \frac{1}{N+1}(\mathbf{C}_N - \mathbf{X}_{N+1})\right]\left[(\mathbf{X}_k - \mathbf{C}_N) + \frac{1}{N+1}(\mathbf{C}_N - \mathbf{X}_{N+1})\right]^T \tag{3.14b}$$

$$= \frac{1}{N}\sum_{k=1}^{N}[(\mathbf{X}_k - \mathbf{C}_N)(\mathbf{X}k - \mathbf{C}_N)^T] + \frac{1}{N}\sum_{k=1}^{N}\left[\frac{1}{(N+1)^2}(\mathbf{C}_N - \mathbf{X}_{N+1})(\mathbf{C}_N - \mathbf{X}_{N+1})^T\right]$$

$$+\frac{1}{N}\sum_{k=1}^{N}\left[\frac{1}{N+1}(\mathbf{X}_k-\mathbf{C}_N)(\mathbf{C}_N-\mathbf{X}_{N+1})^T\right]+\frac{1}{N}\sum_{k=1}^{N}\left[\frac{1}{N+1}(\mathbf{C}_N-\mathbf{X}_{N+1})(\mathbf{X}_k-\mathbf{C}_N)^T\right] \quad (3.14c)$$

$$=\frac{N-1}{N}\mathbf{Q}_N+\frac{1}{(N+1)^2}(\mathbf{X}_{N+1}-\mathbf{C}_N)(\mathbf{X}_{N+1}-\mathbf{C}_N)^T \quad (3.14d)$$

Notice on the right side of (3.13c) that the first term is similar to $\mathbf{Q}_N$, the second term is a sum of constants, and the third and the fourth terms are both equal to d-by-d zero matrices. This is from (3.10) we have: $N\mathbf{C}_N = \sum_{k=1}^{N}\mathbf{X}_k$, or $\sum_{k=1}^{N}(\mathbf{X}_k-\mathbf{C}_N) = \mathbf{0}$ (d-by-1 vector zero), and $\sum_{k=1}^{N}(\mathbf{X}_k-\mathbf{C}_N)^T = \mathbf{0}^T$ (1-by-d vector zero).

Similarly, by substituting from (3.12) into the second term on the right of (3.13b), we obtain:

$$\frac{1}{N}(\mathbf{X}_{N+1}-\mathbf{C}_{N+1})(\mathbf{X}_{N+1}-\mathbf{C}_{N+1})^T=\frac{1}{N}(\mathbf{X}_{N+1}-\frac{\mathbf{X}_{N+1}+N\mathbf{C}_N}{N+1})(\mathbf{X}_{N+1}-\frac{\mathbf{X}_{N+1}+N\mathbf{C}_N}{N+1})^T \quad (3.15a)$$

$$=\frac{N}{(N+1)^2}(\mathbf{X}_{N+1}-\mathbf{C}_N)(\mathbf{X}_{N+1}-\mathbf{C}_N)^T \quad (3.15b)$$

Substituting from (3.14d) and (3.15b) for the terms on the right of (3.13b) yields:

$$\boxed{\mathbf{Q}_{N+1}=\frac{N-1}{N}\mathbf{Q}_N+\frac{1}{N+1}(\mathbf{X}_{N+1}-\mathbf{C}_N)(\mathbf{X}_{N+1}-\mathbf{C}_N)^T} \quad (3.16)$$

Update the inversion of the cluster covariance matrix:

In Def.3.2, we must compute the Mahalanobis distance from a new point to existing cluster centers, which involves the inversion of the covariance matrix. Therefore, it would be computationally efficient to derive an iterative formula for this inversion matrix, so that the inversion matrix operation need not be directly computed each time a new point is added.

The Woodbury identity, or the matrix inversion lemma, [24], is given by:

$$(\mathbf{A}+\mathbf{U}\mathbf{\Lambda}\mathbf{V})^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{U}(\mathbf{\Lambda}^{-1}+\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1} \quad (3.17)$$

where $\mathbf{A}$ is an n-by-n non-singular matrix, $\mathbf{\Lambda}$ is a k-by-k non-singular matrix, $\mathbf{U}$ is an n-by-k matrix, and $\mathbf{V}$ is a k-by-n matrix.

For the special case of k = 1, $\mathbf{\Lambda}$ becomes a scalar λ, $\mathbf{U}$ becomes a column vector $\mathbf{u}$, $\mathbf{V} = \mathbf{u}^T$, and $\mathbf{A}$ is symmetric, such that (3.17) reduces to:

$$(\mathbf{A} + \lambda\mathbf{u}\mathbf{u}^{T})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{u}\,(1/\lambda + \mathbf{u}^{T}\mathbf{A}^{-1}\mathbf{u})^{-1}\,\mathbf{u}^{T}\mathbf{A}^{-1} = \mathbf{A}^{-1} - \gamma(\mathbf{A}^{-1}\mathbf{u})\,(\mathbf{A}^{-1}\mathbf{u})^{T} \tag{3.18}$$

where $\gamma = 1/(1/\lambda + \mathbf{u}^{T}\mathbf{A}^{-1}\mathbf{u})$ is a scalar because $\mathbf{u}^{T}\mathbf{A}^{-1}\mathbf{u}$ is 1-by-1.

To apply (3.18) for inversion of $\mathbf{Q}_{N+1}$ on the left hand side of (3.13a) and (3.13b), with $\mathbf{A}=((N-1)/N)\mathbf{Q}_N$, a symmetric matrix, $\lambda=1/(N+1)$, and $\mathbf{u}=(\mathbf{X}_{N+1} - \mathbf{C}_N)$, we first compute:

$$\begin{aligned}\gamma &= \frac{1}{(N+1) + \frac{N}{N-1}(\mathbf{X}_{N+1}-\mathbf{C}_N)^{T}\mathbf{Q}_N^{-1}(\mathbf{X}_{N+1}-\mathbf{C}_N)} \\ &= \frac{N-1}{(N^2-1) + N\left(\|\mathbf{X}_{N+1}-\mathbf{C}_N\|_{\mathbf{Q}_1^{-1}}\right)^2}.\end{aligned} \tag{3.19}$$

Define the dx1 vector $\mathbf{q}_N$ by:

$$\mathbf{q}_N = \mathbf{A}^{-1}\mathbf{u} = \frac{N}{N-1}\mathbf{Q}_N^{-1}(\mathbf{X}_{N+1} - \mathbf{C}_N) \tag{3.20}$$

Then we iteratively compute the inverse of the covariance matrix by:

$$\boxed{\mathbf{Q}_{N+1}^{-1} = \frac{N}{N-1}\mathbf{Q}_N^{-1} - \gamma\mathbf{q}_N\mathbf{q}_N^{T}} \tag{3.21}$$

Given matrix $\mathbf{Q}$, one can compute $\mathbf{Q}^{-1}$ without actually inverting $\mathbf{Q}$ by using (2.14) and avoid dividing by-zero by setting $1/\lambda = \kappa$ if $\lambda \leq \varepsilon$, where $\varepsilon \ll 1$ and $\kappa \gg 1$ are pre-defined numbers, as suggested in [12].

The following lemma addresses the size of clusters resulting from M-ART:

<u>Lemma 3.1:</u> Size of an Ellipsoidal Shaped Cluster

The size R_j (Def. 3.1) *of any ellipsoidal cluster resulting from M-ART is always less than or equal to the value of the vigilance,* ρ*:*

$$R_j \leq \rho, \quad j=1..M \tag{3.22}$$

<u>Proof:</u>

Consider an arbitrary cluster $(\mathbf{C}_j,\mathbf{Q}_j,R_j,N_j)$, $1\leq j\leq M$, resulting from M-ART, where M is the current number of clusters. From Def.3.2, if $N_j=1$ then $R_j= 0$. From (3.9), if an input vector is

incorporated into this cluster, the cluster size is greater than or equal its old value: $R_j^{new} \geq R_j^{old}$. Therefore, from (3.5), this cluster of the M-ART have size less than or equal to ρ:

$$R_j^{old} \leq \rho \ , \forall j = 1..M \tag{3.23}$$

Now consider the two remaining cases:

Case 1: Input $\mathbf{X}_k$ forms a new cluster:

According to Def.3.2, the $(M+1)^{th}$ cluster of the M-ART is ($\mathbf{X}_k$,**I**,0,1) having radius $R_{M+1} = 0 < \rho$.

Case 2: Input $\mathbf{X}_k$ becomes a new member of the "nearest" existing cluster J, $1 \leq J \leq M$

For this case, we note (3.4) and (3.5) must be satisfied, yielding:

$$\left\|\mathbf{X}_k - \mathbf{C}_J^{old}\right\|^2_{\left(\mathbf{Q}_J^{old}\right)^{-1}} = (\mathbf{X}_k - \mathbf{C}_J^{old})^T \left(\mathbf{Q}_J^{old}\right)^{-1} (\mathbf{X}_k - \mathbf{C}_J^{old}) \leq \rho^2 . \tag{3.24}$$

The square of new Mahalanobis distance from $\mathbf{X}_k$ to the new cluster center $\mathbf{C}_J^{new}$, employing the new cluster covariance matrix $\mathbf{Q}_J^{new}$, is:

$$\left\|\mathbf{X}_k - \mathbf{C}_J^{new}\right\|^2_{(\mathbf{Q}_J^{new})^{-1}} = (\mathbf{X}_k - \mathbf{C}_J^{new})^T \left(\mathbf{Q}_J^{new}\right)^{-1} (\mathbf{X}_k - \mathbf{C}_J^{new}) . \tag{3.25}$$

Substitute expressions for $\mathbf{C}_J^{new}$ and $\left(\mathbf{Q}_J^{new}\right)^{-1}$ in (3.12) and (3.21) into the right side of (3.25) and following the form of the left hand side of (3.24), we obtain:

$$\left\|\mathbf{X}_k - \mathbf{C}_J^{new}\right\|^2_{(\mathbf{Q}_J^{new})^{-1}} = (\mathbf{X}_k - \frac{\mathbf{X}_k + N_J \mathbf{C}_J^{old}}{N_J + 1})^T \left[\frac{N_J}{N_J - 1}\left(\mathbf{Q}_J^{old}\right)^{-1} - \gamma \mathbf{q}_N \mathbf{q}_N^T\right] (\mathbf{X}_k - \frac{\mathbf{X}_k + N_J \mathbf{C}_J^{old}}{N_J + 1}) \tag{3.25a}$$

$$= \left(\frac{N_J}{N_J + 1}\right)^2 \frac{N_J}{N_J - 1} (\mathbf{X}_k - \mathbf{C}_J^{old})^T \left(\mathbf{Q}_J^{old}\right)^{-1} (\mathbf{X}_k - \mathbf{C}_J^{old}) - \gamma \left(\frac{N_J}{N_J + 1}\right)^2 (\mathbf{X}_k - \mathbf{C}_J^{old})^T \mathbf{q}_N \mathbf{q}_N^T (\mathbf{X}_k - \mathbf{C}_J^{old}) \tag{3.25b}$$

$$= \frac{N_J^3}{N_J^3 + N_J^2 - N_J - 1} \left\|\mathbf{X}_k - \mathbf{C}_J^{old}\right\|^2_{\left(\mathbf{Q}_J^{old}\right)^{-1}} - \gamma \left(\frac{N_J}{N_J + 1}\right)^2 (\mathbf{X}_k - \mathbf{C}_J^{old})^T \mathbf{q}_N \mathbf{q}_N^T (\mathbf{X}_k - \mathbf{C}_J^{old}) , \tag{3.25c}$$

where $\gamma > 0$ is defined in (3.19) and $\mathbf{q}_N$ is defined in (3.20).

We note that the second term in (3.25c) contains a quadratic term, namely:

$$(\mathbf{X}_k - \mathbf{C}_J^{old})^T \mathbf{q}_N \mathbf{q}_N^T (\mathbf{X}_k - \mathbf{C}_J^{old}) = [(\mathbf{X}_k - \mathbf{C}_J^{old})^T \mathbf{q}_N][(\mathbf{X}_k - \mathbf{C}_J^{old})^T \mathbf{q}_N]^T \geq 0. \quad (3.26)$$

Moreover, in the first term of (3.25c), the quantity $N_J^3/(N_J^3+N_J^2-N_J-1)$ is less than 1, because $N_J^2-N_J-1>0, \forall N_J \geq 2, N_J \in \mathbb{N}$. Accordingly, we can conclude that:

$$\|\mathbf{X}_k - \mathbf{C}_J^{new}\|^2_{(\mathbf{Q}_J^{new})^{-1}} < \left\|\mathbf{X}_k - \mathbf{C}_J^{old}\right\|^2_{(\mathbf{Q}_J^{old})^{-1}} \leq \rho^2 . \quad (3.27)$$

From (3.9), (3.23), and (3.27), the new value of this cluster size is:

$$R_J^{new} = \max\left(R_J^{old}, \quad \|\mathbf{X}_k - \mathbf{C}_J^{new}\|_{(\mathbf{Q}_J^{new})^{-1}} \right) \leq \rho \quad (3.28)$$

Hence, Lemma 3.1 is proven.

The following two lemmas address the existence of the M-ART network that produces the correct number of clusters in the event that the data set contains only one and two clusters.

Lemma 3.2: Existence of the M-ART for 1 cluster

Assume a data set **S** *contains only one cluster* (**C**,**Q**,R,N). *Then*

a. *M-ART clustering with*

$$\rho \geq 2\,R, \quad (3.29)$$

will always result in 1 cluster.

b. *M-ART clustering with*

$$\rho < R \quad (3.30)$$

will result in M > 1 *(unpredictable) clusters, depending on the order of feeding data to M-ART.*

Proof of Lemma 3.2a:

From Def.3.2, M-ART forms at least 1 cluster. We now prove that the M-ART cannot create any new cluster after the first one. Assume that there is one cluster ($\mathbf{C}^{(M)}$, $\mathbf{Q}^{(M)}$,$R^{(M)}$,$N^{(M)}$) consisting of $N^{(M)} < N$ input vectors $\mathbf{X}_m^{(M)}$, m=1..$N^{(M)}$, in M-ART, and a new input vector $\mathbf{X}_k$ will not belong to this cluster. (The superscript (M) indicates the cluster is formed by M-ART, differentiating from the true cluster in the given data set). From Def.3.2, this means that:

$$\left\|\mathbf{X}_k - \mathbf{C}_J^{(M)}\right\|_{\mathbf{Q}^{-1}} = \left\|\mathbf{X}_k - \frac{1}{N^{(M)}}\sum_{m=1}^{N^{(M)}} \mathbf{X}_m^{(M)}\right\|_{\mathbf{Q}^{-1}} > \rho \tag{3.31}$$

For simplicity, we now assume that $\mathbf{Q}^{(M)} = \mathbf{Q}$. This is reasonable, because $\mathbf{Q}^{(M)} \rightarrow \mathbf{Q}$ when $N^{(M)} \rightarrow N$.

Multiplying both sides of (31) with $N^{(M)} > 0$, and applying the *subadditivity* property (or *triangle inequality*) of the norm to the left side yields:

$$\left\|\mathbf{X}_k - \mathbf{X}_1^{(M)}\right\|_{\mathbf{Q}^{-1}} + \left\|\mathbf{X}_k - \mathbf{X}_2^{(M)}\right\|_{\mathbf{Q}^{-1}} + \ldots + \left\|\mathbf{X}_k - \mathbf{X}_{N^{(M)}}^{(M)}\right\|_{\mathbf{Q}^{-1}} \geq \left\|N^{(M)}\mathbf{X}_k - \sum_{m=1}^{N^{(M)}} \mathbf{X}_m^{(M)}\right\|_{\mathbf{Q}^{-1}} > N^{(M)}\rho \tag{3.32}$$

From (3.32) and given $\rho \geq 2R$, there exists $1 \leq q \leq N^{(M)}$ such that

$$\left\|\mathbf{X}_k - \mathbf{X}_q^{(M)}\right\|_{\mathbf{Q}^{-1}} > 2R \tag{3.33}$$

On the other hand, from Def.3.1, the distance between any two vectors in the same cluster cannot be greater than twice its size. However $\mathbf{X}_k$ and $\mathbf{X}_q^{(M)}$ belong to the true cluster ($\mathbf{C}$,Q,R,N). It is evident from (3.1) and the *subadditivity* property of the norm that:

$$\left\|\mathbf{X}_k - \mathbf{X}_q^{(M)}\right\|_{\mathbf{Q}^{-1}} = \left\|(\mathbf{X}_k - \mathbf{C}) + (\mathbf{C} - \mathbf{X}_q^{(M)})\right\|_{\mathbf{Q}^{-1}} \leq \left\|\mathbf{X}_k - \mathbf{C}\right\|_{\mathbf{Q}^{-1}} + \left\|\mathbf{X}_q^{(E)} - \mathbf{C}\right\|_{\mathbf{Q}^{-1}} \leq 2R\,, \tag{3.34}$$

which contradicts (3.33). Thus any new vector inputs to M-ART will become a new member of only one cluster ($\mathbf{C}^{(M)}$, $\mathbf{Q}^{(M)}$,,$R^{(M)}$,$N^{(M)}$) which proves Lemma 3.2a.

Proof of Lemma 3.2b: Assume there is only one cluster ($\mathbf{C}^{(M)}$, $\mathbf{Q}^{(M)}$,$R^{(M)}$,$N^{(M)}$) resulting from M-ART, or:

$$\mathbf{X}_k \in (\mathbf{C}^{(M)}, \mathbf{Q}^{(M)}, R^{(M)}, N^{(M)}) \text{ for } \forall\, \mathbf{X}_k \in (\mathbf{C}, Q, R, N) \tag{3.35}$$

According to (3.7) and (3.8), $\mathbf{C}^{(M)}$ and $\mathbf{Q}^{(M)}$created by M-ART are the mean and the covariance matrix of all vectors belong to this cluster, respectively, or:

$$\mathbf{C}^{(M)} = \mathbf{C} \text{ and } \mathbf{Q}^{(M)} = \mathbf{Q} \tag{3.36}$$

By Def.3.1, there exists $\mathbf{X}_m$ such that:

$$\left\|\mathbf{X}_m - \mathbf{C}\right\|_{\mathbf{Q}^{-1}} = \left\|\mathbf{X}_m - \mathbf{C}^{(M)}\right\|_{\mathbf{Q}^{-1}} = R \tag{3.37}$$

Following (3.30), we have

$$\|\mathbf{X}_m - \mathbf{C}^{(M)}\|_{\mathbf{Q}^{-1}} = R > \rho \tag{3.38}$$

Hence, $\mathbf{X}_m$ does not satisfy (3.4) and (3.5), or does not belong to the cluster ($\mathbf{C}^{(M)}$, $\mathbf{Q}^{(M)}$,$R^{(M)}$,$N^{(M)}$), which contradicts our assumption. Hence, M-ART will produce more than one cluster, which proves Lemma 3.2b.

Lemma 3.3: Existence of M-ART for 2 clusters

Assume a data set **S** *contains only two clusters* ($\mathbf{C}_1$, $\mathbf{Q}_1$,R_1,N_1) *and* ($\mathbf{C}_2$,$\mathbf{Q}_2$,R_2,N_2) with $R_1 \geq R_2$.

Define:

$$\Delta_{min} = \min_{1\leq j\leq N_1; 1\leq k\leq N_2} \left(\left\|\mathbf{X}_j^{(1)} - \mathbf{X}_k^{(2)}\right\|_{\mathbf{Q}_1^{-1}}; \left\|\mathbf{X}_j^{(1)} - \mathbf{X}_k^{(2)}\right\|_{\mathbf{Q}_2^{-1}} \right) \tag{3.39}$$

where $\mathbf{X}_j^{(1)} \in (\mathbf{C}_1, \mathbf{Q}_1, R_1, N_1)$ *and* $\mathbf{X}_k^{(2)} \in (\mathbf{C}_2, \mathbf{Q}_2, R_2, N_2)$.

Then M-ART clustering with

$$2R_1 \leq \rho < \Delta_{min} \tag{3.40}$$

will always result in 2 clusters.

Proof:

Let $\mathbf{X}_i^{(1)}$, $\mathbf{X}_j^{(1)} \in (\mathbf{C}_1, \mathbf{Q}_1, R_1, N_1)$,$\forall i,j = 1..N_1$, and $\mathbf{X}_k^{(2)}$, $\mathbf{X}_n^{(2)} \in (\mathbf{C}_2, \mathbf{Q}_2, R_2, N_2)$,$\forall k,n = 1..N_2$

From Def.3.1, the "intra-class" distance, Δ_{ij} or Δ_{kn}, of any two points belonging to the same cluster satisfies:

$$\begin{aligned} \Delta_{ij} &= \left\|\mathbf{X}_i^{(1)} - \mathbf{X}_j^{(1)}\right\|_{\mathbf{Q}_1^{-1}} \leq 2R_1 \quad , \forall i, j=1..N_1 \\ \Delta_{kn} &= \left\|\mathbf{X}_k^{(2)} - \mathbf{X}_n^{(2)}\right\|_{\mathbf{Q}_2^{-1}} \leq 2R_2 \leq 2R_1 \quad , \forall k, n=1..N_2 \end{aligned} \tag{3.41}$$

Assume that the first input vector feed to the M-ART network is $\mathbf{X}_j^{(1)} \in (\mathbf{C}_1, \mathbf{Q}_1, R_1, N_1)$. According to Def.3.2, M-ART forms the first cluster, ($\mathbf{C}_1^{(E)} = \mathbf{X}_j^{(1)}$, $\mathbf{Q}_1^{(E)} = \mathbf{I}$, $R_1^{(E)}=0$, $N_1^{(E)}=1$), to contain this input vector.

Assume the second input vector fed to the M-ART is $\mathbf{X}_k^{(2)} \in (\mathbf{C}_2,\mathbf{Q}_1,R_2,N_2)$. From (3.39) and (3.40), we have: $\left\|\mathbf{X}_k^{(2)} - \mathbf{C}_1^{(E)}\right\|_{\mathbf{Q}_1^{-1}} \geq \Delta_{min} > \rho$. The condition (3.5) of Def.3.2 is NOT satisfied. Therefore, M-ART creates a new cluster, ($\mathbf{C}_2^{(E)} = \mathbf{X}_k^{(2)}$, $\mathbf{Q}_2^{(E)} = \mathbf{I}$, $R_2^{(E)}=0$, $N_2^{(E)}=1$), to contain this input.

Consider any third input fed to the M-ART network, for which there are two cases:

Case 1: The third input is $\mathbf{X}_i^{(1)}$, which actually belongs to the first cluster: $\mathbf{X}_i^{(1)} \in (\mathbf{C}_1, \mathbf{Q}_1,R_1,N_1)$

From (3.39), (3.40), and (3.41), we obtain:

$$\begin{aligned} &\left\|\mathbf{X}_i^{(1)} - \mathbf{C}_1^{(E)}\right\|_{\mathbf{Q}_1^{-1}} \leq 2R_1 \leq \rho \\ &\left\|\mathbf{X}_i^{(1)} - \mathbf{C}_2^{(E)}\right\|_{\mathbf{Q}_2^{-1}} \geq \Delta_{min} > \rho \end{aligned} \tag{3.42}$$

In other words, this input $\mathbf{X}_i^{(1)}$ will be classified by M-ART into the first cluster, $(\mathbf{C}_1^{(E)},\mathbf{Q}_1^{(E)},R_1^{(E)},N_1^{(E)})$.

Case 2: The third input is $\mathbf{X}_n^{(2)}$, which actually belongs to the second cluster: $\mathbf{X}_n^{(2)} \in (\mathbf{C}_2,\mathbf{Q}_2,R_2,N_2)$

As before, from (3.39), (3.40), and (3.41), we obtain:

$$\begin{aligned} &\left\|\mathbf{X}_n^{(2)} - \mathbf{C}_1^{(E)}\right\|_{\mathbf{Q}_1^{-1}} \geq \Delta_{min} > \rho \\ &\left\|\mathbf{X}_n^{(2)} - \mathbf{C}_2^{(E)}\right\|_{\mathbf{Q}_2^{-1}} \leq 2R_1 \leq \rho \end{aligned} \tag{3.43}$$

So this input $\mathbf{X}_n^{(2)}$ will be classified by M-ART into the second cluster, $(\mathbf{C}_2^{(E)},\mathbf{Q}_2^{(E)},R_2^{(E)},N_2^{(E)})$.

Hence, M-ART does not create a new cluster, which proves Lemma 3.3.

Special case $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{I}$:

Lemma 3.3 states that if two clusters are “separated enough”, there exist certain values of vigilance ρ so that M-ART produces the correct two clusters. We can easily visualize conditions

(3.39) and (3.40) in the special case when $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{I}$, and in 2 dimensions, the clusters are circular shaped. Assume that for two clusters, with centers $\mathbf{C}_1$ and $\mathbf{C}_2$, we have the center-to-center distance D_{12} given by:

$$D_{12} = \|\mathbf{C}_1 - \mathbf{C}_2\|_{\mathbf{Q}^{-1}} = \|\mathbf{C}_1 - \mathbf{C}_2\| \geq 3(R_1 + R_2) \tag{3.44}$$

Then, (3.40) becomes:

$$\Delta_{min} = \min_{1 \leq j \leq N_1} \left(\left\| \mathbf{X}_j^{(1)} - \mathbf{C}_2 \right\|_{\mathbf{Q}^{-1}} \right) = \min_{1 \leq k \leq N_2} \left(\left\| \mathbf{X}_k^{(2)} - \mathbf{C}_1 \right\|_{\mathbf{Q}^{-1}} \right) \geq 2(R_1 + R_2) \tag{3.45}$$

Fig.3.1 illustrates the case of two circular shaped clusters, $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{I}$, that are "separated enough" so that M-ART with any ρ satisfying (3.40) will produce the correct 2 clusters.

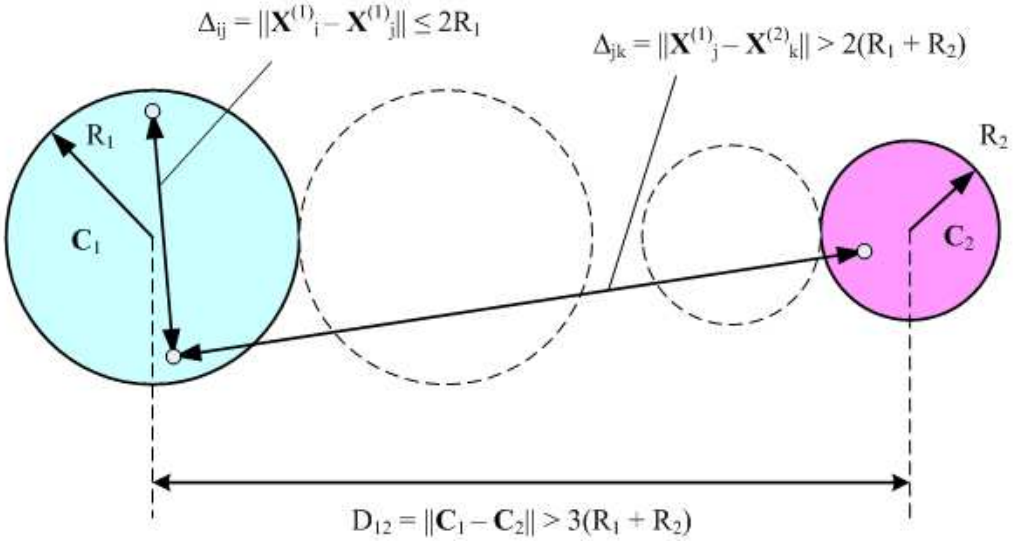


Figure 3.1: Two clusters, with $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{I}$, with circular shapes, are "separated enough" so that M-ART with any ρ satisfying (3.40) will produce the correct 2 clusters.

3.2 Selecting a Good Value for ρ:

There exist values of ρ in a certain range such that M-ART produces the "correct number" (as would be determined by humans) of clusters in a particular data set, but the appropriate value of ρ depends heavily on the distribution of patterns in the data set, which is normally unknown a priori. The principal remaining concern is determination of a good value for ρ. Small values of ρ

produce small clusters, such that the number of clusters is large, and large values produce larger clusters, such that the number of clusters is small. Conventionally, a trial-and-error approach is used to tune the value of ρ for each input data set. We propose an automatic procedure that could be applied for any input data set in what follows. We first introduce another definition:

Definition 3.3: Separation Factor

The separation factor SV_{jk} *of two clusters* $(\mathbf{C}_j,\mathbf{Q}_j,R_j,N_j)$ *and* $(\mathbf{C}_k,\mathbf{Q}_k,R_k,N_k)$ *is a positive real number defined by:*

$$SV_{jk} = \frac{\|\mathbf{C}_j - \mathbf{C}_k\|_{\mathbf{Q}_j^{-1}} + \|\mathbf{C}_k - \mathbf{C}_j\|_{\mathbf{Q}_k^{-1}}}{2(R_j + R_k)} \tag{3.46}$$

These two clusters are said to be "separated enough" if $SV_{ik} \geq SV$ *for some pre-defined number* SV*, or to be "too close" otherwise.*

Note that for $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{I}$*,(spheroidal shaped clusters)* (3.46) *becomes:*

$$SV_{jk} = \frac{\|\mathbf{C}_j - \mathbf{C}_k\|}{R_j + R_k} \tag{3.47}$$

Recall that the Mahalanobis distance (M-distance) is used as a similarity measurement between a point **X** and a center of the cluster **C**, which indicates how likely this new point should belong to this cluster. M-distance differs from Euclidean distance (E-distance), which is isotropic and does not depend on the distribution of the cluster data points. M-distance puts high weights along axes with high variance (major axes) of the cluster data points, while the direction with low variance of cluster data points (minor axes) is weighted lower. If the covariance matrix of the cluster data points is the identity matrix, the M-distance becomes the E-distance. If the covariance matrix is diagonal, then the resulting distance measure is called the normalized E-distance. The separation factor in (3.46) represents how much separation or "overlap" exists between any two clusters. As SV_{jk} increases from 0, the j^{th} and k^{th} clusters move from maximum overlap to maximum separation.

Some examples of two circular shaped clusters that have separation factors equal to 0.5, 1, and 1.1 are shown in Fig.3.2. Figure 3.3 illustrates other examples of separation factors between two clusters, circular or ellipsoidal shaped. In Fig.3.3, each number beside a cluster (represented by a blue circle or blue ellipse) indicates the separation factor between this cluster and the cluster represented by the red ellipse with light-cyan filled color.

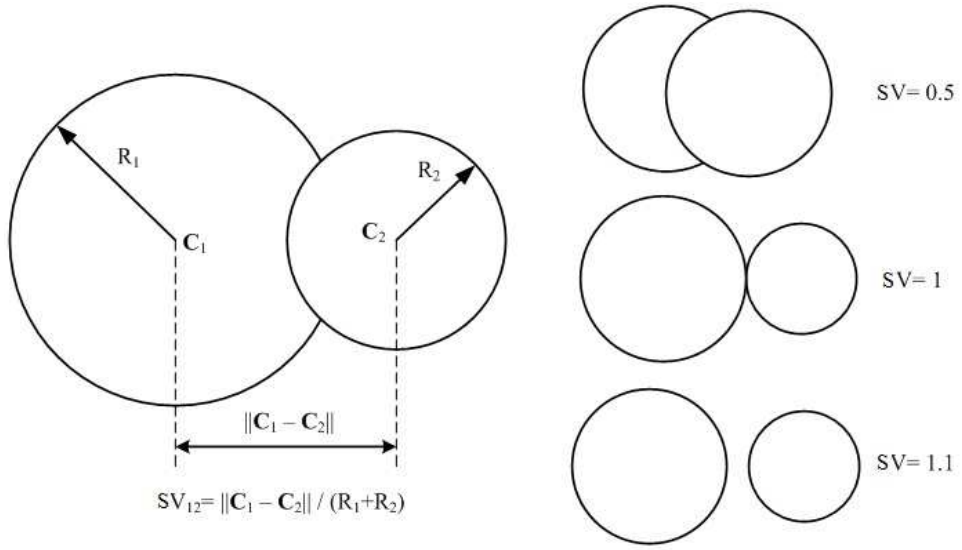


Figure 3.2: (a) Separation-factor between Two Circular Shaped Clusters.
(b): Some examples with: (upper) SV=0.5, (middle) SV=1.0, (bottom) SV=1.1

The idea for auto-adjustment of ρ is to use M-ART with small ρ, producing clusters that are excessively close together, or heavily overlapped, and then graduately increasing ρ until all clusters are "sufficiently" separated or less heavily overlapped. This process is equivalent to merging clusters that are deemed excessively close together into larger ones.

Figure 3.4 (a) shows a sample data set (data a2 in Chapter 4) with 35 natural clusters (as determined by humans). A very small value of ρ=0.06 produces from M-ART 86 clusters that are excessively close, Fig.3.4 (b). With a larger value of ρ=0.11, M-ART produces 35 clusters that are sufficiently separated, Fig.3.4 (c), and in line with what most humans would produce.

Discussion on selecting a value for SV is given in Chapter 4, Eqn (4.6), and Table 4.3, and Fig.4.5.

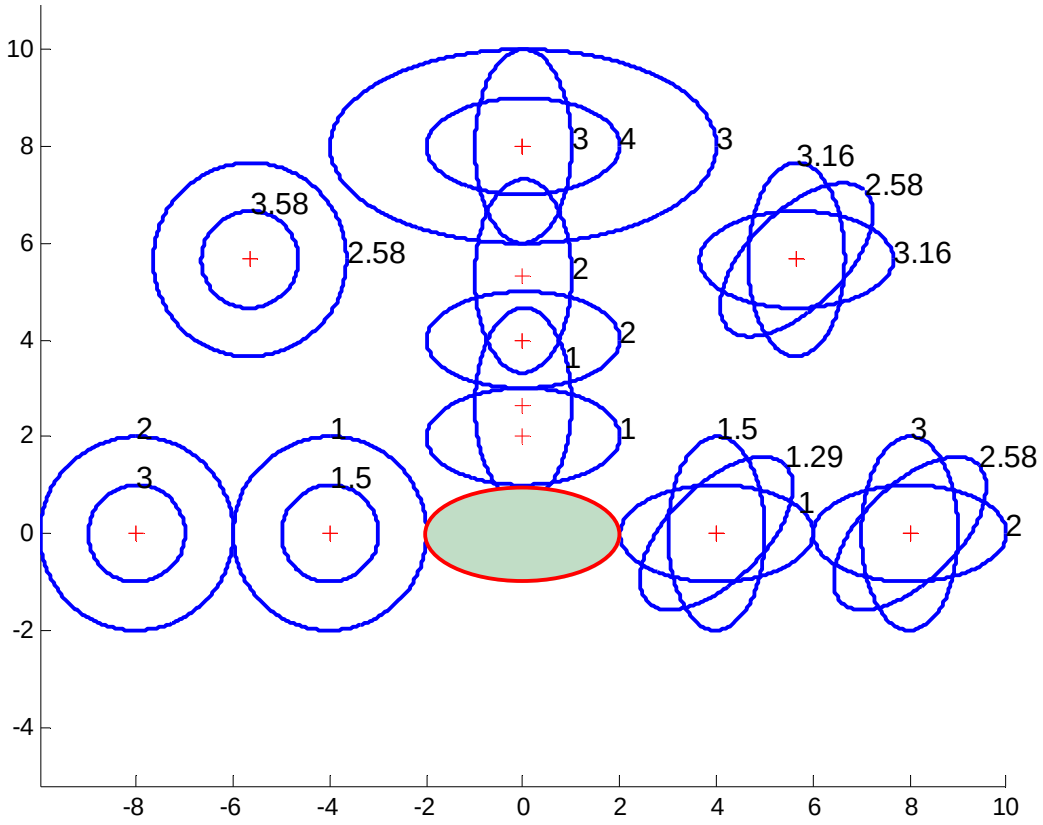
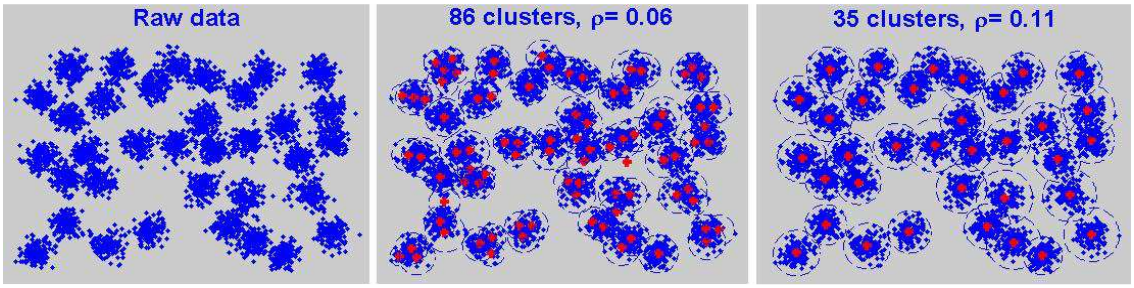


Figure 3.3: Example of separation factors between circular shaped and ellipsoidal shaped clusters. Each number beside a cluster (represented by a blue circle or blue ellipse) indicates the separation factor between this cluster and the cluster represented by the red ellipse with light-cyan filled color.



(a) (b) (c)

Figure 3.4: (a) Data Set a1. (b) and (c) Clustering Results with Different Vigilance Values.

The pseudo-code for M-ART with auto adjustment of vigilance ρ is shown in Table 3.1. The auto adjustment process starts with an arbitrarily small value of vigilance ρ. The first *Repeat-Until* loop in Table 3.1 reduces the value of ρ to guarantee that there are at least two clusters resulting from M-ART, which will produce clusters that are excessively close (Def.3.3). The second *Repeat-Until* loop in Table 3.1 gradually increases the value of ρ to guarantee that all clusters

resulting from the M-ART are sufficiently separated according to Def.3.3 and a user-selected value of SV.

Table 3.1: Pseudo-code of the M-ART with Auto-adjust Vigilance Algorithm

Choose a desired separation factor SV

Initialize an arbitrarily small value vigilance ρ

$K_{down} = 0.1$

$K_{up} = 1.1$

Repeat

$\rho_{new} = K_{down}.\rho_{old}$

Run M-ART with ρ_{new} for entire data set

$$D_{jk} = 0.5(\|\mathbf{C}_j - \mathbf{C}_k\|_{\mathbf{Q}_j^{-1}} + \|\mathbf{C}_k - \mathbf{C}_j\|_{\mathbf{Q}_k^{-1}}) \quad , 1 \leq j \neq k \leq M$$

Until $D_{jk} \leq SV(R_j+R_k)$, for some j and k, $1 \leq j \neq k \leq M$

Repeat

$\rho_{new} = K_{up}.\rho_{old}$

Run M-ART with ρ_{new} for entire data set

$$D_{jk} = 0.5(\|\mathbf{C}_j - \mathbf{C}_k\|_{\mathbf{Q}_j^{-1}} + \|\mathbf{C}_k - \mathbf{C}_j\|_{\mathbf{Q}_k^{-1}}) \quad , 1 \leq j \neq k \leq M$$

Until $\min_{j,k}\left(\frac{D_{jk}}{Rj+Rk}\right) > SV$

3.3 Density-based removal pre-processing

M-ART is essentially distance-based clustering, which means M-ART considers dense or sparse input vectors equally. That is why M-ART often produces spurious clusters consisting of vectors that lie in the outlying regions of these clusters. By itself, M-ART does not mimic well our eye-brain system for identifying clusters. From observation, it is believed that the first step in human selection of 2-D clusters involves detecting high-density areas of points, which they label as

cluster centers. The second step in human cluster detection would then be seeking "gaps" between clusters to define the outliers. In this paper, we propose pre-processing of data by removing points from an input data set that lie in areas of low point density before feeding to the M-ART network. The idea is to remove some outlying vectors such that the cluster structure (number of clusters and cluster centers) of the remaining data will not likely change significantly (if at all) from those of the original data set. Removing some points in areas on the margins of clusters may render remaining points more easily identifiable as part of a distinct cluster and easier to identify by M-ART.

Table 3.2: Pseudo-code of Density-based Removal Pre-processing

Input data set $\mathbf{S} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L]$ consisting of total L input vectors

Select a keeping ratio, $0 < k_r \leq 1$, and ranking ratio $0 < r_r \leq 1$

$n_1 = \text{round}(Lk_r)$

$n_2 = \text{round}(Lr_r)$

For each input vector $\mathbf{X}_i$

Compute pair-wise distance to all other vectors $\Delta_{ij} = \sqrt{(\mathbf{X}_i - \mathbf{X}_j)^T(\mathbf{X}_i - \mathbf{X}_j)}$, $j = 1..L$

Rank all L values of Δ_{ij}, j=1..L, from smallest to largest

Compute the density at the point $\mathbf{X}_i$: $r_i = 1/\left(\frac{1}{n_1}\sum_{j=1}^{n_1}\Delta_{ij}\right)$

End

Find r_{cut} as the n_2^{th} smallest value of r_i array (i=1..L).

Remove any pattern that is in a low density region

$\mathbf{S}_{remaining} = \mathbf{S} - \{\mathbf{X}_i \text{ such that } r_i \leq r_{cut}\} = \{\mathbf{X}_i \text{ such that } r_i > r_{cut}\}$

The density-based removal pre-processing algorithm is given in Table 3.2. We define the *point density* near a point in the feature space as the total number of input vectors that are inside the unit hyper-sphere surrounding this point. Therefore, the density at the point $\mathbf{X}_i$, called r_i, could be

approximated by one divided by the average of distances between $\mathbf{X}_i$ and a specific number of its nearest input vectors. Small average distances produce larger values for r_i, which means that neighbors of $\mathbf{X}_i$ are close. In other words, such a point should be near the center of a true cluster. In contrast, larger average distances produce small value for r_i, which means that $\mathbf{X}_i$ should be near the boundaries of a true cluster. Thus, we remove some input vectors $\mathbf{X}_i$ with small values of r_i (low density).

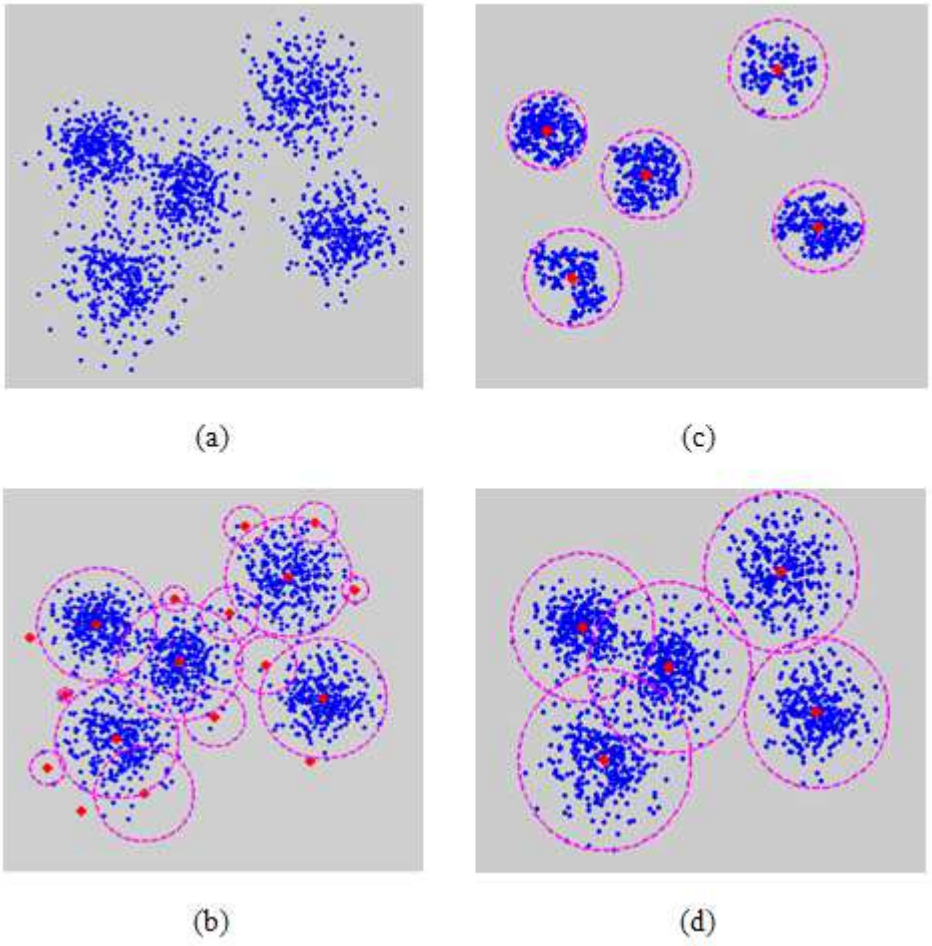


Figure 3.5: (a): Original Data Set t3;
(b) Result of Directly Applying the M-ART Clustering with Many Spurious Clusters
(c) $S_{remaining}$ with Five Clusters Easily Identified by the M-ART Clustering
(d) Final Partition of the Original Set with These Five Centers Retained from (c)

Figure 3.5 (a) shows the data set t3 (Chapter 4). M-ART clustering produces a total of 18 clusters using SV=0.8, ρ=0.21 shown in Fig.3.5 (b). Most humans would quickly conclude that there are only 5 clusters. After pre-processing by density-based removal, described in Table 3.2, the

remaining data is fed to M-ART clustering, which easily identifies five separated clusters shown in Fig.3.5 (c), with the same values of SV and ρ. These five cluster centers are retained to decide which is the closest center to which each input vector of the original data set should belong Fig.3.5 (d). Results showed in Fig.3.5 were obtained from the best choice of SV=0.8 after experimented with difference values of SV as described more detail later in Chapter 4.

3.4 Automatic clustering procedure

Using density-based removal pre-processing could benefit the M-ART process in locating clusters. However, it is evident that insufficient removal would not provide sufficient help to M-ART, while excessive removal could lead to a spurious cluster structure. We define two parameters that affect removal pre-processing: keeping ratio k_r and ranking ratio r_r. It is not trivial to select suitable combinations of these parameter values. Keeping ratio affects the number of nearest neighbors to approximate the density at a given point, such that it presents local information. Ideally, the number of nearest neighbors should approximate the number of input vectors of a cluster, so we define k_r by:

$$k_r = (\text{total number of input vectors})/ (\text{number of true clusters}) \quad (3.48)$$

which assumes that clusters are somewhat equal in their number of members. However, the number of true clusters is unknown. By experiments with a number of sample data sets (given in Chapter 4), we suggest a rule of thumb:

$$0.01 \leq k_r \leq 0.10 \quad (3.49)$$

For example if the number of total input vector (patterns) is several thousand, the number of counting neighbors of each point should be several hundred or less. If we select $k_r > 0.1$, the cluster structure may become distorted.

Ranking ratio, similar to global thresholding of hierarchical clustering methods, is a parameter used to determine when to ignore low densities of a pattern. It directly affects how many vectors

or patterns will be removed in pre-processing. Again, from experience with data sets given in Chapter 4, we propose the rule of thumb:

$$0.10 \leq r_r \leq 0.40 \tag{3.50}$$

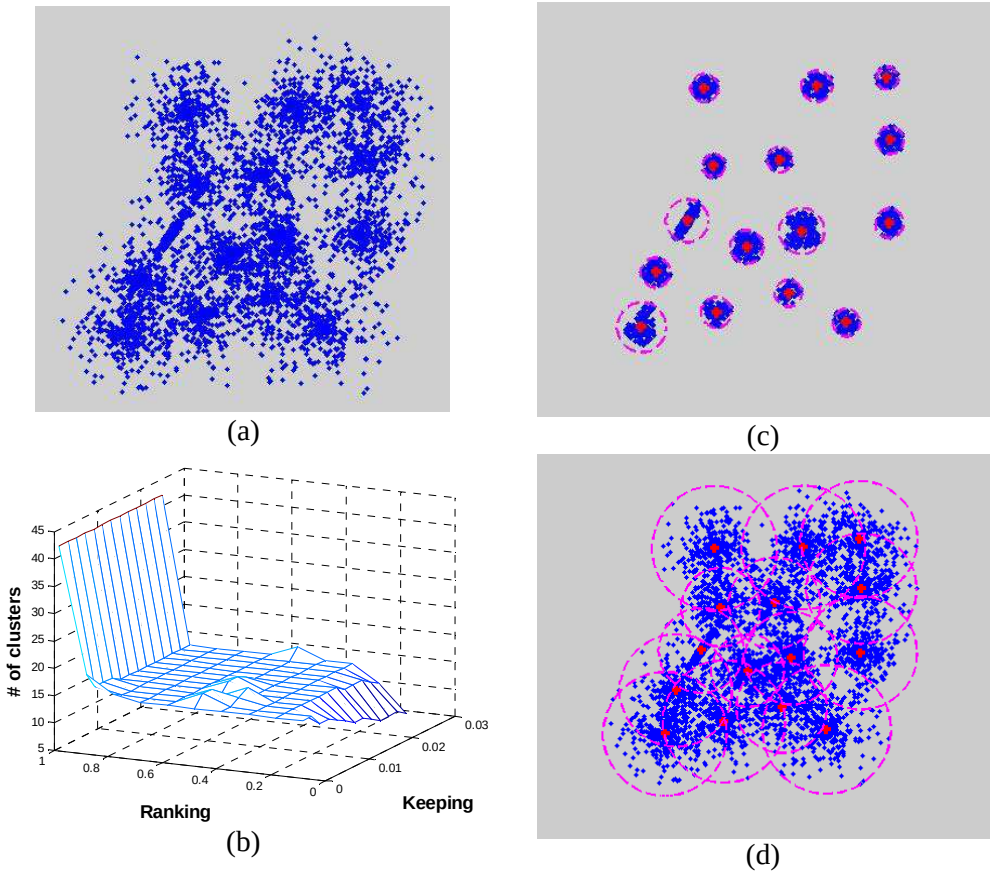
For example, if $r_r = 0.10$, then data points with densities ranking in the lowest 10% of all point densities will be removed.

Table 3.3: Pseudo-code of Automatic Clustering Procedure

Use density-based removal pre-processing, varying k_r and r_r
Use M-ART with auto-adjust vigilance algorithm for each remaining data set
Plot the number of clusters vs. k_r and r_r
Select a pair of k_r and r_r that produce the same number of clusters (fall on a plateau of the plot). When multiple values for k_r and r_r exist on the plateau, select the smallest values.
Save cluster centers corresponding to the same number of clusters on the plateau
Assign each pattern of the original data set to the closest (M-distance) cluster

Table 3.3 illustrates the automatic clustering procedure. The goal of this is to automatically select the proper values of k_r and r_r of density-based removal pre-processing to clean the data set before feeding it to M-ART with the auto-adjustment of vigilance, discussed above. While varying values of k_r and r_r in a certain range, for example using (3.49) and (3.50), apply removal pre-processing to an original data set for each pair of k_r and r_r. Then, use M-ART with auto-adjustment of vigilance for each remaining data set corresponding to each pair of k_r and r_r. Plot the number of clusters vs. k_r and r_r in a 3-D plot. Experiments and observations with data sets in Section 5 suggest that clustering that agrees with human clustering occurs when the number of clusters is relatively insensitive to small changes in k_r and r_r. This will happen in regions of the plot of cluster numbers vs. k_r and r_r where a plateau (or flat) occurs. Therefore, a pair of k_r and r_r lying on this plateau is selected. Cluster centers for M-ART for the remaining data set with selected values of k_r and r_r are retained to cluster the original data set. Then the task of finding

which cluster that a given pattern of the original data set should belong to becomes trivial, by finding the nearest distances from that pattern to cluster centers.



(a) (c) (b) (d)

Figure 3.6: (a) Original Data Set s3; (b) Plot of Number of Clusters vs. k_r and r_r;
(c) Clustering after Removing with k_r=0.01, r_r=0.40;
(d) Final Partition of the Original Set with 15 Clusters

Figure 3.6 illustrates the process of clustering for the 2-D data set s3 (a). Appling the process given in Table 3.3 with $0.01 \leq k_r \leq 0.30$ and $0.01 \leq r_r \leq 1.00$ (which are larger than range of k_r and r_r given in (3.49) and (3.50)), we produce the plot of number of cluster vs. k_r and r_r, Fig.3.6 (b), with SV=0.8, ρ=0.12. Select the smallest value of k_r and r_r in the plateau region of the plot, namely k_r=0.01, r_r=0.40. The result of removal pre-processing with these values of k_r and r_r is given in Fig.3.6 (c), showing 15 clusters, using SV=0.8, ρ=0.12. Using these cluster centers to decide the closest center (M-distance) that each input vector of the original data set should belong is straightforward, with results given in Fig.3.6 (d). Again, these results are consistent with what a human would produce.

Chapter 4

RESULTS OF THE PROPOSED DATA CLUSTERING TECHNIQUE

4.1 Test with artificial data

The 21 data sets used in this study to obtain results are described in Table 4, and the fifteen 2-D data sets are depicted graphically in Figure 4.1. Data sets labeled a1 to a3 and s1 to s4 are taken from http://cs.joensuu.fi/sipu/datasets/. Data sets a1 to a3 are synthetic 2-D data with varying numbers of clusters. Data sets s1 to s4 are synthetic 2-D data with 5000 vectors and 15 Gaussian clusters with different degrees of cluster separation. Data sets s1m1, s2m1, s2m2, s3m1, and s4m1 are modified sets taken from original data sets s1, s2, s3, and s4, respectively, by manually removing some ellipsoidal-shaped clusters. Data sets t1 to t8 are synthetic 2-D, 3-D, 4-D, 5-D, and 10-D data with the number of Gaussian clusters described in Table 4.1, prepared by the authors. By applying the automatic clustering procedure described in Section 4, all data sets whose clusters are reasonably separated, such as 2-D data sets a1, a2, a3, s1, s1m1, s1m2, s2, s2m1, s2m2, s3, s3m1, t2, and t3, yield the number of clusters identical with the number actually generated and what most human observers would detect. For higher dimensional data sets, the results (number of clusters determined by M-ART for each data set) agree with what were generated.

Table 4.1: Data Sets and Characteristics

Data set	Data Dimension	# Vectors	# Clusters	Data set	Data Dimension	# Vectors	# Clusters
a1	2-D	3000	20	**s3m1**	2-D	4634	14
a2	2-D	5250	35	**s4m1**	2-D	3903	13
a3	2-D	7500	50	**t1**	2-D	1500	5
s1	2-D	5000	15	**t2**	2-D	2500	10
s2	2-D	5000	15	**t3**	3-D	1200	8
s3	2-D	5000	15	**t4**	3-D	1500	10
s4	2-D	5000	15	**t5**	3-D	1500	10
s1m1	2-D	4685	14	**t6**	4-D	1500	5
s1m2	2-D	4356	13	**t7**	5-D	1500	5
s2m1	2-D	4356	13	**t8**	10-D	1500	5
s2m2	2-D	4326	12				

Results are tabulated in Table 4.2, together with computational time for automatic clustering using a PC Pentium dual 3.0GHz, 2GB Ram, Window XP and Matlab-R2009a. Parameters used in the program to generate the data in Table 4.2 are:

Separation ratio: $SV = 0.8$ (4.1)

Range of k_r: $k_r = 0.010$ to 0.055, increment by 0.005 (4.2)

Range of r_r: $r_r = 0.1$ to 1.0, increment by 0.1 (4.3)

Initial vigilance: $\rho_{start} = 0.05$ (4.4)

Scale factor for increasing/decreasing vigilance (see Table 1):

$K_{up} = 1.1$; $K_{down} = 0.1$ (4.5)

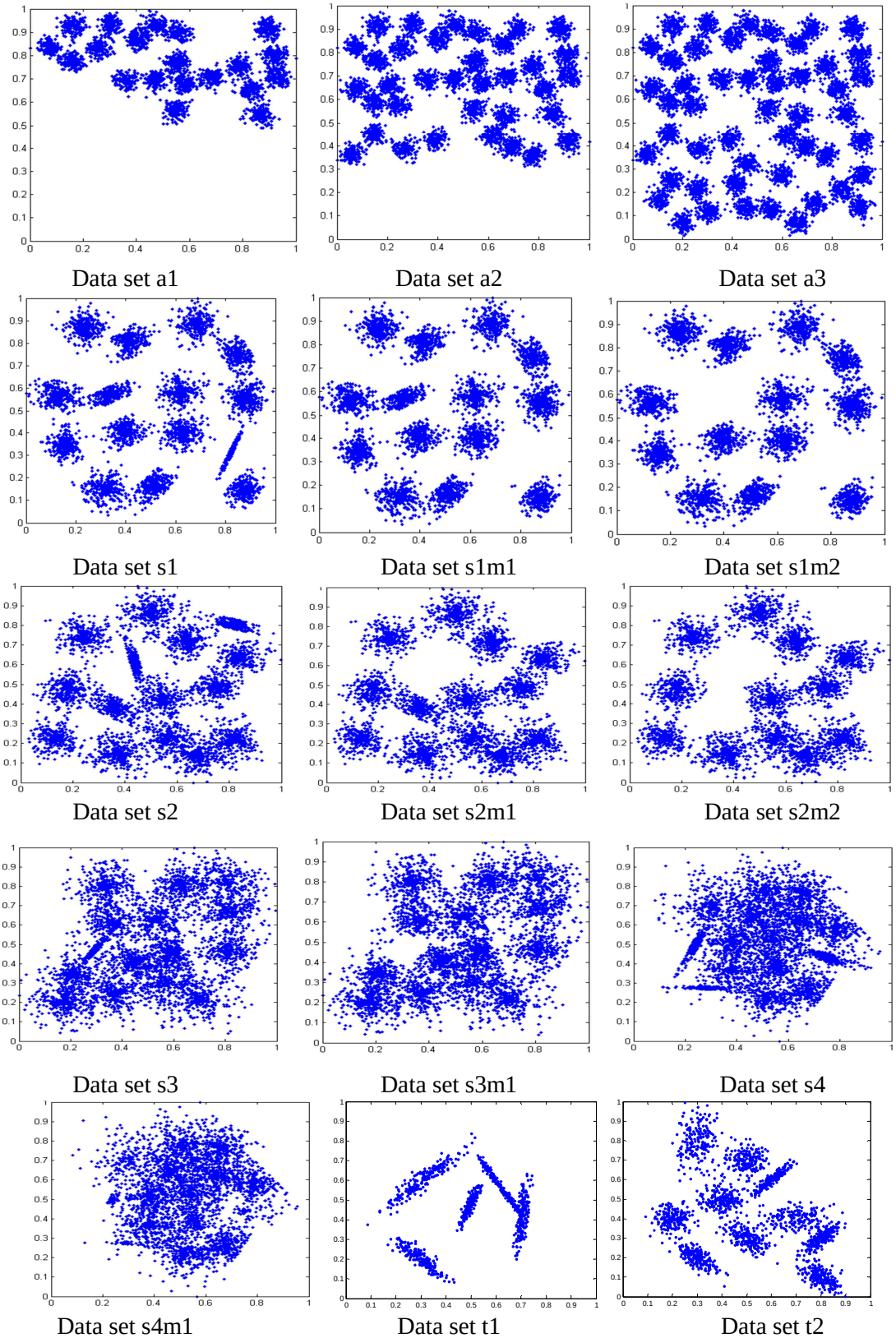


Figure 4.1: Graphical Representation of Data Sets

Table 4.2: Summary Results of Clustering (SV=0.8)

Data sets	Data Dimension	No. clusters used to generate data	No. clusters from automatic clustering	Computational time (sec)	k_r	r_r
a1	2-D	20	20	433.9	0.010	0.300
a2	2-D	35	35	1540.9	0.010	0.300
a3	2-D	50	50	3316.9	0.010	0.300
s1	2-D	15	15	1148.4	0.010	0.200
s1m1	2-D	14	14	1033.6	0.010	0.200
s1m2	2-D	13	13	1608.4	0.010	0.200
s2	2-D	15	15	1414.5	0.010	0.200
s2m1	2-D	13	13	1153.8	0.010	0.100
s2m2	2-D	12	12	1017.3	0.010	0.100
s3	2-D	15	15	1694.2	0.010	0.200
s3m1	2-D	14	14	1564.0	0.010	0.100
s4	2-D	15	**14**	1727.1	0.025	0.400
s4m1	2-D	13	**11**	1239.5	0.010	0.100
t1	2-D	5	5	23.0	0.010	0.100
t2	2-D	10	10	28.4	0.010	0.100
t3	3-D	8	8	21.6	0.010	0.200
t4	3-D	10	10	25.3	0.010	0.200
t5	3-D	10	10	25.1	0.010	0.100
t6	4-D	5	5	26.4	0.010	0.200
t7	5-D	5	5	27.6	0.020	0.200
t8	10-D	5	5	26.9	0.010	0.200

The k_r and r_r were chosen as the smallest values corresponding to the "largest" plateau region of the plot of the number of clusters vs. k_r and r_r.

Note that the clusters in data set s3 and s3m1 are very close (small separation), yet the algorithm still produces "correct" cluster counts. For data set s4 and s4m1, the algorithm produces results different from those used to generate the clusters. However, clusters of those data sets have very small separation, such that different humans often produce different numbers of clusters.

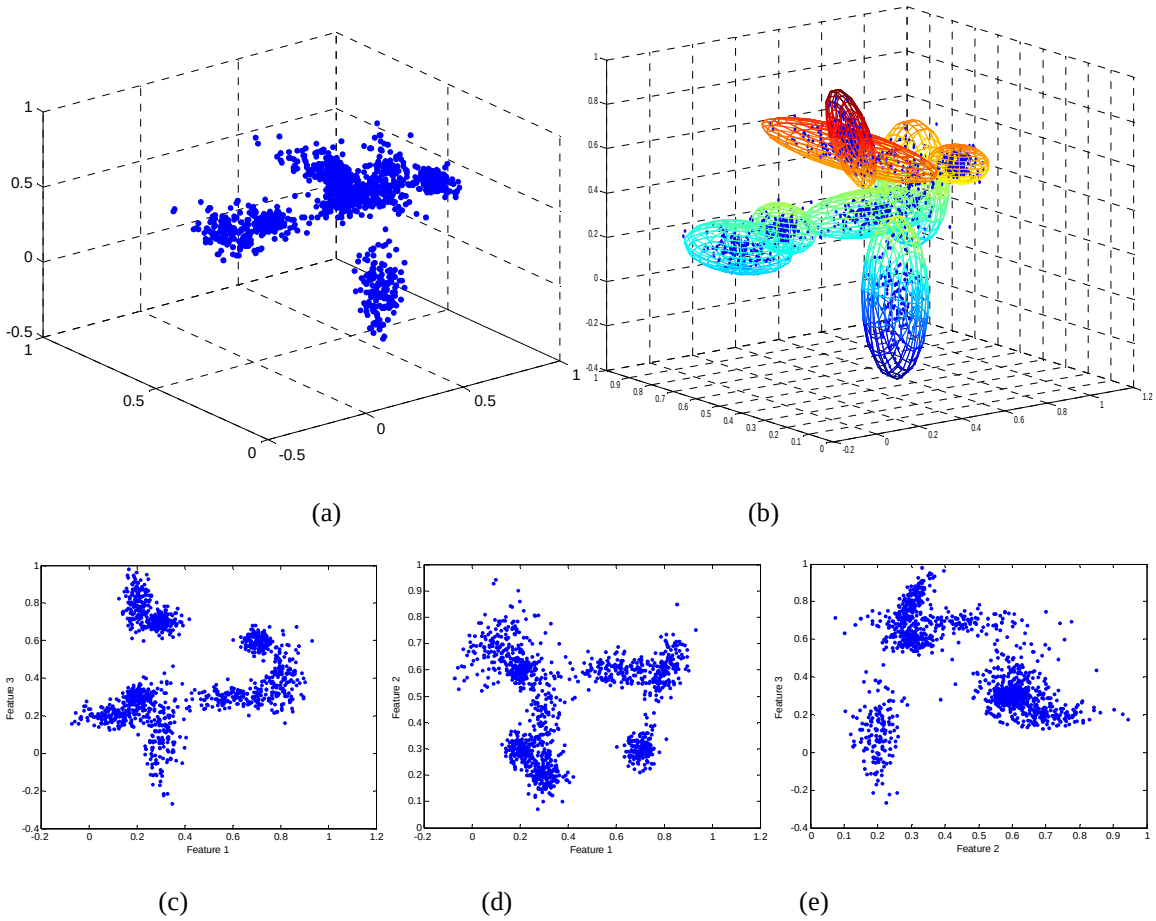
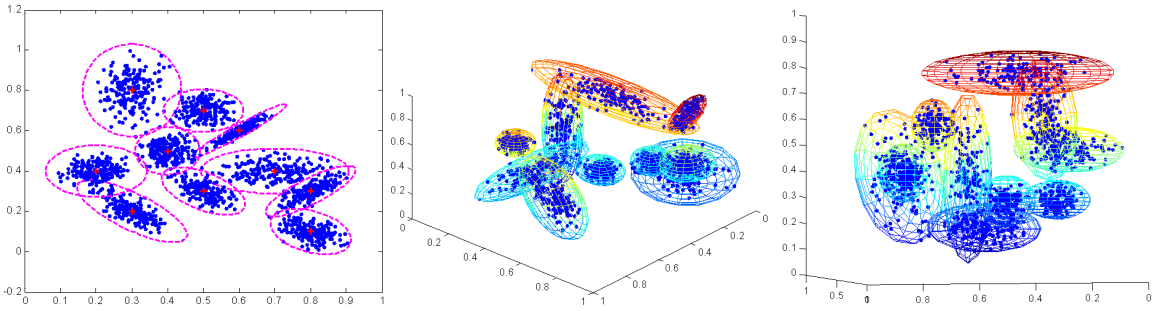


Figure 4.2: Data Set t3 in 3-D (a), and its projection in xy, xz, yz planes (c,d, and e- respectively). Clustering result is shown in (b) with 8 clusters selected by M-ART.

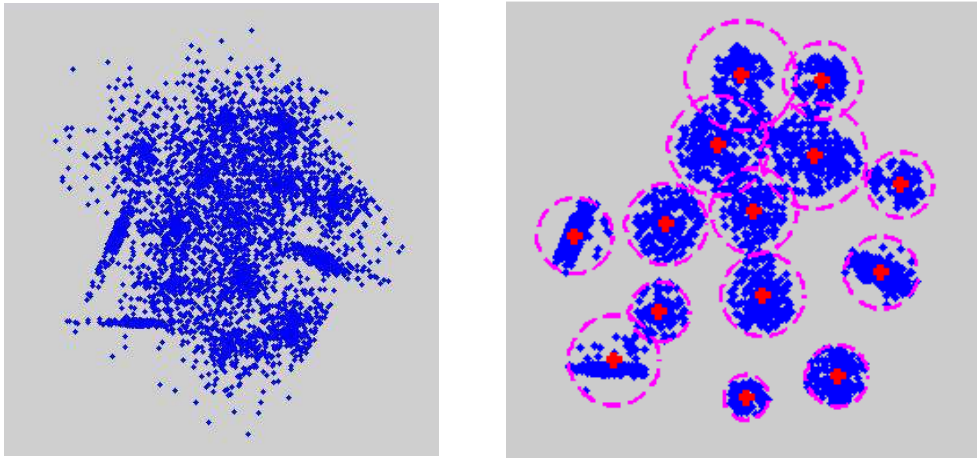
Figure 4.2 presents the original data set t4 in three dimensional space xyz (a), and its projection in xy, xz, yz planes in (c), (d), and (e), respectively. The clustering results is shown in (b) with 8 clusters selected by M-ART, with SV=0.8.



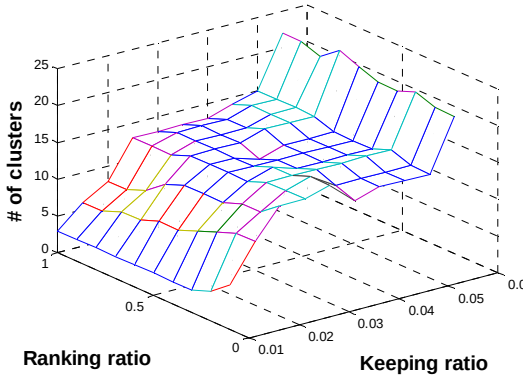
(Data set t3) (Data set t4) (Data set t5)

Figure 4.3: Clustering results of the data set t3, t4, and t5.

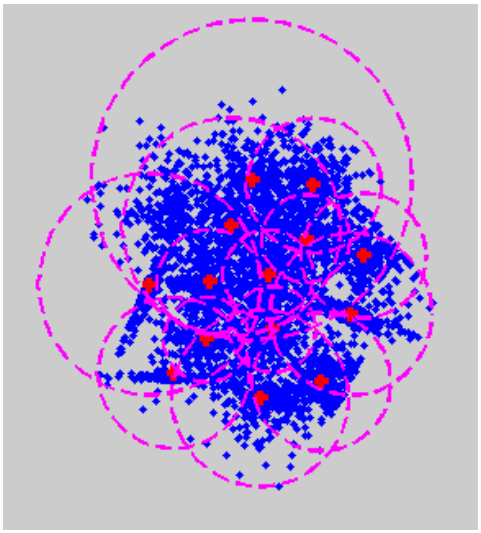
Figure 4.3 illustrates the clustering results of our method for the data set t3 (2-D), the data set t4 (3-D), and the data set t5 (3-D), with SV=0.8.



(a) (c)



(b)



(d)

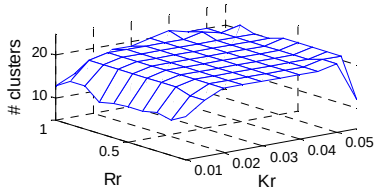
Figure 4.4: (a) Original Data Set s4 Created by 15 Gaussian Clusters; (b) Clustering after Removing with k_r=0.01 and r_r=0.50; (c) Plot of Number of Clusters vs. k_r and r_r; (d) Final Partition with "Incorrect" 14 Clusters.

Figure 4.4 illustrates details on the process of clustering for data set s4 (Fig.4.4 (a)), with SV=0.8. Appling the process that discussed in Table 3.3, we obtain the plot of number of cluster vs. k_r and r_r (Fig.4.4 (b)). Select the pair of smallest values of k_r and r_r in the "largest" plateau region of this plot: k_r=0.025, r_r=0.40. The result of clustering after removal pre-processing with these values of k_r and r_r is shown in Fig.4.4 (c). This process produces 14 clusters, while the "correct" answer, according to what was generated, is 15 clusters. Some points belong to a true cluster (as generated) on the top, left-most of Fig.4.4 (a), showing low point density, and were removed by pre-processing, which reduced the number of detected clusters by 1, to 14. However, if those data points were retained, the clusters after cleaning would not be separated enough for M-ART and the auto-adjust vigilance algorithm to select the correct number of clusters. This is an example in which pre-processing removal does not help M-ART to deal with sparse clusters. In this case, pre-processing removal changes the structure of the data set, or does not produce a cleaned data set with separated clusters.

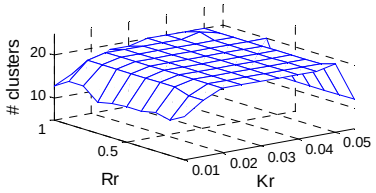
Table 4.3: Effect of Separation Ratio on the Number of Clusters

Data Set / SV	a1	a2	a3	s1	s1m1	s1m2	s2	s2m1	s2m2	s3	s3m1
0.5 or 0.6	20	35	50	15	14	13	15	13	12	15	14
0.6, 0.7, 0.8, or 0.9	20	35	50	15	14	13	15	13	12	15	14
1.0 or 1.1	20	**17**(35)	50	15	14	13	15	13	12	15	14
Data Set / SV	**s4**	**s4m1**	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**	**t8**	
0.5 or 0.6	**14**(15)	**11**(13)	5	10	8	10	10	5	5	5	
0.6, 0.7, 0.8, or 0.9	**14**(15)	**11**(13)	5	10	8	10	10	5	5	5	
1.0 or 1.1	**14**(15)	**11**(13)	5	10	8	10	10	5	**4**(5)	**3**(5)	

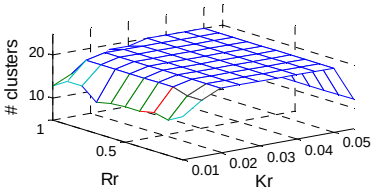
Table 4.3 illustrates the effect of separation ratio SV on the number of clusters resulting from automatic clustering. In this experiment, all parameters were kept the same as given in (4.2) to (4.5), except that SV was varied from 0.5 to 1.1, incremented by 0.1. In Table 4.3, numbers in bold underlined font-style represent the results that are different from those used to generate the clusters ("correct" numbers given in parentheses), while the normal font-style indicates the results are consistent with those used to generate the data. The numbers of clusters for all data sets are consistent while SV changes from 0.6 to 0.9. Results for data set a2, t7, and t8 are "incorrect" for SV=1.0 and 1.1. For all tested values of SV, data set s4 and s4m1 are "incorrect".



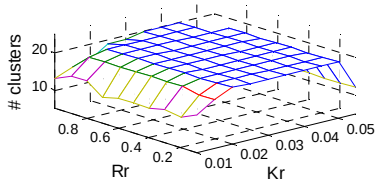
SV=0.5



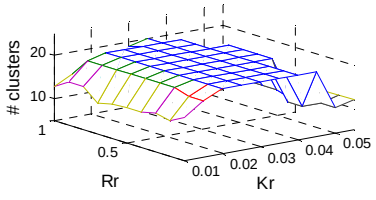
SV=0.6



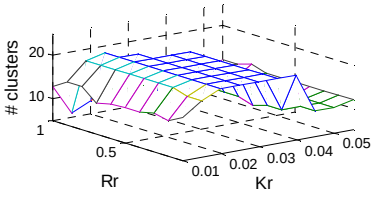
SV=0.7



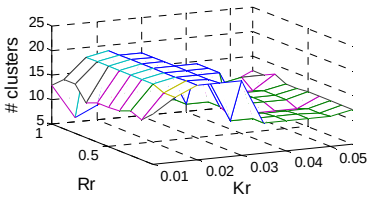
SV=0.8



SV=0.9



SV=1.0



SV=1.1

Figure 4.5: Plots of Number of Clusters vs. k_r and r_r for Different Values of SV (from 0.5 to 1.1 as labeled in each plot) for data set a1.

Figure 4.5 illustrates the effect of separation ratio on the size and shape of the plateau region of the plot of number of clusters vs. k_r and r_r for data set a1. Even though the plateaus of these plots occur at the same height, so that the clustering algorithms can select the correct number of clusters as 20, the plateau regions for SV=1 and SV=1.1 are smaller in size comparing with others.

From experience, we propose the rule of thumb:

$$0.6 \leq SV \leq 1.0 \qquad (4.6)$$

4.2 Case study: texture classification

In this section, we test our proposed clustering method on a real world application of texture classification. Texture is a phenomenon that is widespread, easy to recognize and hard to define. In order to analyze and/or classify texture, we need a mechanism to represent texture accurately so that each texture image will correspond to a point in high dimensional feature vector space. The fundamental assumption is that this representation is matched (or at least most likely matched) with human visual assessment. In other word, textures whose represented feature vectors are similar should be visually similar. Commonly, it suggests representing textures in terms of the response of a collection of filters (such as a Laplacian Pyramid or Gabor filter bank [25, 26]), in which each filter would recover a pattern of the texture, such as a spot or bar (with different sizes and orientations). To summarize the filtered output images, a set of statistics, such as mean, variance, kurtosis, and skewness, is commonly involved.

The 59 texture images used in this experiment were extracted from the Brodatz texture image library [27]. Because the available texture image library is limited in number, we manually divide each original texture image, 640x640 pixels, into small 100 blocks of size 64x64 pixels each, and manually label these 5900 small images into 26 categories based on their visual appearance. Figure 4.6 shows 26 small texture images, each from different categories. For each small image, a Gabor filter bank with 4 scales and 6 orientations (24 totally different Gabor filters) are convolved with this image to produce 24 filtered images. We note that the filtered images have large intensity values (strong responses) at locations where the structure of the original image matches well with the structure defined in the corresponding filters. We then summarize each of the 24 filtered output images by a statistics measurement, namely variance. Hence, each texture image of size 64x64 pixels corresponds to a point in 24 dimensional space and will be classified in this space. These 5900 vectors in 24 dimensional space, each corresponding to a small texture image, were fed into the automatic clustering procedure described in Table 3.3, with the same parameter settings given in (4.1) to (4.5), except that we used SV = 0.6.

Our proposed clustering method correctly produced 26 clusters with an accuracy of 88.5%. However, in order to fairly compare our method with the K-means method, which requires knowing the number of clusters and is sensitive to initialization, we randomly selected 780 feature vectors corresponding to 260 small texture images (30 images from each of 26 categories) to initialize 26 cluster centers, and we forced M-ART to create no new clusters. The remained 5120 small texture images were classified by M-AT and the K-means method. M-ART produced 94.7% accuracy, compared with 91.2% accuracy from the K-means method.

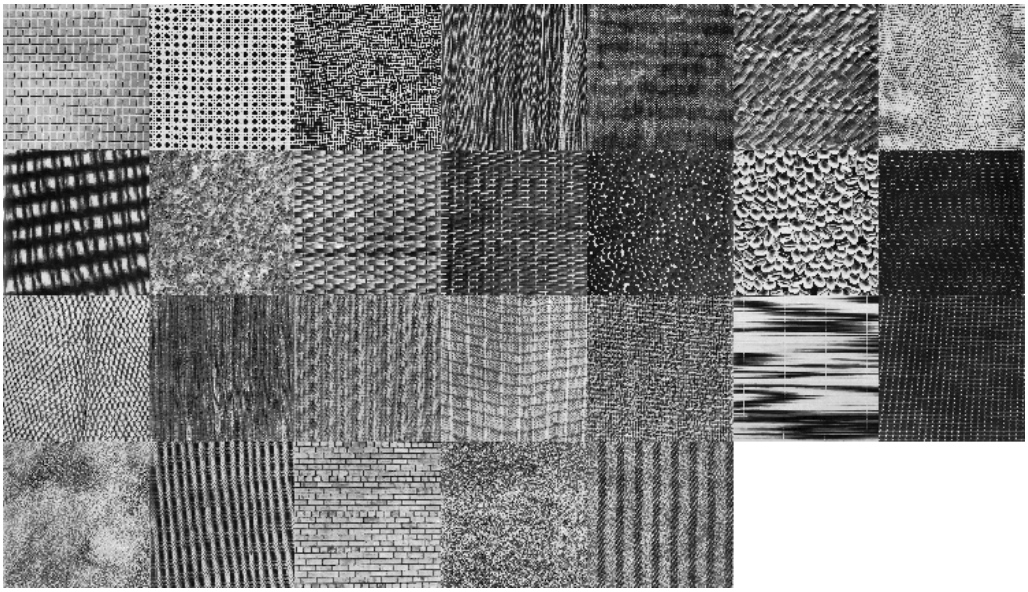


Figure 4.6: The 26 small texture images of size 64x64 pixels from 26 categories.

4.3 Case study: Texture segmentation

In this section, we test our proposed clustering method on a real world application of texture segmentation. The data set includes 50 mosaic images of size 256x256 pixels created by composing 3-6 different gray-scaled textures (chosen randomly from [26]) into designed regions. Figure 4.7 shows five mosaic images and their ground truth maps (in pseudo color).

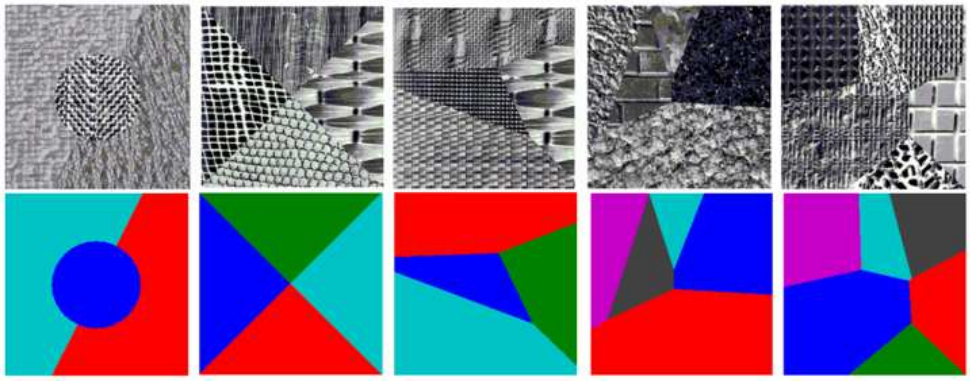


Figure 4.7: Some Test Images for Segmentation (top row)
and Their Ground Truth Maps (bottom row)

For each pixel of a given image, the corresponding 24x1 dimension feature vector was extracted by convolving a Gabor filter bank [27-29] with 4 scales and 6 orientations (24 totally different Gabor filters) with this image. These 65536 vectors in 24 dimensional space, each corresponding

to a pixel in the original image, were fed into the automatic clustering procedure described in Table III, with the same parameter settings given above except that we used SV = 0.7. The order of feeding input vectors to our M-ART was randomly selected.

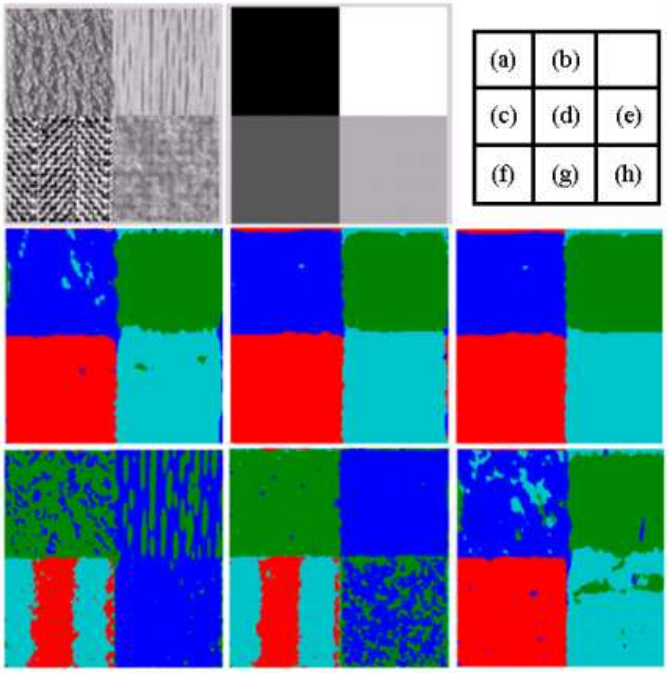


Figure 4.8: Effect of Initialization. (a) the Test Image; (b) the Ground Truth Map; (c), (d), and (e) the Results of K-means, EM, and Our Method, respectively, with a Good Initialization; (f), (g), and (h) the Results of These 3 Methods with a Bad Initialization.

Results were compared with those from the well-known K-means and Expectation Maximization (EM) [30] methods and tabulated in Table VI. Note that K-means and EM both require knowing the number of segments a priori, but our method does not. However, all three methods were given the same initialization for fair comparisons. Segmented images resulting from the three methods were compared pixel-by-pixel with corresponding ground truth maps to determine accuracy.

In Table 4.4, good/bad initialization means each cluster center was given as a pixel inside the correct/incorrect segments of the original image. Figure 4.8 visually illustrates the effect of initialization on the three methods. Notice that the K-means and EM methods are severely affected by bad initialization. Our method is superior in both accuracy and insensitivity to initialization.

Table 4.4: Accuracy of Texture Segmentation of Three Methods

	K-means	EM	**Our method**
Good initialization	87.2%	91.4%	**97.5**%
Bad initialization	55.8%	62.6%	**85.2**%
Average of 100 random initializations	68.5%	72.3%	**89.8**%

Chapter 5

CONCLUSTIONS FOR DATA CLUSTERING

In previous chapters, an automatic clustering procedure is proposed. The first step is density-based removal pre-processing, which tends to produce more distinct clusters in most cases. This process is equivalent to removing input vectors near the boundaries of each cluster, which in many cases does not change the data structure or the number of "correct" clusters in the data. The next step is to employ the M-ART neural network to group similar input vectors into clusters.

The vigilance ρ in the M-ART network determines the maximum size of clusters, and consequently affects the number of clusters. Conventionally, a trial-and-error approach is used to tune this value of ρ. In this paper, a procedure to auto-adjust the value of ρ based on a user-selected allowable separation between clusters is proposed. Even though one must still select the value of a parameter, choosing the allowed separation factor is intuitively easier than selecting the value of ρ. The appropriate value of ρ is strongly dependent upon on the specific data set, and is therefore very difficult to select a-priori. On the other hand, the allowed separation factor value simply represents how much separation between adjoining clusters a user is willing to accept. Accordingly, the separation factor could be chosen for multiple data sets before running the algorithm.

The “optimal” number of clusters could be chosen according to some criteria, such as cluster compactness or variation within a cluster and/or separation or isolation between clusters [11]. Cluster compactness (variation within cluster) and/or separation (isolation between clusters) are normally considered as major factors in forming validation indexes [10, 11]. Almost all clustering algorithms are not parameter-free and require user supplied values for input parameters. Determining these values is difficult, and is usually guided by trial-and-error. Moreover, the results produced could be very sensitive to these values, producing significantly different partition results with only slightly different parameter values [6], rendering them unusable.

Experiments conducted on different synthetic 2-D, 3-D, 4-D, 5-D, and 10-D Gaussian data sets, some published and some generated by the authors, with varying numbers of vectors, numbers of clusters, and different degrees of separation between clusters, demonstrate the effectiveness and reliability of the proposed clustering method. Two case studies of texture classification and texture segmentation are also presented, showing very good results when compared with those from the well-known K-means method.

We would like to investigate and solve the segmentation problem for general, real scene color images. In applying our automatic clustering technique presented herein to this problem, we represent each image pixel as a vector in high dimensional feature space, which are usually based on color, texture, and **xy**-coordinators in the image plane. Then these vectors are grouped into clusters, which is equivalent to dividing image pixels into corresponding segments. However, our method works well only with convex-shaped (ellipsoid-shaped) clusters that are sufficiently separated, which is normally not the case with general, real scene images. Accordingly, in the next chapters, we investigate and propose a more advanced technique for color image segmentation.

PART II: IMAGE SEGMENTATION

Chapter 6

INTRODUCTION TO IMAGE SEGMENTATION

6.1 Introduction to image segmentation

Image segmentation is a process in computer vision that partitions a digital image into multiple segments or non-overlapping regions. Partitioning an image into non-overlapping regions assures that pixels in each region share the same visual characteristics, such as color or texture, while pixels in different regions exhibit significant differences in these features. In other words, segmentation can be viewed as the process of labeling all pixels of the input image such that pixels with the same label are connected and share certain visual properties. Moreover, pixels in adjacent regions with different labels are significantly different in the same criteria. The result of this process is a set of non-overlapping segments whose union forms the whole input image. Segmentation is one of typical methods to separate the foreground from the background and to locate objects (and their boundaries) of the input image. It is a first step to simplify and represent an input image into a form that is more meaningful and easier to analyze. Then, properties of objects resulting from the segmentation process can be determined (such as size, shape, color distribution) for purposes of recognition, classification, and forming higher knowledge.

Therefore, segmentation serves as a fundamental step in extracting knowledge from the image, and can be widely applied in many fields, such as classification, object recognition, object tracking, content-based image retrieval, surveillance, and medical imaging, among others [31-34]. Some of the practical applications of image segmentation are: medical imaging (including locating tumors and other pathologies, measuring tissue volumes, and computer-guided surgery [34]), locating and measuring objects in satellite images (roads, forests, etc.) [35], face recognition [36], iris recognition [37], fingerprint recognition [38], recognition in traffic control systems [39], and crop disease detection [40].

Several general-purpose algorithms and techniques have been developed for image segmentation. Since there is no general solution to the image segmentation problem, these techniques often must be combined with domain knowledge in order to effectively solve an image segmentation problem for a specific problem domain. Most algorithms work well with specific, well-known scene images or images under well controlled lighting condition, but fail with general scene images [35-37, 39-40]. A general-purpose segmentation technique is needed that provides acceptable and reliable results on a wide variety of real, general scene images without excessive computational cost. Another challenge in segmentation problems is how to quantitatively evaluate a given image segmentation method, of which there are many approaches [39, 41-47].What constitutes good segmentation is a problem similar to what constitutes good clustering, mainly because of the lack of a precise definition of “good” clustering results or segmentation results [47-48]. Accordingly, it is difficult to compare two given segmentation techniques. Normally, results of a segmentation method are compared with manually segmented result by humans on a set of test images (ground truth segments). However, it is very time consuming and tedious to construct such a ground truth database. Even worse, different persons often provide significantly different segmentation results on the same image.

6.2 Previous work

6.2.1 Histogram-based methods

In histogram-based methods, the characteristics of the intensity, or color, histogram of all image pixels, such as peaks and valleys, are used to separate clusters in the histogram, and therefore separate the corresponding segments in the image [49]. Since each pixel is addressed only once in each application, these methods are efficient comparing with other segmentation methods.

Improvements in these techniques include applying the histogram-seeking method recursively to segments in the image in order to divide them into smaller segments. Specified stopping criteria are applied to terminate the repetition when no more segments are formed [49, 50]. However, there are several drawbacks of histogram-based methods. For example, the peaks and valleys in the histogram of the image can be difficult to identify due to natural noise in pixel assesment. Poor segmentation results might be expected from inappropriate detection of peaks and valleys. Even worse, small changes in these peak and valley positions could produce significantly different segmentation results. Another disadvantage of this method is that small objects (that might be important in the image) might not show up in the histogram, and therefore will be ignored in segmentation results [50].

6.2.2 Feature-space- based methods

Similar to histogram-based techniques, feature-space-based clustering approaches ignore spatial information in the image. These methods represent each image pixel as a vector in high dimensional feature space, which is usually based on color or texture. Then a clustering algorithm, such as our M-ART [51] discussed in Part I of this dissertation, is employed to separate these vectors into clusters. Image segmentation is essentially a clustering process in which each pixel in the image corresponding to a vector in the high-dimensional feature space is grouped into an appropriate class or cluster. A distance between two vectors in this feature space

is defined to represent similarity in a visual characteristic, such as color or local information, of two corresponding pixels. Then vectors or pixels can be partitioned into clusters such that those in the same cluster share similar characteristics, while those in difference clusters exhibit significantly differences in these characteristics.

Some well-known and simple clustering techniques, such as K-means [52] and mean shift [53], are often used for segmentation of a simple image which contains an object that is significantly different from the background. The K-means algorithm is an iterative technique consisting of the following steps: (1) Initialize K cluster centers (randomly or based on some heuristic); (2) Assign each vector to the closest cluster center (based on a predefined distance metric, such as Euclidean distance); (3) Re-assign the cluster centers by averaging all of the vectors that belong to the cluster; (4) Repeat the assigning and re-assigning steps until some stopping criteria are met (e.g. no pixels change clusters). Both K-means and mean shift algorithms are guaranteed to converge, but they may not return the optimal solution. The quality of the solution depends on the initial set of clusters and the value of K. The main drawback of these algorithms is that the number of clusters K is an input parameter, which is almost always unknown. An inappropriate choice of K may yield poor results. Furthermore, in this approach, the image spatial structures, such as edges, are not preserved, and pixels from disconnected image regions can be placed in the same group. One approach to avoid the effect of the number of clusters K is employed in our M-ART algorithm (presented in Part I). However, all three techniques (K-means, mean-shift, and M-ART) work only for convex-shaped clusters.

6.2.3 Graph partitioning methods

An image can be modeled as a weighted, undirected graph, in which a pixel or a group of pixels is associated with nodes of the graph, and the similarity or dissimilarity measure (in some visual characteristics such as color and texture) between the neighborhood pixels or groups of pixels is associated with weights of edges in the graph (The terminology “edge” in the graph theory means

the connection between two nodes of a graph; and should not be confused with edges/boundaries of objects in an image). Then various graph partition techniques, such as normalized cuts [54], minimum cut [55] and minimum spanning tree partitioning [56], can be employed for segmenting images of interest. Each segment in the image corresponds to a partition of the nodes in the graph produced from these algorithms. The graph partitioning results, and hence the image segmentation results, are dependent upon how these techniques define a "good" cluster (of nodes). Often a global, fixed, and predefined threshold is needed. Unfortunately, the results might change significantly due to small change in this threshold [54-56].

6.2.4 Region-growing methods

Region-growing methods, e.g. seeded region growing [57], initialize a set of “seeds” or pixel locations of the input image such that these seeds are considered as regions or objects to be segmented. Then at each step, an unallocated neighboring pixel to a region is grouped into an appropriate region according to some criteria. One simple approach uses the difference between the region's mean and the pixel's intensity value as a measure of similarity. The region with the smallest difference measured is the region into which the pixel is grouped. This iterative process is done only after all image pixels are visited. Since seeded region growing requires the user to provide the seeds, segmentation results very dependent upon seed choices. Noise in the image can cause the seeds to be poorly placed [58, 59].

Improvements, called unseeded region growing [60], do not require explicit initial seeds. This algorithm starts with random seeds. At each step, a neighboring pixel is either grouped into the appropriate region as in seeded region growing, or a new region is formed. One simple approach is predefining a threshold to decide whether or not to form a new region. If the smallest difference between the neighboring pixel’s intensity value and an existing region’s mean is greater than the predefined threshold, a new region is created with this pixel. A more advanced unseeded region growing technique, introduced in the work herein, will be discussed in Chapter 7.

6.2.5 Watershed-based methods

The watershed algorithm can be viewed as a special type of region-based segmentation [61, 62]. In this algorithm, the gradient magnitude of pixel intensity in an image is considered as a topographic surface. The technique envisions that water placed on any pixel enclosed by a common watershed line flows downhill to a common local intensity minimum. Then, a catch basin formed from pixels draining to a common minimum presents a segment. The main drawback of watershed-based methods is that they often produce over-segmentation of the image, in which many small basins are produced due to many local minima in the real-scene input image [61].

6.3 Proposed image segmentation method

In this work, we introduce a general-purpose segmentation method that works for a large variety of natural scene images in color, with reasonable computational times. The proposed method is a type of unseeded region-based segmentation technique that preserves the spatial relationship between pixels in the image, and hence preserves the detailed edges and the image spatial structure. There are number of important modifications made in our proposed method. First, our method operates at a "superpixel" [63] level, rather than at the image pixel level. The original region growing techniques that operate directly at the pixel level often produce undesirable small, but quasi-homogeneous, regions and are computational expensive [63]. By utilizing superpixels, the proposed method avoids both issues. Second, the proposed method works for both color and gray images rather than for only gray scale images as used in the original techniques [57-60]. Our similarity measurement is defined based on statistics, mainly interquartile range, of pixel color (in **L*a*b** color space [64]) in regions and neighboring superpixels. Third, the decision of grouping an adjacent superpixel to an existing region is dynamically depended upon the statistics, or "shape and size" of this region. The segmentation results show significant improvements when compared with using a fixed, global threshold as used in the original techniques.

The reminder of this report is organized as follows. In Chapter 7, we introduce the process of creating superpixels and our modified region growing technique. In Chapter 8, we demonstrate the results of the proposed method on a large number and variety of colored natural scene images. Evaluation and comparing the performance of the proposed method with existing methods are presented in Chapter 9, and Chapter 10 presents conclusions and recommendations.

Chapter 7

NEW IMAGE SEGMENTATION METHOD

In this chapter, we introduce a new general purpose segmentation method that works for a large variety of colored natural scene images with reasonable computational times. Our method consists of two main steps: (1) Over-segment an input image into many small segments (called superpixels); (2) Then apply a modified version of unseeded region-growing on these superpixels to obtain the final segmentation. There are a number of advantages to employ the region-growing technique at the superpixel level, rather than at the image pixel level. For example, it is more computationally efficient because of reducing the complexity of images from hundreds of thousands of pixels to only a few hundred superpixels. The superpixels are also perceptually meaningful in such a way that each superpixel is a consistent unit consisting of pixels most likely uniform in color and texture. More importantly, in the process of generating superpixels, most structures in the image, such as edges, are conserved.

The superpixel formulation is introduced in Section 7.1. The modified region-growing technique is provided in Section 7.2, and Section 7.3 summarizes our proposed image segmentation method.

7.1 Superpixels

Superpixels are becoming increasingly popular for use in computer vision applications [33, 39, 41]. Our superpixel algorithm, based on the idea of SLIC (Simple Linear Iterative Clustering [63]) that produces a desired (predefined) number of regular, compact superpixels with low computational overhead. The superpixels preserve the detail edges and the spatial structure of an input image, and hence prevent pixels from disconnected regions of the image from being grouped together. Our approach generates superpixels by clustering pixels based on their color similarity and proximity in the image plane. This is done in the five-dimensional (**L*a*b*xy**) space, where (**L*a*b***) is the pixel color in CIE-LAB [65] color space, and (**xy**) is the pixel coordination in the image plane (pixel location).

The first coordinate of the CIE-LAB represents the lightness of the color (**L*** = 0 yields black and **L*** = 100 indicates diffuse white). The last two coordinates represent the relative colors, where a* indicates color between magenta and green (**a*** = -128 indicates green and **a*** = +127 indicates magenta); and **b*** indicates color between yellow and blue (**b*** = -128 indicates blue and **b*** = +127 indicates yellow). The asterisks (*) after **L**, **a** and **b** are part of the full name, (**L***, **a*** and **b***), to distinguish them from Hunter's **L**, **a**, and **b** [65]. The nonlinear relations for **L***, **a***, and **b*** are intended to mimic the nonlinear response of the eye. Furthermore, uniform changes of components in the **L*a*b*** color space aim to correspond to uniform changes in perceived color, so the relative perceptual differences between any two colors in **L*a*b*** can be approximated by taking the Euclidean distance between two corresponding points in this three dimensional color space. The **L*a*b*** color space is widely considered as perceptually uniform for small color distances. Figure 7.1 illustrates the **RGB** and **L*a*b*** color space representation.

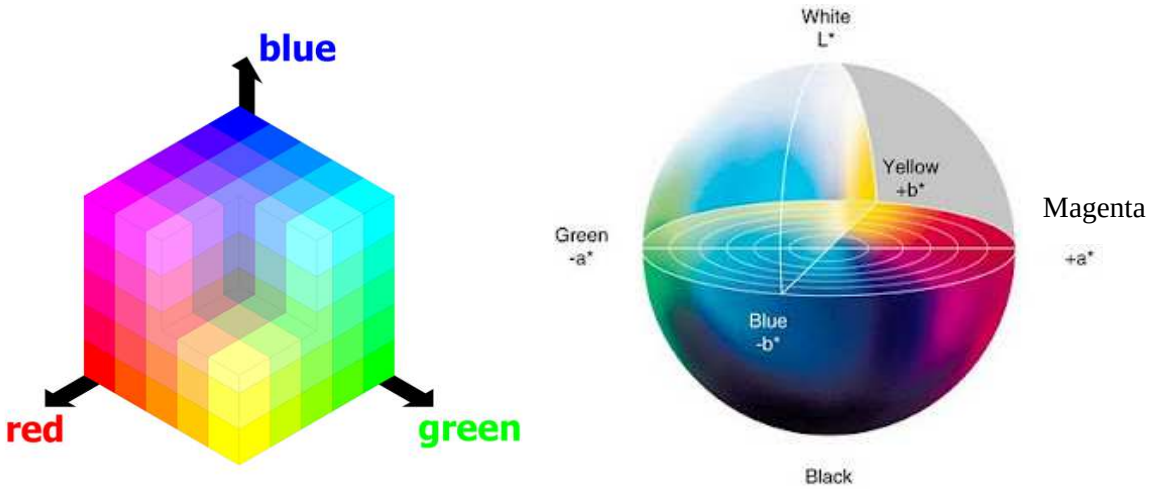


Figure 7.1: **RGB** (left) and **L*a*b*** (right) color space. Pictures from [66]

Notice that, in Matlab's unsigned 8-bit integer representation, the **L*** coordinate ranges from 0 to 100, while **a*** and **b*** coordinates range from 0 to 255. The conversion between **RGB** and **L*a*b*** color space normally takes an intermediate conversion through CIE-XYZ [65] color space. Equation (7.1) shows the linear relationship between **RGB** and **XYZ** color space [64]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{7.1}$$

Equation (7.2) shows the conversion from **XYZ** to **L*a*b*** color space:

$$\begin{aligned} L^* &= 116 f(Y/Y_n) - 16 \\ a^* &= 500\big(g(X/X_n) - g(Y/Y_n)\big) \\ b^* &= 200\big(g(Y/Y_n) - g(Z/Z_n)\big) \end{aligned} \tag{7.2}$$

where $g(t) = \begin{cases} t^{1/3} & \text{if } t > (6/29)^3 \\ \left(\frac{29}{6}\right)^2 \frac{t}{3} + \frac{4}{29} & \text{otherwise} \end{cases}$

and X_n, Y_n and Z_n are the CIE-XYZ tri-stimulus values of the reference white point (the subscript n suggests "normalized", and the white point value is dependent upon the hardware device used to display color images).

Our superpixel generating algorithm is essentially a K-mean based clustering in 5D (**L*a*b*xy**) space. The idea of utilizing K-mean clustering for superpixel generation was first introduced in

SLIC (Simple Linear Iterative Clustering) by R. Achanta, 2010 [63]. In our work herein, the distance calculation measuring the similarity between two points in the 5D space is generalized. Note that five coordinates in this space represent different properties of a pixel: **L*** for light intensity, **a*** and **b*** for color, and **x** and **y** for spatial coordinates in the image plane. While the maximum possible distance between two color points in the CIE-LAB space is limited, the spatial distance in the xy plane depends on the image size. It is not possible to simply use the Euclidean distance in this 5D space (**L*a*b*xy**) without normalizations. Each 5D point (or vector) $\boldsymbol{p}_k = [l_k, a_k, b_k, x_k, y_k]^T,\ 1 \le k \le N$in **L*a*b*xy** space, corresponds to a pixel in the given image, where the upper-script $(.)^{\mathrm{T}}$means the vector/matrix transpose operation. Each such point is then normalized, producing $\overline{\boldsymbol{p}_k}$, such that each element $\mathrm{l_k}, \mathrm{a_k}, \mathrm{b_k}, \mathrm{x_k}, \mathrm{y_k}$ is in the range [0,1]:

$$\overline{\boldsymbol{p}_k} = (\boldsymbol{p}_k - \boldsymbol{p}_{min})./(\boldsymbol{p}_{max} - \boldsymbol{p}_{min}) \tag{7.3}$$

where (./) means an element-by-element division of two vectors, and $\mathrm{p_{min}}$ ($\mathrm{p_{max}}$) is a new vector in which each element is the minimum (maximum) over all corresponding elements of all N input vectors, namely [66]:

$$\boldsymbol{p}_{min} = \left[\min_{1\le k\le N}(l_k), \min_{1\le k\le N}(a_k), \min_{1\le k\le N}(b_k), \min_{1\le k\le N}(x_k), \min_{1\le k\le N}(y_k)\right]^{\mathrm{T}} \tag{7.4a}$$

$$\boldsymbol{p}_{max} = \left[\max_{1\le k\le N}(l_k), \max_{1\le k\le N}(a_k), \max_{1\le k\le N}(b_k), \max_{1\le k\le N}(x_k), \max_{1\le k\le N}(y_k)\right]^{\mathrm{T}} \tag{7.4b}$$

To simplify the notation, from now on, we omit the upper bar from a vector $\boldsymbol{p}_k$ remembering that all five elements of this vector have been normalized in range [0,1].

After normalization, in order to cluster pixels in **L*a*b*xy** space, we introduce a generalized weighted Euclidean distance $\mathrm{d_{ik}}$ between two points (or vectors) $\boldsymbol{p}_i = [l_i, a_i, b_i, x_i, y_i,]^T$ and $\boldsymbol{p}_k = [l_k, a_k, b_k, x_k, y_k,]^T$in this 5D space as follows:

$$\begin{aligned} d_{ik}^2 &= \boldsymbol{W}^T(\boldsymbol{p}_i - \boldsymbol{p}_k)^2 \\ &= w_l(l_i - l_k)^2 + w_a(a_i - a_k)^2 + w_b(b_i - b_k)^2 + w_x(x_i - x_k)^2 + w_y(y_i - y_k)^2 \end{aligned} \tag{7.5}$$

where $W = [w_l, w_a, w_b, w_x, w_y]^T$ is a predefined weight vector. Normally, $w_a = w_b$, and $w_x = w_y$, since there is no specific reason to weight **a*** and **b*** color as well as **x** and **y** spatial coordinates differently.

As a K-mean approach, our algorithm consists of four main steps: (1) Select *K* cluster centers (detailed below), with each cluster forming a superpixel; (2) Assign each pixel in the image to the cluster (or superpixel) that minimizes the generalized weighted Euclidean distance between the pixel and the superpixel center; (3) Re-compute the superpixel centers by averaging the **xy** coordinates of all the pixels in the superpixel; (4) Repeat the assigning and re-computing steps until a stopping criteria is achieved (detailed below). Notice that a desired number of approximately equally-sized superpixels, *K*, is assumed known as an input of our algorithm. For an image with *N* pixels, the approximate size of each superpixel is therefore *N/K* pixels. For roughly equally sized superpixels there would be a superpixel center at every grid interval $S = \sqrt{N/K}$. Notice that the size of each superpixel represents the area (in pixels) of the smallest image region that will be differentiated after the process. We call this area "the smallest feature size *SF*". After superpixel generation, any region that is smaller than *SF* is averaged or "smoothed" out. In order to retain image details or to produce fine segmentation, one should select a small value of *SF*. Coarse segmentations are obtained with large values of *SF*. Users can select the "appropriate" value of *SF* depending upon the input image characteristic and the level of coarse or fine segmentation they would like to obtain. With the user-defined smallest feature size *SF*, the number of superpixels *K* can be calculated as:

$$K = \frac{N}{SF} \tag{7.6}$$

The default values of *SF* (in the event the user does not select) and *K* in our program are:

$$\begin{aligned} SF_{default} &= \frac{N}{30 \times 30} \\ K_{default} &= \frac{N}{SF_{default}} = 900 \end{aligned} \tag{7.7}$$

In this case, any image feature that is smaller than a region of 30 pixels image width by 30 pixels image height will be ignored.

Table 7.1: Proposed Superpixel Generating Algorithm

- Input the smallest feature size SF and the weight vector W. For our work herein $\boldsymbol{W} = [1,1,1,0.2,0.2]^T$.
- Normalize all vectors in 5D space (**L*a*b*xy**) corresponding to all image pixel as in (7.3).
- Initialize $K = N/SF$ superpixel centers $\boldsymbol{C}_k, 1 \leq k \leq K$ by sampling pixels at regular grid size $S = \sqrt{N/K}$, where N is the number pixels of the given image.
- Assignment step: The i^{th} pixel$\boldsymbol{p}_i$, $1 \leq i \leq N$, will be assigned to the "nearest" superpixel according to the minimum weighted Euclidean distance measurement as in (7.7).
- Update step: Compute new superpixel centers and number of pixels changing into different superpixels compared with previous iteration.
- Repeat Assignment step until stopping criteria is met.

The details of these four steps are:

- **Initialization step**:

We first initialize K superpixel cluster centers, as vectors $\boldsymbol{C}_k = [l_k, a_k, b_k, x_k, y_k]^T$ in **L*a*b*xy** space and $1 \leq k \leq K$. These centers are initialized at regular grid intervals of S on the given image, and each center vector $\boldsymbol{C}_k$ is a mean of 5D vectors corresponding to all pixels that belong to the k^{th} superpixel:

$$\boldsymbol{C}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{p}_i \tag{7.8}$$

where $\boldsymbol{p}_i = [l_i, a_i, b_i, x_i, y_i,]^T$ and N_k is the number of pixels that belong to the k^{th} superpixel.

- **Assignment step**:

At each iteration, every pixel $\boldsymbol{p}_i$ will be assigned to the closest superpixel, called the J^{th} superpixel, by the weighted Euclidean distance d_{ik} defined in (7.5).

$$J = \underset{1 \leq k \leq K}{arg\,min}(d_{ik}) = \underset{1 \leq k \leq K}{arg\,min}\, \boldsymbol{W}^T(\boldsymbol{p}_i - \boldsymbol{C}_k)^2 \tag{7.9}$$

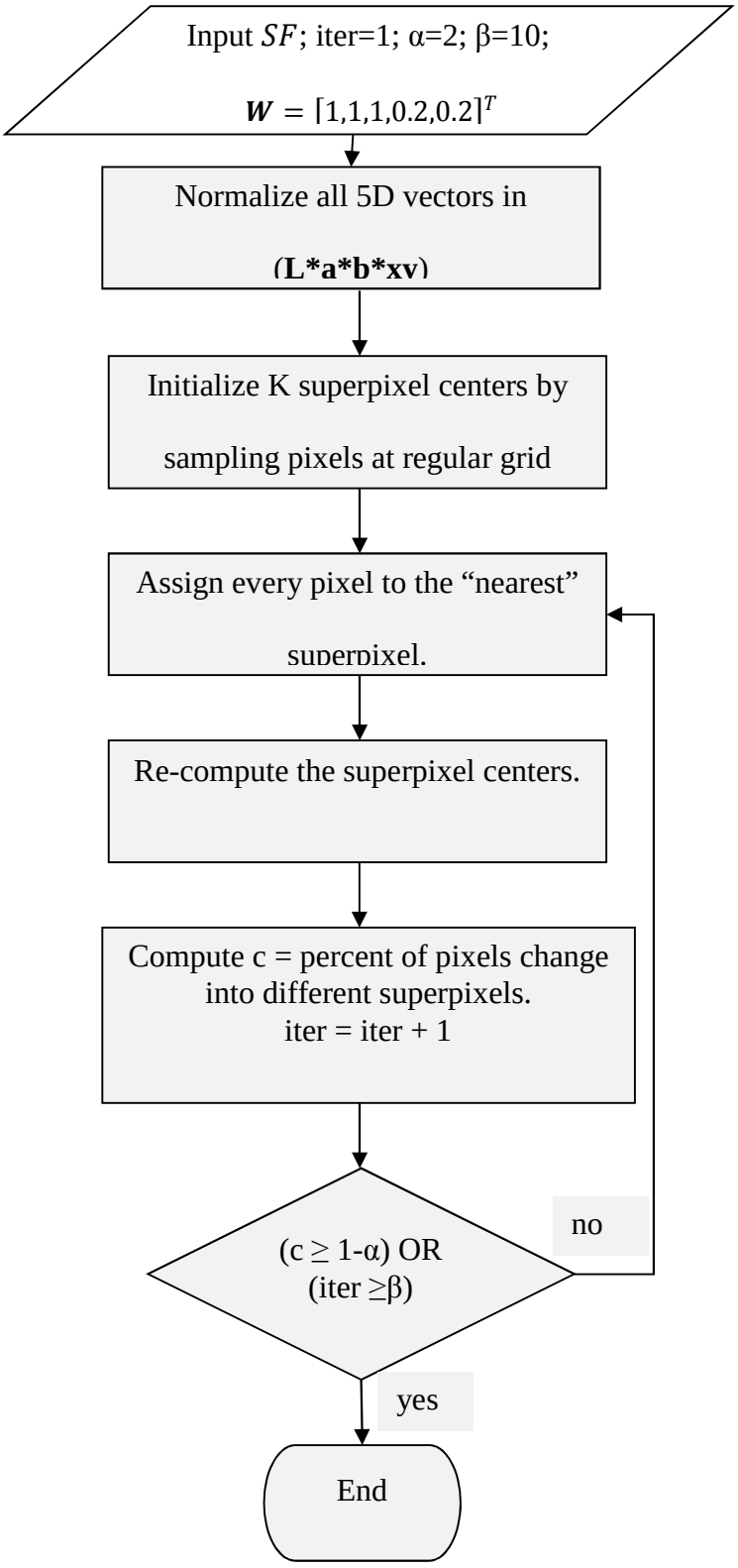


Figure 7.2: Proposed Superpixel Generating Algorithm

- **Update step**:

At the end of each iteration, after re-assigning all pixels, every superpixel cluster centers will be updated as in (7.8).

- **Stopping criteria**:

It is shown in [60] that given enough iterations, the K-mean algorithm will converge to a local minimum. Accordingly, we propose that our algorithm stops when no more than α percent of all pixels change to a different superpixel (compared with the previous iteration), or when maximum of β iterations is reached, whichever occurs first. In what follows, we select α = 2 and β = 10. Our superpixel generating algorithm is summarized in Table 7.1 and Figure 7.2.

Figure 7.3: Superpixel Generating: Effect of $\boldsymbol{W}$
Original Image (top-left) and superpixel results with: $\boldsymbol{W} = [1,1,1,0.04,0.04]^T$ (top-right); $\boldsymbol{W} = [1,1,1,0.2,0.2]^T$ (bottom-left); and $\boldsymbol{W} = [1,1,1,1,1]^T$ (bottom-right) in which the boundaries (in black) of superpixels are overlaid onto the original image.
(For all results: K=200 or $SF \cong$ 32x32)

Figure 7.3 presents an example of superpixel generation with different weight vectors $\boldsymbol{W} = [w_l, w_a, w_b, w_x, w_y]^T$. Choosing these weights allows us to control the effect of each pixel properties, e.g. intensity, color, and spatial location, and hence control the compactness of a superpixel. The greater the value of w_x and w_y the more spatial proximity is emphasized, and the more compact is the superpixel, and vice versa. By selecting appropriate weights, we enforce color similarity as well as pixel proximity in this 5D space, such that the expected superpixel sizes and their spatial extent are approximately equal. We choose $\boldsymbol{W} = [1,1,1,0.2,0.2]^T$, or $w_l = w_a = w_b = 1$ and $w_x = w_y = 0.2$, for all the results in this paper. This roughly matches the empirical maximum perceptually meaningful CIELAB distance, and offers a good balance between color similarity and spatial proximity.

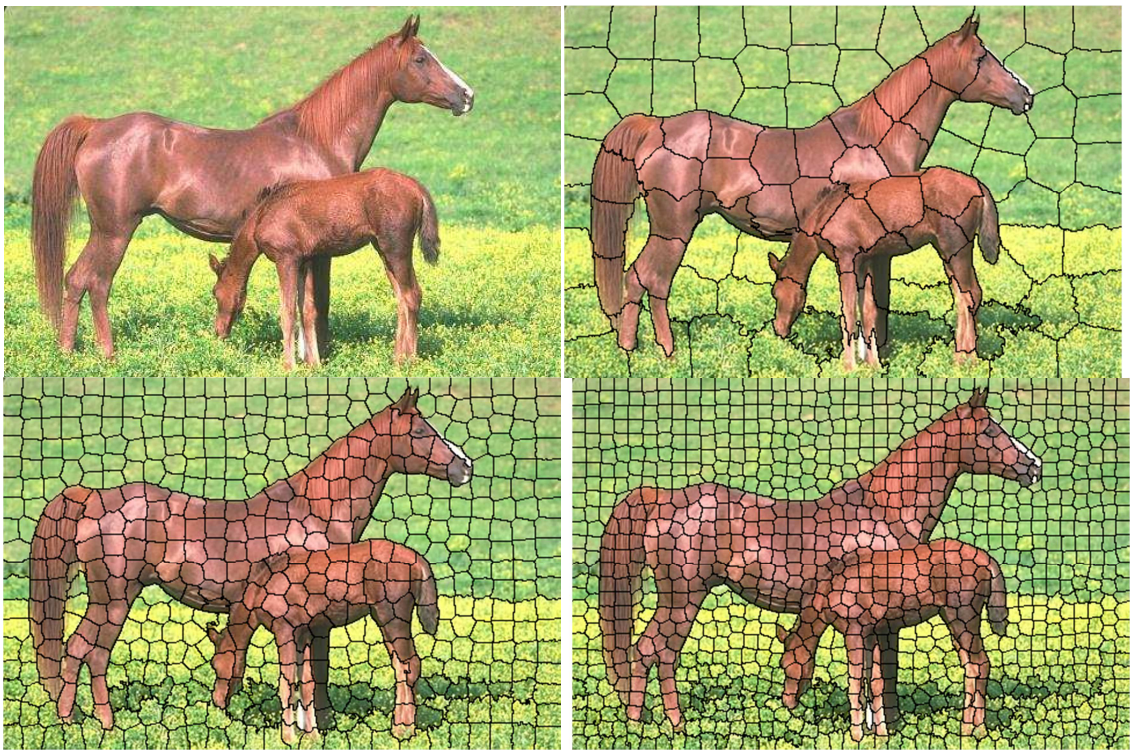


Figure 7.4: Superpixel Generating: Effect of *K* and *SF*
Original Image (top-left) and superpixel results with *K = 100, SF = 2000 ≅ 45x45* (top-right); *K=500, SF = 400 = 20x20* (bottom-left); and *K = 1000, SF = 200 ≅ 14x14* (bottom-right) in which the boundaries (in black) of superpixels are overlaid onto the original image.
(For all results: $\boldsymbol{W} = [1,1,1,0.2,0.2]^T$)

Figure 7.4 presents an example of superpixel generation with different desired numbers of superpixels *K*. Notice that after superpixel generation, any region that is smaller than *SF* is averaged or "smoothed" out. The parameters *SF* and *K* affect the coarseness or fineness of superpixels.

Notice that these superpixels are approximately equal in size (e.g. consist of a similar number of image pixels). Also notice that edges and spatial and color information are well preserved by superpixels. Each superpixel consists of connected pixels that are quite uniform in color and intensity, as expected. However some superpixels are quite similar and should be grouped into the same segment to form a more meaningful representation of "objects" appearing in the image. The next section on our region-growing technique handles this problem.

7.2 Modified region growing segmentation

Region-growing is a simple region-based image segmentation method. It was first introduced as a pixel-based image segmentation method, and it involved the selection of initial seed points. The basic formulation for region-based segmentation satisfies five conditions, as follows [57]:

Condition 1: $$\bigcup_{i=1}^{n} \boldsymbol{R}_i = \boldsymbol{R} \tag{7.10}$$

where each region $\boldsymbol{R}_i, 1 \leq \text{i} \leq \text{n}$, is a set of pixels: $\boldsymbol{R}_i = \{\boldsymbol{p}_k\}, 1 \leq k \leq N_i$; N_i is number of pixels belong to this region $\boldsymbol{R}_i$, and n is the number of regions (note that $\sum_{i=1}^{n} N_i = N$, where N is the total number of pixels in the input image); $\boldsymbol{R}$ is the entire image region. In other words, condition (2.8) means that the segmentation must be complete such that every pixel is in a region.

Condition 2: $\boldsymbol{p}_k$ and $\boldsymbol{p}_l$ are "connected", $\forall\, \boldsymbol{p}_k, \boldsymbol{p}_l \in \boldsymbol{R}_i, 1 \leq i \leq n$ (7.11)

where $\boldsymbol{p}_k$ and $\boldsymbol{p}_l$ are "connected" if there exists a sequence of pixels $\boldsymbol{p}_m, \ldots, \boldsymbol{p}_j$ such that:

(2.11a): $\boldsymbol{p}_m, \ldots, \boldsymbol{p}_j \in \boldsymbol{R}_i$, all pixels in the sequence are in the set $\boldsymbol{R}_i$

(2.11b): and every 2 pixels that are adjacent in the sequence are "neighbors".

Normally, pixels are considered neighbors if they are 4-connected or 8-connected. For example, in terms of pixel coordinates, every pixel that has the coordinates $(x \pm 1, y)$ or $(x, y \pm 1)$ is 4-connected to the pixel at (x, y). The 8-connected relation includes the 4-connected and in addition, every pixel that has the coordinates $(x \pm 1, y \pm 1)$ or $(x\pm, y \mp 1)$ is 8-connected to the pixel at (x, y).

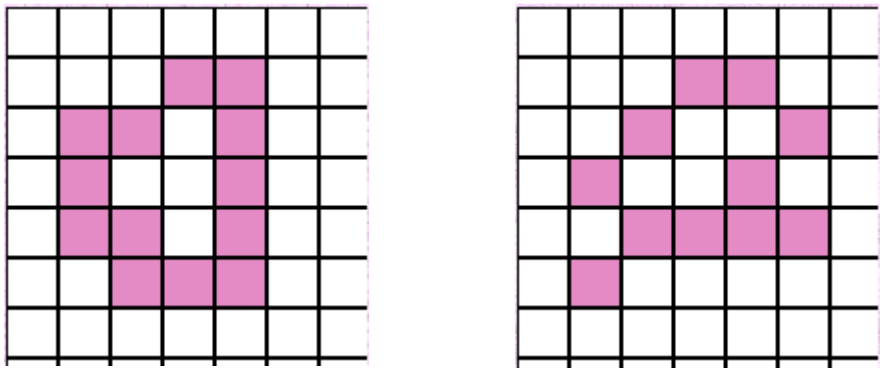


Figure 7.5: The set of pixels (shown as purple squares) are 4-connected (left) and 8-connected (right).

This condition means that every region R_i is a connected region or connected component.

Condition 3: $\boldsymbol{R}_i \cap \boldsymbol{R}_j = \emptyset,\ 1 \le i \ne j \le n$ (7.12)

where $\emptyset$ is the null set. This condition indicates that the regions must be disjoint.

Condition 4: $f(\boldsymbol{R}_i) = TRUE,\ 1 \le i \le n$ (7.13)

Condition 5: $f(\boldsymbol{R}_i \cup \boldsymbol{R}_j) = FALSE, \boldsymbol{R}_i\ and\ \boldsymbol{R}_j\ are\ "adjacent", 1 \le i \ne j \le n$ (7.14)

where $f(\boldsymbol{R}_i)$ is a logical predicate (or Boolean-value function) defined over the set $\boldsymbol{R}_i$, and two regions are adjacent if there exists at least one pixel in each region that are neighbors. Conditions (2.11) and (2.12) suggest that pixels in the same region share the same properties, while pixels in different regions differ in these properties. For example all pixels of a region have a similar gray level.

Different region-growing techniques differ mainly in how the pixel connections are defined and how to specify the $f(\ \ )$ function in (7.13) and (7.14). Our modified region-growing method with new connectivity definition and $f(\ \ )$ function is described at the end of this section.

Seeded region growing:

The first region-growing method introduced in the literature was seeded region growing [57]. This method uses the 4-connected neighbor definition and initializes a set of "seeds" or pixel location of the input image such that these seeds are considered as regions or objects to be segmented. This algorithm starts with n seeds, either by random or heuristic placement. Then at each step, an unallocated neighboring pixel to the regions is grouped into an appropriate region according to some criteria. One simple approach is used the difference between the region's mean and the pixel's intensity value as a measure of similarity. The region with the smallest difference measured is the respective region where the pixel is grouped into. This iterative process is done only after all image pixels are visited. This method uses following f( ) function:

$$\begin{aligned} &\text{Define } J = argmin_{\forall i}\left|I(\boldsymbol{p}_k) - mean\left(I(\boldsymbol{R}_i)\right)\right|,\ CN(\boldsymbol{p}_k, \boldsymbol{R}_i) = TRUE \\ &f\left(\boldsymbol{R}_J\right) = TRUE \\ &\boldsymbol{R}_J \leftarrow \left\{\boldsymbol{R}_J, \boldsymbol{p}_k\right\} \end{aligned} \tag{7.15}$$

where I(.) returns the intensity or gray level of pixel(s) and the notation "A ← B" means that the value of B is assigned to A. $CN(\boldsymbol{p}_k, \boldsymbol{R}_i) = TRUE$ if pixel $\boldsymbol{p}_k$ and region $\boldsymbol{R}_i$are connected, otherwise $CN(\boldsymbol{p}_k, \boldsymbol{R}_i) = FALSE$. Note that in this case, there always exists the region J, $1 \leq J \leq$ n that has the mean intensity closet to the intensity of the neighbor pixel $\boldsymbol{p}_k$, or $f\left(\boldsymbol{R}_J\right) = TRUE$, $1 \leq J \leq n$. The pixel $\boldsymbol{p}_k$ is, then, included in the region $\boldsymbol{R}_J$. The number of regions n is unchanged through the growing process.

Since seeded region growing requires the user to provide the seeds, segmentation results very dependent upon seed choices (e.g., number of seeds and their locations). For example, in the extreme case, where there is only one seed, n =1, the entire image will be grouped into a single region. The location of seeds also affects the growing results, and noise in the image can cause the seeds to be poorly placed [54, 55].

Unseeded region growing:

Unseeded region-growing [59] is a modified version of seeded region-growing that does not require explicit initial seeds. This algorithm starts with n random seeds. At each step, the neighboring pixel $\boldsymbol{p}_k$is either grouped into the appropriate region as in seeded region-growing, or a new region is formed. One simple approach is predefining a threshold T to decide whether or not to form a new region. If the smallest difference between the neighboring pixel's intensity value and an existing regions' mean is greater than the predefined threshold, a new region is created with this pixel. This method uses the same pixel connectivity definitions as in the seeded region growing method. The $f(\quad)$ function used in this method is defined as [59]:

$$\begin{aligned} &if\ \left(J = \underset{1\le i\le n}{argmin}\, d_{ki}\right) AND \left(\underset{1\le i\le n}{min}\, d_{ki} \le T\right) AND\ (CN(\boldsymbol{p}_k, \boldsymbol{R}_i) = TRUE) \\ &then\ f(\boldsymbol{R}_J) = TRUE, \qquad \boldsymbol{R}_J \leftarrow \{\boldsymbol{R}_J, \boldsymbol{p}_k\} \\ &else\ J \leftarrow n+1, \qquad n \leftarrow J, \qquad \boldsymbol{R}_J \leftarrow \{\boldsymbol{p}_k\}, \qquad f(\boldsymbol{R}_J) = TRUE \end{aligned} \tag{7.16}$$

where $d_{ki} = \left|I(p_k) - mean\left(I(\boldsymbol{R}_i)\right)\right|$ is the intensity value difference between the neighboring pixel $\mathrm{p_k}$ and the mean of existing regions $\boldsymbol{R}_i, 1 \le i \le n$, and T is the predefined threshold. We note that the order of execution in (7.14) is very important for proper processing. In this case, the number of regions might change through the growing process (e.g., increase by one whenever a new region is formed).

New proposed region growing:

We proposed a region-growing method based on the unseeded region growing technique, with a number of important modifications. First, our method operates at a "superpixel" level, rather than at the image pixel level. By utilizing superpixels, the proposed method avoids both issues of computational expense and excessively small regions while providing quasi-homogeneous regions similar to those of other existing region-growing techniques [56-59]. However, a new definition of connectivity at the superpixel level is needed. Each region $\boldsymbol{R}_i$ is now a set of superpixels, and Condition 2 in (7.11), is modified as follows:

New Condition 2: $\boldsymbol{S}_k$ and $\boldsymbol{S}_l$ are "connected", $\forall\, \boldsymbol{S}_k, \boldsymbol{S}_l \in R_i, 1 \le i \le n$ (7.17)

where $\boldsymbol{S}_k$ and $\boldsymbol{S}_l$ are the k^{th} and s^{th} superpixel, $1 \le k \ne l \le K$, K is the total number of superpixels, and n is total number of regions. We note that each superpixel itself is a set of connected pixels as discussed in Section 2.1. The connectivity at the pixel level remains conventional (e.g., the 4-connected neighbors). Two superpixels S_k and S_l are "connected" if there exists a *sequence* of superpixels $\boldsymbol{S}_m, \dots, \boldsymbol{S}_j$ such that:

$\boldsymbol{S}_m, \dots, \boldsymbol{S}_j \in \boldsymbol{R}_i$, all superpixels in the sequence are in the set $\boldsymbol{R}_i$ (7.17a)

and every 2 superpixels that are adjacent in the sequence are connected. (7.17b)

Any two superpixels $\boldsymbol{S}_m$ and $\boldsymbol{S}_j$ are "neighbors" or "connected" if there exists a pixel $\boldsymbol{p}_m \in \boldsymbol{S}_m$ and another pixel $\boldsymbol{p}_j \in \boldsymbol{S}_j$ such that $\boldsymbol{p}_m$ and $\boldsymbol{p}_j$ are 4-connected.

A superpixel $\boldsymbol{S}_m$is said to be a "neighbor" of, or "connected" to, a region $\boldsymbol{R}_i$, $CN(\boldsymbol{S}_k, \boldsymbol{R}_i) = TRUE$, if there exists a superpixel $\boldsymbol{S}_j \in \boldsymbol{R}_i$ such that two superpixels $\boldsymbol{S}_m$ and $\boldsymbol{S}_j$ are connected.

A second modification in our proposed method is that it works for color images, rather than for gray scale images as used in the original techniques. Therefore, our similarity measurement is defined based on statistics, mainly the interquartile range, of pixel color (in **L*a*b** color space) in regions and neighboring superpixels. (Our proposed method works for gray scale images, in which the similarity measurement is based on the interquartile range of pixel intensity).

Finally a third modification in our method is that the decision of grouping an adjacent superpixel to an existing region is dynamically depended upon the statistics, or "shape and size", of this region.

Let $\boldsymbol{S}_k$ be the investigating superpixel that is a neighbor of an existed region $\boldsymbol{R}_i$. For notational simplification, let $\overline{\boldsymbol{S}}_k$ also denote the 3D vector consisting of the mean value of **L***, **a***, and **b*** of all image pixels belong to this superpixel (e.g., $\overline{\boldsymbol{S}}_k = [\bar{l}_k, \bar{a}_k, \bar{b}_k]^T$).

Note that region $\boldsymbol{R}_i, 1 \le i \le n$, is a set of superpixels: $\boldsymbol{R}_i = \{\boldsymbol{S}_m\}, 1 \le m \le N_i$; N_i is number of superpixels belong to this region R_i; and n is the number of regions (note that $\sum_{i=1}^{n} N_i = K$,

where K is the total number of superpixels in the input image). Let $\overline{\boldsymbol{R}}_i$ be the 3D vector consisting of the mean value of **L***, **a***, and **b*** of all superpixels belong to this region:

$$\overline{\boldsymbol{R}}_i = \frac{1}{N_i}\sum_{m=1}^{N_i} \overline{\boldsymbol{S}}_m \tag{7.18}$$

Assume that the superpixel S_k belongs to the region $\boldsymbol{R}_i$. We use statistical outlier detection to determine if $\boldsymbol{S}_k$ is not the outlier. Let d_{mi} be the weighted Euclidean distance between the region mean $\overline{\boldsymbol{R}}_i$ and a superpixel $\boldsymbol{S}_m$ belong to this region (vector $\overline{\boldsymbol{S}}_m$):

$$\begin{aligned} d_{mi} &= \|\overline{\boldsymbol{S}}_m - \overline{\boldsymbol{R}}_i\|_{\boldsymbol{W_3}} \\ &= \sqrt{w_l\left(\bar{l}_i - \bar{l}_m\right)^2 + w_a(\bar{a}_i - \bar{a}_m)^2 + w_b\left(\bar{b}_i - \bar{b}_m\right)^2}\,, 1 \le m \le N_i \end{aligned} \tag{7.19}$$

where $\boldsymbol{W}_3 = [w_l, w_a, w_b]^T$ is the weight vector (e.g. $\boldsymbol{W}_3 = [1,1,1]^T$ as the first three components of the weight vector $\boldsymbol{W}$ in (2.5)); $\overline{\boldsymbol{S}}_m = \left[\bar{l}_m, \bar{a}_m, \bar{b}_m\right]^T$ and $\overline{\boldsymbol{R}}_i = \left[\bar{l}_i, \bar{a}_i, \bar{b}_i\right]^T$.

For a group of N_i numbers d_{mi}, $1 \le m \le N_i$, we use the standard statistic outlier test to verify whether a number is an outlier or not. Define Q_1^i, Q_2^i, and Q_3^i respectively the first, second, and third quartiles (or equivalently, the 25%, 50%, and 75%) of this data. (Note that Q_2^i is also the median of these N_i numbers). There are 25%, 50%, and 75% of the numbers that are less than Q_1^i, Q_2^i, and Q_3^i, respectively. Let $IQR^i = Q_3^i - Q_1^i$ be the interquartile range of this data. A number d_{ki} is considered an outlier of this data if it is different from the median more than z times the interquartile range.

$$\begin{aligned} &d_{ki} \text{ is not an outlier, if } \quad Q_1^i - z\,IQR^i \le\ d_{ki} \le Q_3^i + z\,IQR^i \\ &\qquad \text{is an outlier, otherwise.} \end{aligned} \tag{7.20}$$

In this work, we use $z = 1.5$ as normal. Hence, if d_{ki} is an outlier, the superpixel $\boldsymbol{S}_k$ should not belong to the region $\boldsymbol{R}_i$. Otherwise, this superpixel S_k belongs to the region $\boldsymbol{R}_i$.

With the connectivity between superpixels, and between superpixels and regions as defined in (2.17), and the test to verify that a superpixel should belong to an existing region, the $f(\ \ )$ function used in our method is defined as:

$$
\begin{aligned}
&if\ \left(J = \underset{1 \le i \le n}{\operatorname{argmin}}\, d_{ki}\right) AND\left(Q_1^i - z\ IQR^i \le\ d_{ki} \le Q_3^i + z\ IQR^i\right)\ AND\ (CN(\boldsymbol{S}_k, \boldsymbol{R}_i) = TRUE) \\
&then\ f(\boldsymbol{R}_J) = TRUE, \qquad \boldsymbol{R}_J \leftarrow \{\boldsymbol{R}_J, \boldsymbol{S}_k\} \\
&else\ J \leftarrow n+1, \qquad n \leftarrow J, \qquad \boldsymbol{R}_J \leftarrow \{\boldsymbol{S}_k\}, \qquad f(\boldsymbol{R}_J) = TRUE
\end{aligned}
\tag{7.21}
$$

At each iteration, a current superpixel $\boldsymbol{S}_k$ is considered belonging to the "closest" existing regions that are connected to it. The "closest" measurement is in sense of the weighted Euclidean distance between this superpixel and a region mean in the 3D **L*a*b*** color space. Note that we use 3D color space instead of the 5D **L*a*b*xy** space used in the superpixel generating process (Section 7.1), because the spatial information (**xy** coordinates) is already enforced by the connectivity definition. For example, a region may consists two connected superpixels that are far away in the image plane (large difference in **xy** coordinates), but similar in color. If the investigated superpixel $\boldsymbol{S}_k$ is not an outlier of the closest region $\boldsymbol{R}_J$, based on the statistical outlier test, then it will be grouped into this region. Otherwise, a new region, which consists of this superpixel $\boldsymbol{S}_k$, is created. The process is repeated until all superpixels in the given image are visited. The algorithm is illustrated Fig. 7.6, and also as a pseudo code in Table 7.2.

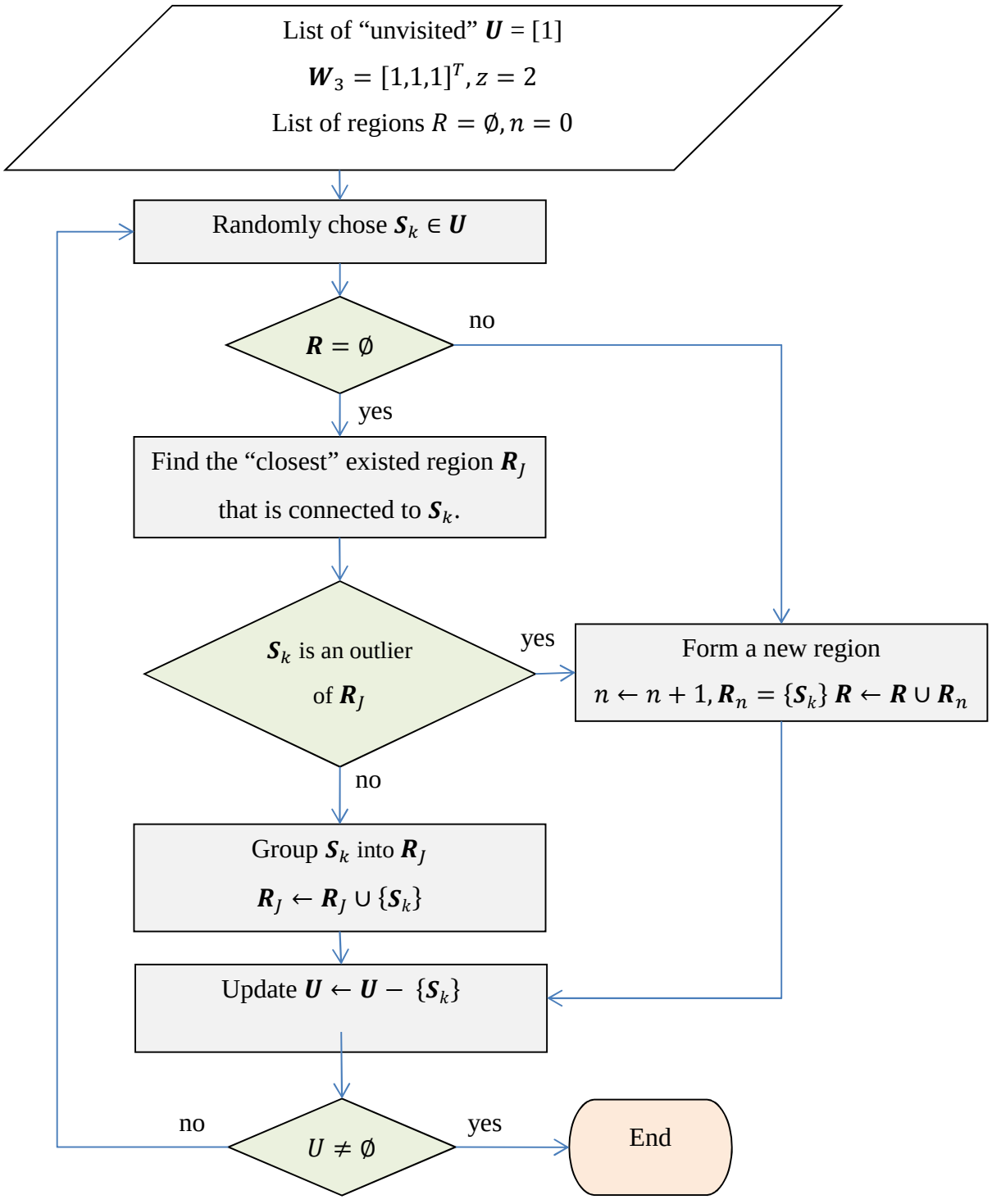


Figure 7.6: Proposed Region Growing Algorithm

Table 7.2: Proposed Region Growing Algorithm

- Given the list of unvisited superpixels $\boldsymbol{U}$ = [1] as the result of our algorithm in Table 7.1. For our work herein, z=2 and $\boldsymbol{W}_3 = [1,1,1]^T$.
- Randomly chose a superpixel $\boldsymbol{S}_k \in \boldsymbol{U}$.
- Find the "closest" existed region $\boldsymbol{R}_J$ that is connected (as defined in (7.15)) to $\boldsymbol{S}_k$ based on the weighted Euclidean distance to regions' centers as in (7.17).
- Validate that $\boldsymbol{S}_k$ is not an outlier of $\boldsymbol{R}_J$ as in (7.18) and (7.19). Include the superpixel $\boldsymbol{S}_k$ to this region $\boldsymbol{R}_J$.
- Otherwise, create a new region to contain this superpixel $\boldsymbol{S}_k$.
- Remove S_k from the list of unvisited superpixels $\boldsymbol{U} \leftarrow \boldsymbol{U} - \{\boldsymbol{S}_k\}$.
- Repeat to choose another superpixel in the list $\boldsymbol{U}$ until all superpixels have been visited (and grouped into appropriate regions).

Note that when we compute interquartile the range for a group having only one data point, for example a region $\boldsymbol{R}_J$ containing only one superpixel (e.g., a new formed region), and an investigated superpixel $\boldsymbol{S}_k$, the interquartile range is simply set equal to the distance d_{kJ}, (e.g. $Q_1^i = 0,\ Q_3^i = d_{kJ},\ IQR^i = d_{kJ}$). Therefore the condition (2.20) ($Q_1^i - z\,IQR^i \le d_{ki} \le Q_3^i + z\,IQR^i$), which equivalent to $\left(-1.5d_{kJ} \le d_{kJ} \le 2.5d_{kJ}\right)$, is always true, or d_{kJ} is always not an outlier. In other words, if the investigated superpixel $\boldsymbol{S}_k$ is closest to a neighboring existing region that is newly formed (contains only one superpixel), then $\boldsymbol{S}_k$ is always grouped into this region.

7.3 Our proposed DUHO segmentation method

Our new image segmentation algorithm, which we designate the DUHO method, illustrated in Figure 7.7, is the combination of the two algorithms discussed in previous sections, the superpixel generating algorithm in Section 7.1 and the modified region-growing algorithm in Section 7.2. First, the superpixel generating algorithm is applied to a given image to build K superpixels. Then

the new region growing algorithm iteratively groups these superpixels into appropriate regions and forms the final image segmentation result.

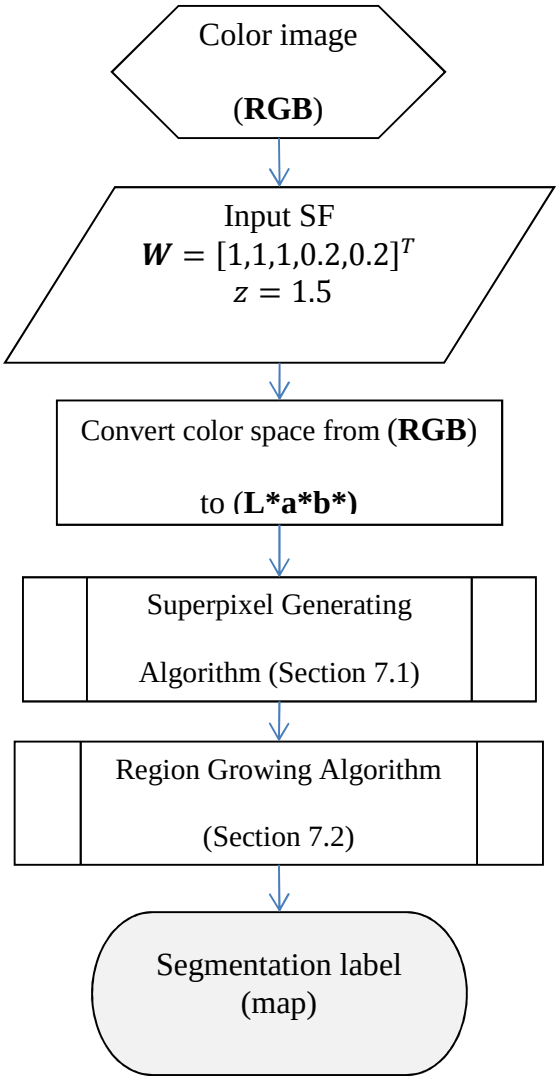


Figure 7.7: Proposed DUHO Image Segmentation Algorithm

Control Parameter:

Our superpixel generating algorithm has two control parameters, the desired number of superpixels K and the 5D weight vector $\boldsymbol{W}$. Since this is an intermediate step, and the superpixels will be grouped or merged in the next step, the final segmentation results are sensitive to the

value selected for K. However, an excessively small value of K might lead to a poor "under-segmentation" result, while an excessively large value of K results in excessive computational time and might produce undesired "over-segmentation". The weight vector $\boldsymbol{W}$ controls the effect of the properties of each pixel, e.g. intensity, color, and spatial location, and hence controls the compactness of a superpixel. In the work herein, these parameters' values are selected manually based on experiments, e.g. $\boldsymbol{W} = [1,1,1,0.2,0.2]^T$, or $w_l = w_a = w_b = 1$ and $w_x = w_y = 0.2$, $SF_{default} = \frac{N}{30\times30}$, $K_{default} = \frac{N}{SF_{default}} = 900$, N is number of pixel in the input image (in most test images $N = 481 \times 321 = 154{,}401$, hence $SF_{default} \cong 171 \cong 13 \times 13$; see Section 7.1). This roughly matches the empirical maximum perceptually meaningful CIELAB distance and offers a good balance between color similarity and spatial proximity. The final number of superpixels provides a good balance between reasonable visual effect of segmentation and practical computational time.

Our region growing has two parameters, namely the 3D weight vector $\boldsymbol{W}_3$ and the outlier range z. The weight parameter controls the emphasis of each component, intensity and color, in the (**L*a*b***) space. However, there is no specific reason to select different values of $\boldsymbol{W}_3$ from the first three components of $\boldsymbol{W}$ used in the superpixel generating algorithm. Moreover, since all data in (**L*a*b***) space is computed from (**RGB**) color space and has been normalized in the range [0,1], we advise selecting $w_l = w_a = w_b = 1$. The outlier range z affects the test to accept a superpixel belonging to (and hence to be grouped within) a region. Therefore it controls the sensitivity of segmentation results. A small value of z means that only a superpixel that has very similar color and intensity properties to those of the region's mean will be accepted to join that region, such that a large number of small regions are produced("over-segmentation"). In contrast, a large of value of z tends to produce a small number of large regions ("under-segmentation"). In this work, the value $z = 1.5$ was selected to produce a reasonable balance. In summary, only two parameters in the first step, the superpixel generating algorithm, need to be selected.

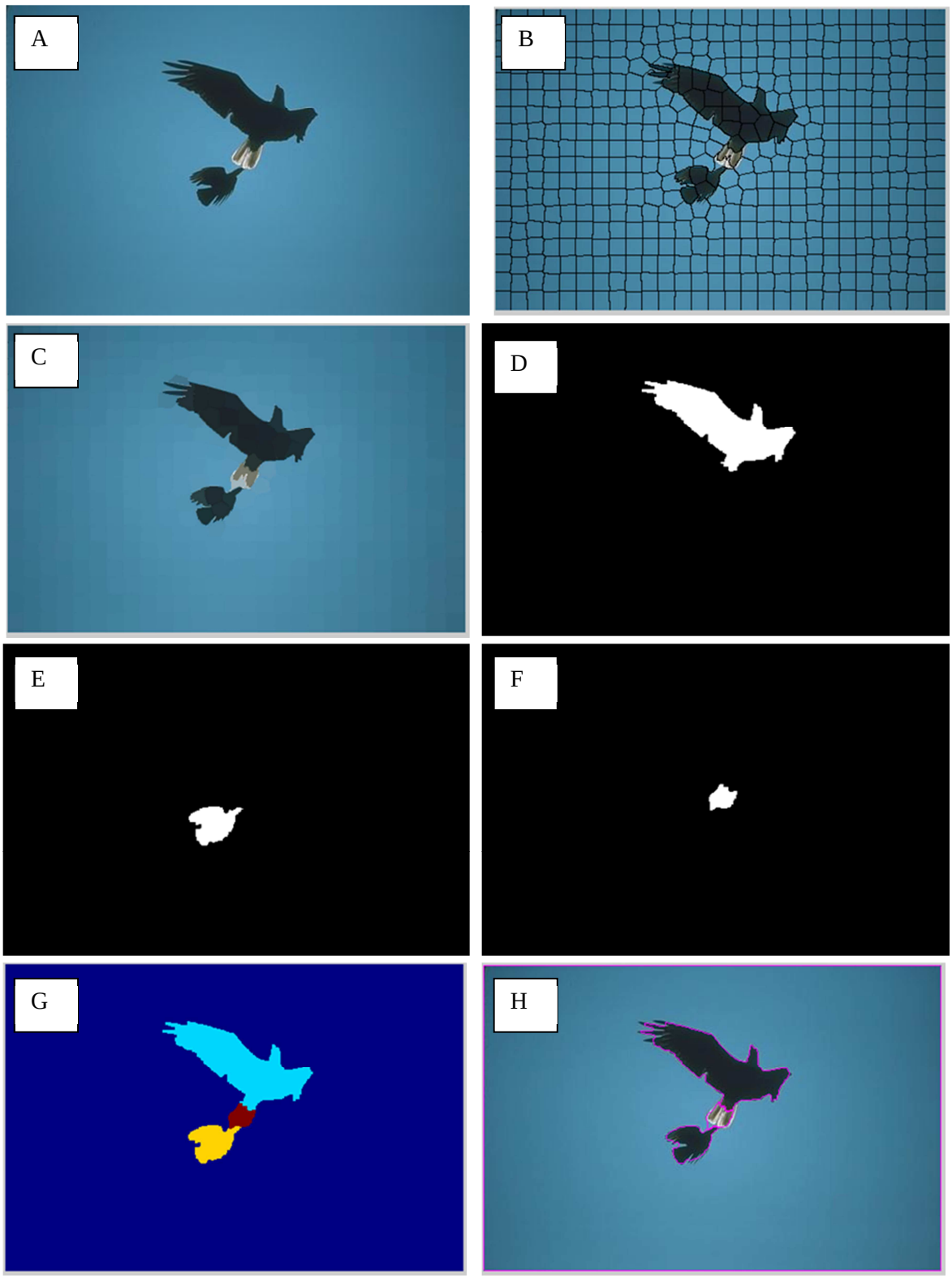


Figure 7.8: DUHO Segmentation Process (See text for detail).

Figure 7.8 illustrates our segmentation process for an image: (A) the input color image, (B) results from our superpixel generating algorithm;(C) mean value of each superpixel; (D,E,and F) three regions after applying our region growing algorithm; (G) the segmentation map, in which each region is shown in a different color; and (H) segments' boundaries (in purple) overlaid on the original image. Note how this new method preserves the spatial relationship between pixels in the image, and hence preserves the detail edges and the image spatial structure.

7.4 Complexity analysis of our DUHO segmentation algorithm

7.4.1 Complexity of our superpixel generating algorithm

Our superpixel generating algorithm, presented in Section 7.1, includes the following:

Step 1 Input SF and $\boldsymbol{W}$. Normalize all vectors in 5D space. Initialize $K = N/SF$ superpixel centers $\boldsymbol{C}_k, 1 \leq k \leq K$ at regular grid size $S = \sqrt{N/K}$

Step 2 Find the nearest superpixel to each pixel $\boldsymbol{p}_i$, $1 \leq i \leq N$

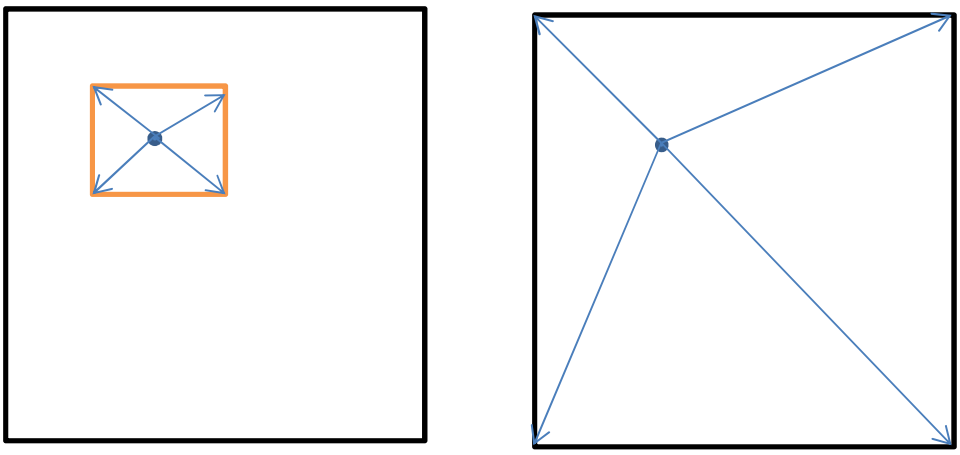
Step 3 Update new superpixel centers.
Compute the stopping criteria (number of pixels changed)

Step 4 Repeat Steps 2-3 until stopping criteria is met.

We use the standard "big O" notation for analyzing the complexity of the algorithm, characterized by computation time, in the worst case scenario and as a function of N (number of pixels of the input image) when $N \to \infty$. Therefore, higher computation times indicate higher complexity. Computational complexity for Step 1 is $T_1 = \mathrm{O}(1)$, or the computation time complexity of this step does not depend on N.

At Step 2, for each pixel, we compute its distance (Euclidean distance in 5D space) to all K superpixel centers. Therefore, the naïve implementation (such as in the original K-means algorithm) has a computational time cost $T_2 = \mathrm{O}(KN) = \mathrm{O}(N^2)$, $K = N/SF$, where SF is a constant. However, we take advantage of the fact that in our algorithm all superpixels are adjusted and compact because the spatial information (**xy**-coordinators) of pixels is restrained. A

superpixel should contain only pixels that are not very far away from its center. In other words, there is no need to compute the distance from a given pixel to every superpixels center but only to those that are in the 2S-by-2S proximity to this pixel. (Normally, there are 4 superpixels in this range). Accordingly, the computational time for Step 2 is significantly reduced: $T_2 = O(c_1 N) = O(N)$, $c_1 = 4$. Figure 7.9 illustrates that the search space of our algorithm is significantly reduced from that of the original K-means algorithm. This is the reason for the fast speed of our segmentation procedure.



(a) Our algorithm searches a space of 2S-by-2S pixels (A superpixel is roughly S-by-S pixels)

(b) K-means searches the whole image

Figure 7.9: The search space for each pixel at current step
for our superpixel generating (a) and the original K-means (b)

Computational time for the Steps 3 and 4 are $T_3 = T_4 = O(1)$.

For the worst case, the algorithm terminates only after the maximum number of iterations ($\beta = 10$) is reached. Hence, the computational time for our superpixel generating algorithm is:

$$\begin{aligned} Time_1 &= T_1 + \beta(T_2 + T_3 + T_4) \\ &= O(1) + \beta(c_1 O(N) + O(1) + O(1)) \\ &= \beta c_1 O(N) \end{aligned} \tag{7.22}$$

7.4.2 Complexity of our region growing algorithm

Our region growing algorithm, presented in Table 7.2, Section 7.2, includes following steps:

Step 1 Given U = [1], z=2 and $\boldsymbol{W}_3 = [1,1,1]^T$

Step 2 Randomly chose a superpixel $\boldsymbol{S_k} \in \boldsymbol{U}$

Step 3 Find the "closest" existed region $\boldsymbol{R_J}$ that is connected to $\boldsymbol{S_k}$.

Step 4 Validate that S_k is not an outlier of $\boldsymbol{R_J}$. Include the superpixel $\boldsymbol{S_k}$ to this region $\boldsymbol{R_J}$.
Otherwise, create a new region to contain this superpixel $\boldsymbol{S_k}$.
Remove S_k from the list of unvisited superpixels $\boldsymbol{U} \leftarrow \boldsymbol{U} - \{\boldsymbol{S_k}\}$.

Step 5 Repeat Step 2 until all superpixels have been visited.

The computational times of all steps except the Step 3 of this algorithm are independent of N: $T_1 = T_2 = T_4 = T_5 = \mathrm{O}(1)$.

In Step 3, we must compute the distance (weighted Euclidean distance in 3D space) from a given superpixel to existing regions that are connected to this superpixel. The distance computing and connectivity check task are both independent of N. In the worst case, the number of existing regions that are connected to a superpixel is n (the total number of regions in the image). Hence, the computational time for this step is $T_3 = \mathrm{O}(n^2)$.

This algorithm always terminates after K iterations (K is the number of superpixels). Therefore, the computational time for our region growing algorithm is:

$$\begin{aligned} Time_2 &= T_1 + K(T_2 + T_3 + T_4 + T_5) \\ &= \mathrm{O}(1) + K(\mathrm{O}(1) + \mathrm{O}(n^2) + \mathrm{O}(1) + \mathrm{O}(1)) \\ &= K\mathrm{O}(n^2) \end{aligned} \tag{7.23}$$

7.4.3 Complexity of our DUHO segmentation algorithm:

Our DUHO segmentation algorithm is a combination of the two above algorithms. Hence, the total computational time is:

$$\begin{aligned} Time &= Time_1 + Time_2 \\ &= \beta c_1 \mathrm{O}(N) + K\mathrm{O}(n^2) \end{aligned} \tag{7.24a}$$

However, since $n \ll N$, the term $\mathrm{O}(n^2)$ is very small, and since β, c_1 and K are constant parameter given by users, we can simplify (7.24a) to read

$$Time = \mathrm{O}(N) \tag{7.24b}$$

So, our DUHO segmentation computation time varies linearly with the input image size N.

7.5 Objective function and convergence of DUHO segmentation algorithm

7.5.1 Objective function of our superpixel generating algorithm

For our superpixel generating algorithm, which is essentially based on K-means clustering with modifications in similarity measurement and a time-reducing implementation, we consider an objective function similar to the original K-means algorithm. The superpixel generating problem can be viewed as finding the "best" way (according to the objective function defined later) to divide a finite set of N pixels (represented by a 5 dimensional vector) $\boldsymbol{p}_i, 1 \leq i \leq N$ into K disjoint superpixels $\boldsymbol{S}_k, 1 \leq k \leq K$ among all possible way of distributing, in which K is given. Our superpixel generating algorithm can be mathematically presented as:

Input: Finite set $\boldsymbol{I} = \{\boldsymbol{p}_i\} \subset \mathbb{R}^5, |\boldsymbol{I}| = N$; integer K (7.25)

Output: Finite set $\boldsymbol{S} = \{\boldsymbol{S}_k\}$ such as $|\boldsymbol{S}| = K, \bigcup_{k=1}^{K} \boldsymbol{S}_k = \boldsymbol{I}$ and $\boldsymbol{S}_k \cap \boldsymbol{S}_{m \neq k} = \emptyset$ (7.26)

Goal: Minimize $Cost(\boldsymbol{S}) = \sum_{k=1}^{K} \sum_{\boldsymbol{p}_i \in \boldsymbol{S}_k} \|\boldsymbol{p}_i - \overline{\boldsymbol{S}}_k\|^2$ (7.27)

where $|.|$ is the cardinal (or size) of a set, and other notations as introduced earlier.

It has been proved that during K-means iterations, the cost monotonically decreases with each iteration [51]. Hence, this would also hold for our superpixel algorithm given in Section 7.1. Therefore our superpixel generating algorithm converges to a local minimum of its objective function $Cost(\boldsymbol{S})$ in (7.27).

7.5.2 Objective function of our region growing algorithm

In similar fashion, the segmentation problem can be viewed as finding the "best" way (according to some objective function, defined later) to divide a finite set of K superpixels $\boldsymbol{S}_k, 1 \leq k \leq K$ into n disjoint regions $\boldsymbol{R}_j, 1 \leq j \leq n$ among all possible ways of distributing. Note that n being unknown presents a much more difficult problem compared with the problem in Section 7.5.1, in

which the number of groups is given. Our region growing algorithm can be mathematically presented as:

Input: Finite set $\boldsymbol{I} = \{\boldsymbol{S}_k\} \subset \mathbb{R}^5, |\boldsymbol{I}| = K$ (7.28)

Output: Finite set $\boldsymbol{R} = \{\boldsymbol{R}_j, 1 \leq j \leq n\}$ such as $\bigcup_{j=1}^{n} \boldsymbol{R}_j = \boldsymbol{I}$ and $\boldsymbol{R}_j \cap \boldsymbol{R}_{m \neq j} = \emptyset$ (7.29)

Goal: Minimize $Cost(\boldsymbol{R}) = Cost_1(\boldsymbol{R}) + Cost_2(\boldsymbol{R})$ (7.30)

We propose the objective function consist of two parts. The first part, $Cost_1(\boldsymbol{R})$, is to minimize the variation within each region or the intra-region relationship, which serves the same purpose as in K-means algorithm, given by:

$$Cost_1(\boldsymbol{R}) = \sum_{j=1}^{n} \sum_{\boldsymbol{S}_k \in \boldsymbol{R}_j} \left\| \boldsymbol{S}_k - \boldsymbol{z}_j \right\|^2 \tag{7.31}$$

where $\boldsymbol{z}_j$ is the representative vector of the region $\boldsymbol{R}_j$. The second part of the objective function, $Cost_2(\boldsymbol{R})$, takes into account the inter-region relationship between regions that are neighbors, given by:

$$Cost_2(\boldsymbol{R}) = \sum_{j=1}^{n} \left(\sum_{CN(\boldsymbol{R}_j, \boldsymbol{R}_m) = TRUE} \left\| \bar{\boldsymbol{R}}_j - \bar{\boldsymbol{R}}_m \right\|^2 \right) \tag{7.32}$$

where $\bar{\boldsymbol{R}}_j$ is the mean of the region $\boldsymbol{R}_j$ as defined in (7.18). $CN(\boldsymbol{R}_j, \boldsymbol{R}_m) = TRUE$ only if the regions $\boldsymbol{R}_j$ and $\boldsymbol{R}_m$ are connected.

Combining (7.30), (7.31), and (7.32) gives the objective function of our region growing algorithm as:

$$Cost(\boldsymbol{R}) = \left(\sum_{j=1}^{n} \sum_{\boldsymbol{S}_k \in \boldsymbol{R}_j} \left\| \boldsymbol{S}_k - \boldsymbol{z}_j \right\|^2 \right) + \sum_{j=1}^{n} \left(\sum_{CN(\boldsymbol{R}_j, \boldsymbol{R}_m) = TRUE} \left\| \bar{\boldsymbol{R}}_j - \bar{\boldsymbol{R}}_m \right\|^2 \right) \tag{7.33}$$

7.5.3 Discussion of proof of convergence:

We would like to prove the convergence of our region growing algorithm presented in Section 7.2 by showing that during the iteration in of the algorithm, the cost monotonically decreases with each iteration. However, because of the complexity of the problem, we can only analytically prove that the first part, $Cost_1(\boldsymbol{R})$, monotonically decreases with each iteration in the algorithm (i.e., based on k-means proved convergence). However, for $Cost_2(\boldsymbol{R})$, note that even for a fix number of subsets n, there are enormously large combinations of ways to distribute a set of K elements into n subsets, ($K > n$). For example with $K = 100$ and $n = 2$, the task is to divide 100 elements into 2 subsets. There are $\binom{1}{100} = 100$ ways to distribute 1 element into the first subset and 99 elements into the second subset. (The "binomial coefficient" notation $\binom{n}{K}$, often read as "chose n from K", can be computed as: $\binom{n}{K} = \frac{K!}{n!(K-n)!}$, where $n! = n \times (n-1) \times \ldots \times 2 \times 1$ denotes the factorial of n, and $0! = 1$). There are $\binom{2}{100} = 4950$ ways to distribute into 2 subsets of 2 and 98 elements, respectively. There are $\binom{3}{100} = 16170$ ways to distribute into 2 subsets of 3 and 97 elements respectively, and so on. Hence, even if $n = 2$ is given (and small), the total number of ways to distribute K elements into n subsets is very large. And this number of ways grows exponentially with increasing values of n.

If n is unknown, this process must be repeated for every possible value of $n, 1 \leq n \leq K$. In this general case, the total number of ways to distribute K elements into n subsets is exceedingly large, so that it is intractable for a brute-force approach to find the global optimum of $Cost_2(\boldsymbol{R})$ among all possibilities.

Another difficulty in proof of convergence is that the "connectivity" condition in $cost_2(\boldsymbol{R})$ is difficult to represent mathematically. At each iteration of our region growing algorithm, a random

superpixel is considered either to be grouped into the (closest) existing region or to form a new region. Hence, it appears analytically intractable to represent the connectivity between regions.

***Lemma 7.1*: Optimal solution for $Cost_1(\boldsymbol{R})$ for a single region ($\boldsymbol{n = 1}$)**

The optimal solution for a single region $\boldsymbol{R}$ is when the representative $\boldsymbol{z}$ is the same as the mean $\overline{\boldsymbol{R}}$ of this region. Then we have:

$$Cost_1(\boldsymbol{R}; \boldsymbol{z}) = \sum_{\boldsymbol{S}_k \in \boldsymbol{R}} \|\boldsymbol{S}_k - \boldsymbol{z}\|^2 \rightarrow min \quad when\ \boldsymbol{z} = \boldsymbol{R} \tag{7.34}$$

The new notation $Cost_1(\boldsymbol{R}; \boldsymbol{z})$ is used to emphasize that vector(s) after the semi-colon is(are) the representative(s) to calculate the cost.

In addition, if $\boldsymbol{z} \neq \boldsymbol{R}$ is chosen, then the cost will be increased by:

$$cost_1(\boldsymbol{R}; \boldsymbol{z}) = cost_1(\boldsymbol{R}; \overline{\boldsymbol{R}}) + |\boldsymbol{R}|(\| \boldsymbol{z} - \overline{\boldsymbol{R}}\|^2) \tag{7.35}$$

The proof of Lemma 1 is quite straight forward and can be found in [51].

***Lemma 7.2:* Part of the objective function for multiple regions**

The $Cost_1(\boldsymbol{R})$ monotonically decreases during the course of our region growing algorithm.

Proof:

Let $\boldsymbol{R}_j^{(t)}$and $\overline{\boldsymbol{R}}_j^{(t)}$, $1 \leq j \leq n$, be, respectively, a region (a set of superpixels) and its center (a 3D vector mean) at the start of the t^{th} iteration of our algorithm. At first, our algorithm assigns each superpixel to its closest center. Herein, for simplicity, we ignore the connectivity and outlier test between the current superpixel and the region into which it will be grouped. Assume that a superpixel is added into the $\boldsymbol{R}_j^{(t)}$ region, such that this region will change to $\boldsymbol{R}_j^{(t+1)}$, but its representative is still $\overline{\boldsymbol{R}}_j^{(t)}$ and is not its center. Therefore we have:

$$Cost_1\left(\boldsymbol{R}_j^{(t+1)}; \overline{\boldsymbol{R}}_j^{(t)}, 1 \leq j \leq n\right) \leq Cost_1\left(\boldsymbol{R}_j^{(t)}; \overline{\boldsymbol{R}}_j^{(t)}, 1 \leq j \leq n\right) \tag{7.36}$$

Next, each region center is updated to the current region mean denoted as $\boldsymbol{R}_j^{(t+1)}$, such that by Lemma 7.1 we have:

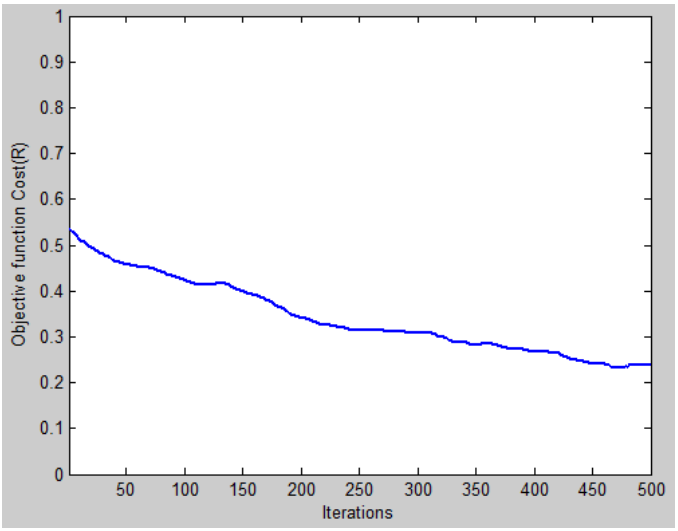
$$Cost_1\left(\boldsymbol{R}_j^{(t+1)}; \overline{\boldsymbol{R}}_j^{(t+1)}, 1 \leq j \leq n\right) \leq Cost_1\left(\boldsymbol{R}_j^{(t+1)}; \overline{\boldsymbol{R}}_j^{(t)}, 1 \leq j \leq n\right) \tag{7.37}$$

Combine (7.36) and (7.37) to obtain:

$$Cost_1\left(\boldsymbol{R}_j^{(t+1)}; \bar{\boldsymbol{R}}_j^{(t+1)}, 1 \le j \le n\right) \le Cost_1\left(\boldsymbol{R}_j^{(t)}; \bar{\boldsymbol{R}}_j^{(t)}, 1 \le j \le n\right)$$
or in shortened notation: $Cost_1\left(\boldsymbol{R}^{(t+1)}\right) \le Cost_1\left(\boldsymbol{R}^{(t)}\right)$ (7.38)

Accordingly, we proved that the first part of the objective function $Cost_1(\boldsymbol{R})$ monotonically decreases with each iteration during the course of our region growing algorithm. As mentioned above, due to its complexity, we were unable to prove that the second part of the objective function $Cost_2(\boldsymbol{R})$ also monotonically decreases during the course of our region growing algorithm. Such proof, if it exists, is left for future work.

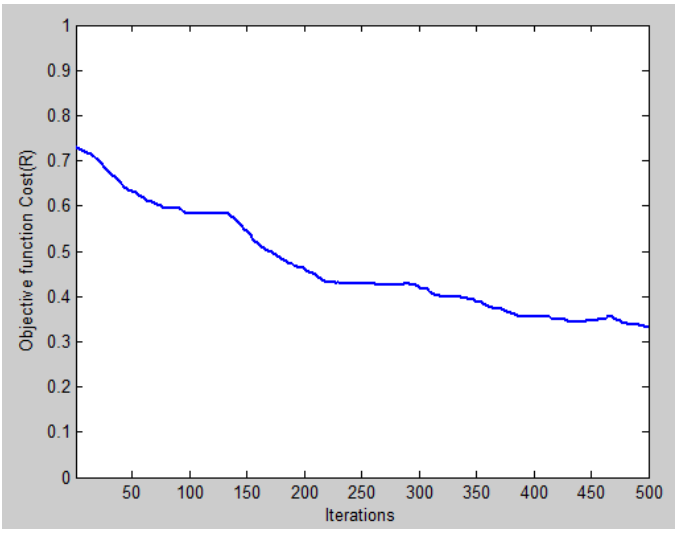
Figure 7.10(b) shows an example of the objective function $Cost(\boldsymbol{R})$ as in (7.30) of a single run our DUHO segmentation on the image in Fig. 7.10(a). Since there are 500 superpixels in the image, the DUHO algorithm terminates after 500 iterations.



(a) (b)

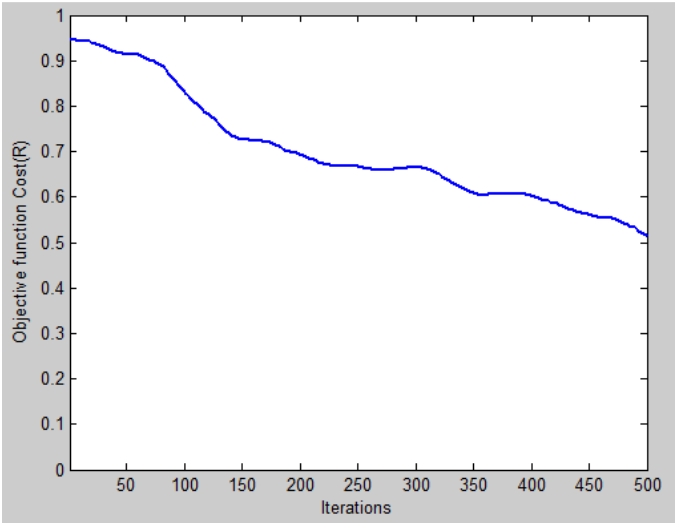
Figure 7.10: Objective function in (b) $Cost(\boldsymbol{R})$ of a single run our DUHO segmentation for the image in (a)

Four other examples of convergence are shown in Figure 7.11-7.14. During the course of our studies with 300 original different image, we found that $Cost(\boldsymbol{R})$ always decreased with iterations up to the maximum of *K* iterations.



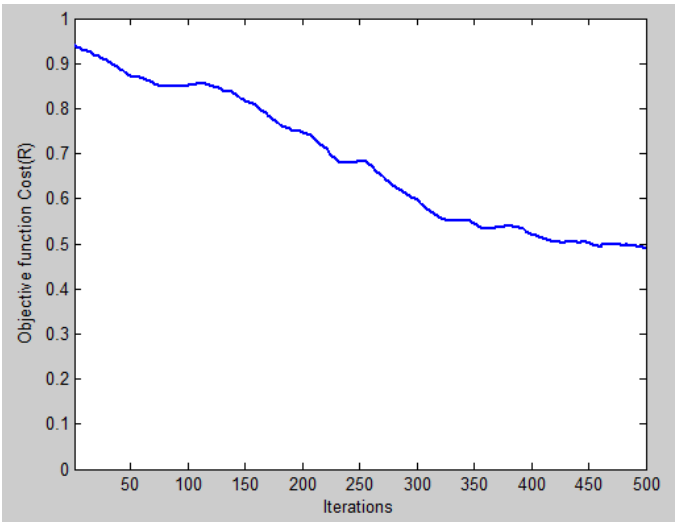
(a) (b)

Figure 7.11: Objective function in (b) $Cost(\boldsymbol{R})$ of a single run our DUHO segmentation for the image in (a)



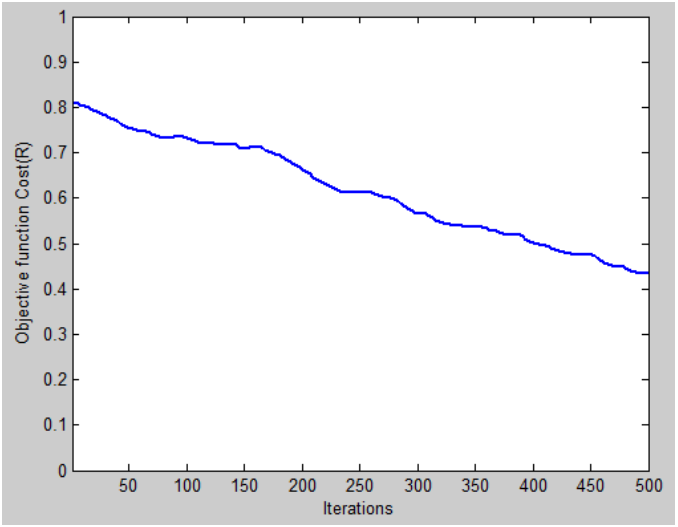
(a) (b)

Figure 7.12: Objective function in (b) $Cost(\boldsymbol{R})$ of a single run our DUHO segmentation for the image in (a)



(a) (b)

Figure 7.13: Objective function in (b) $Cost(\boldsymbol{R})$ of a single run our DUHO segmentation for the image in (a)



(a) (b)

Figure 7.14: Objective function in (b) $Cost(\boldsymbol{R})$ of a single run our DUHO segmentation for the image in (a)

Chapter 8

RESULTS AND EVALUATIONS

In this chapter, we present segmentation results of our new DUHO method discussed in Chapter 7 on a set of a large variety of natural scene images in color, which is published and available in the literature. A number of evaluation metrics for segmentation are discussed and we propose a framework to select the best metric. We also compare the results from our DUHO method with other state-of-art segmentation techniques.

8.1 Dataset

The dataset used in our test is the public Berkeley Segmentation Dataset and Benchmark (BSDB) [68]. It consists of 300 color images of natural scenes, some of which are shown in Figure 8.1. Note that some images contain one or two "stand-out" objects that are fairly easily detected from the background, such as images with name/ID number: 8068, 106025 and 135069 (see Fig.8.1). Other images have multiple objects with many details, such that different human observers might segment them differently. The dataset in [68] also contains hand-implemented segmentations for each image as shown in Figure 8.2.

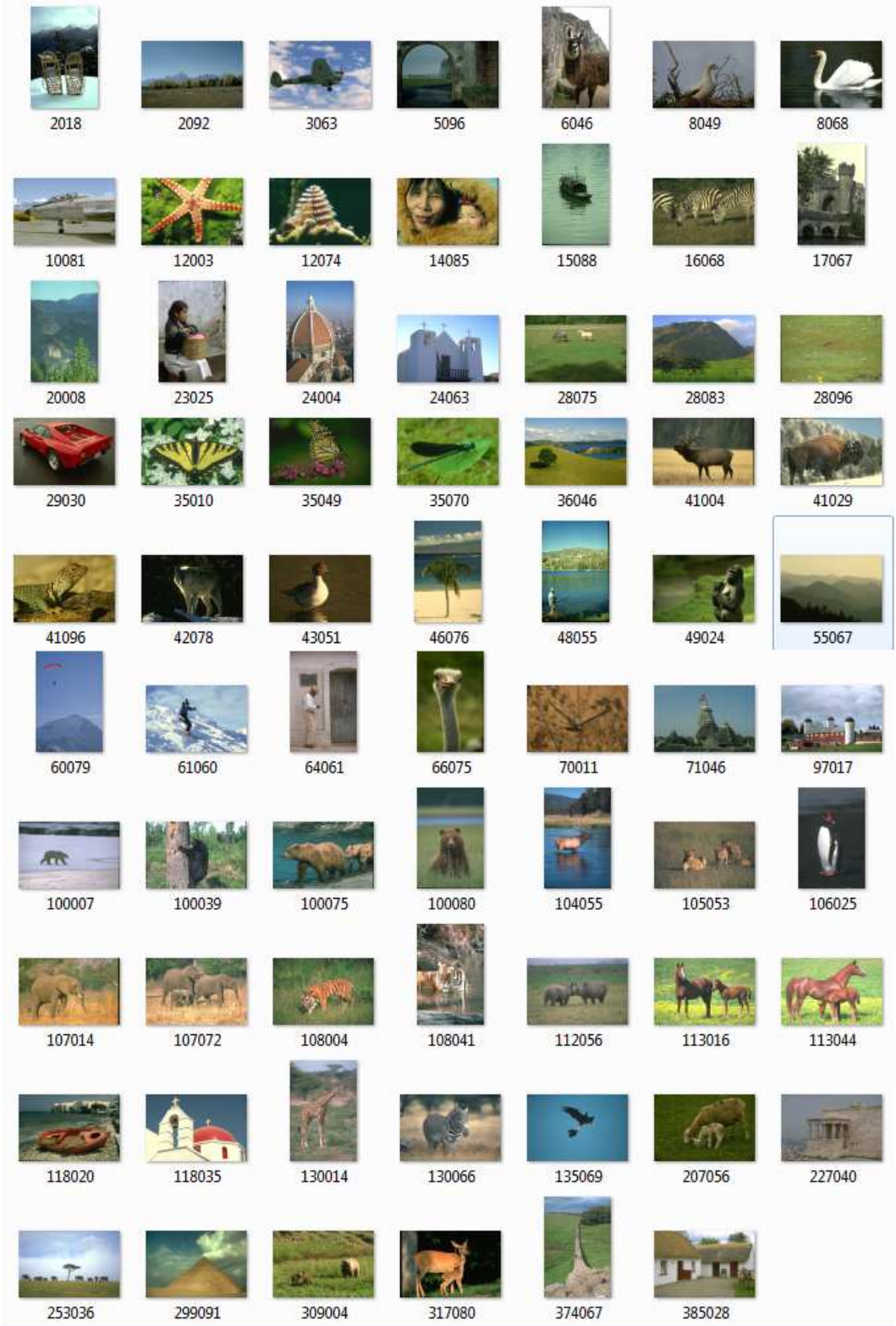


Figure 8.1: Some dataset's images from [68]

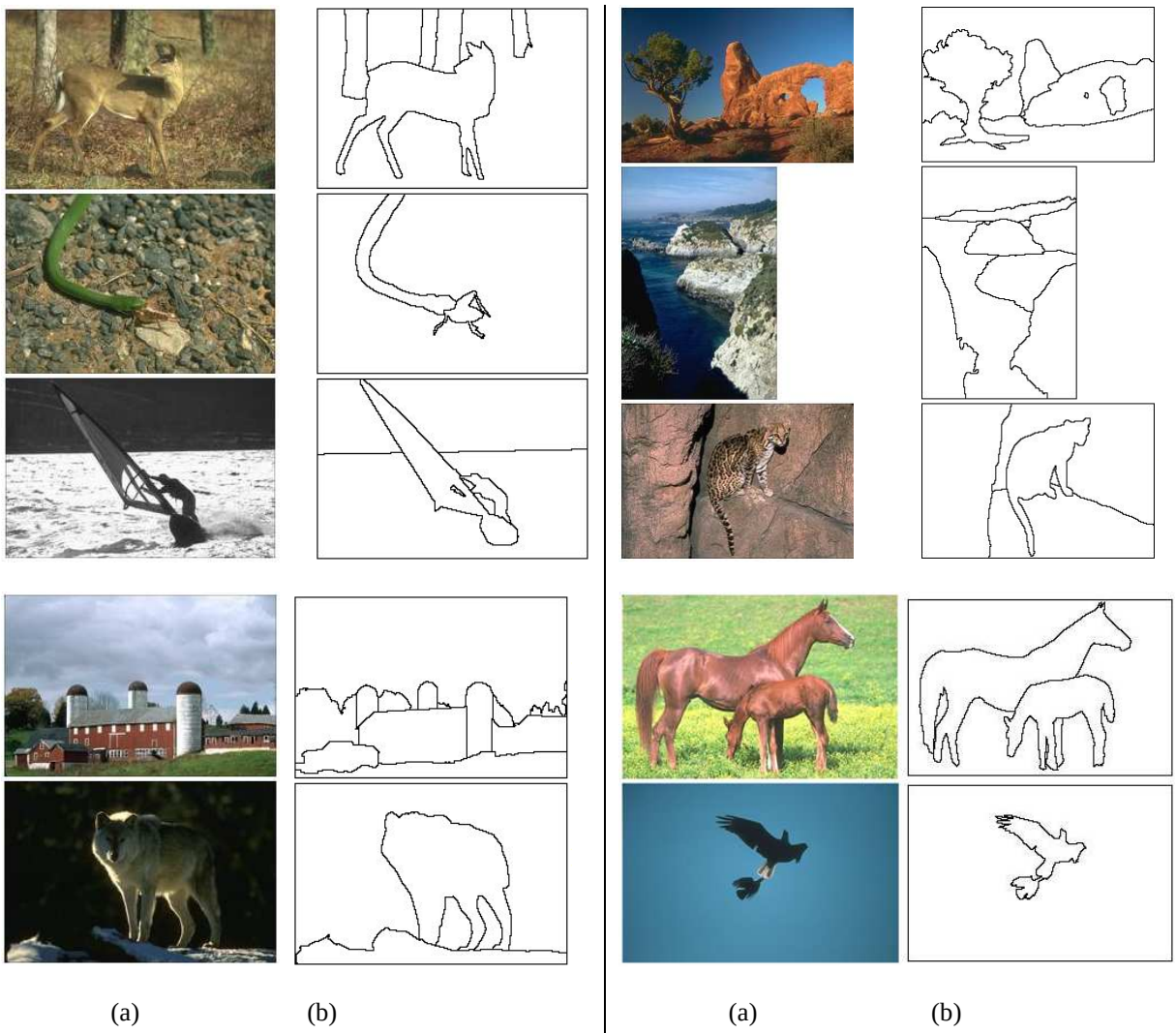


Figure 8.2: Some pairs of original image (a) and manual-implemented segmentation (b), which served as ground truths

8.2 Results of our DUHO algorithm

Sample results of our DUHO segmentation algorithm, discussed in Chapter 7, are shown in Figure 8.3. The first column (a) of Fig.8.3 contains the original images, while the second column (b) shows segmentation results in which the boundaries of segments identified by DUHO in purple are overlaid onto the original image, and the last column (c) shows the same segmentation results in which each DUHO segment is presented in a distinct color.

(a) (b) (c)

Figure 8.3: Our DUHO segmentation results on some images

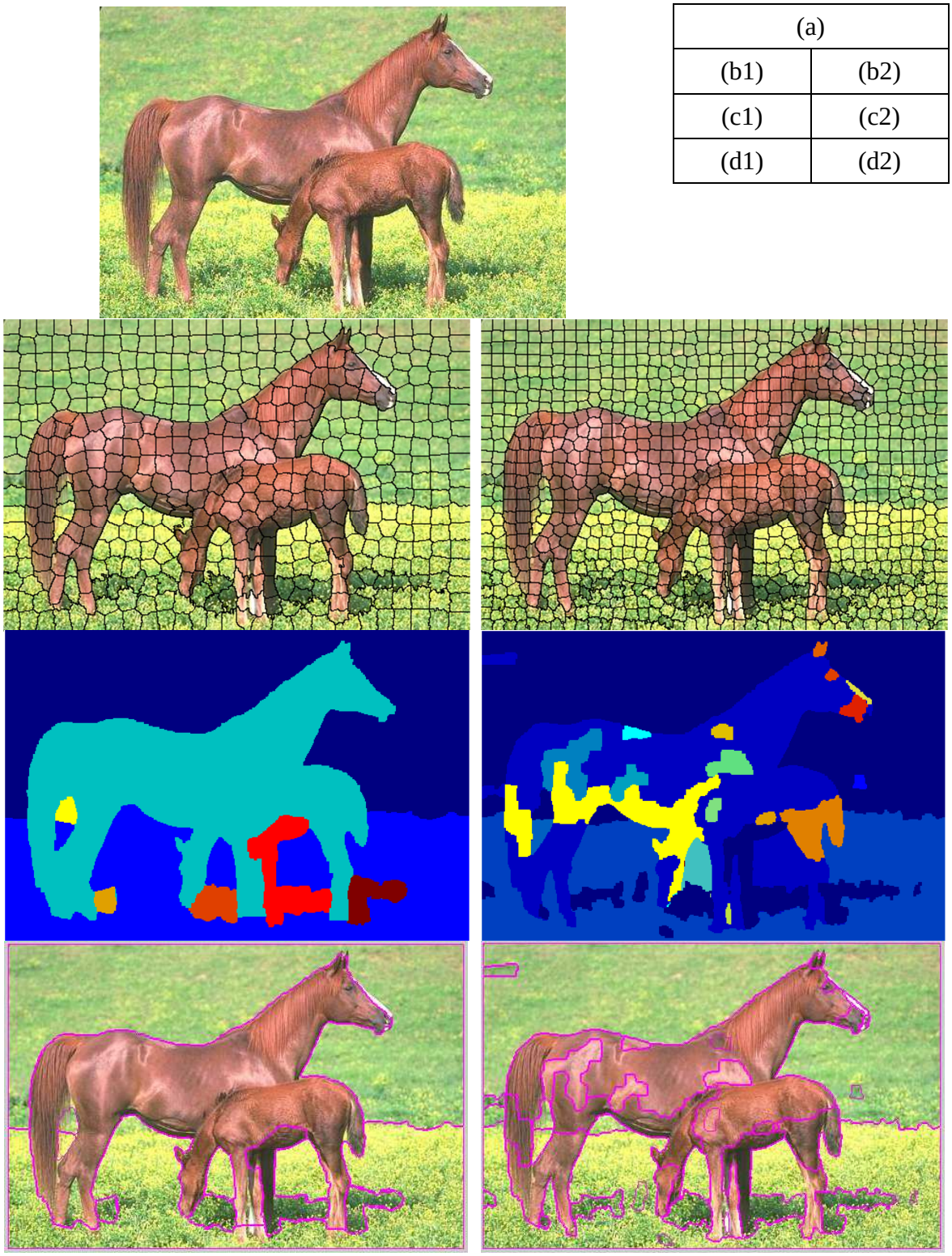
Note that our DUHO segmentation algorithm preserves well the detailed edges/boundaries of "true" segments or objects and the image spatial structure. Compared with human segmentation, our algorithm normally produces smaller segments. In our algorithm, different segments differ from each other in color and location. Humans often take advantage of prior knowledge to combine into a larger segment some different regions (that are connected but distinguishable in color), about which he or she has prior knowledge that these should belong to the same object. This type of information must be learned, and is beyond the information contained in a single image. Also, human observers often easily ignore noise, and focus only on "salient" objects in a given image, despite the fact that detecting objects from the background and detecting salient object are difficult tasks themselves.

As indicated in Section 8.3, the parameter values we used for the above results are: $\boldsymbol{W} = [1,1,1,0.2,0.2]^T$ (or $w_l = w_a = w_b = 1$ and $w_x = w_y = 0.2$), $SF_{default} = \frac{N}{30\times30}$, $K_{default} = \frac{N}{SF_{default}} = 900$, where N is number of pixels in the input image. For example, a given image of size 481-by-381 pixels has $N = 481 \times 321 = 154{,}401$, such that $SF_{default} \cong 171 \cong 13 \times 13$.

8.3 Effect of the Control Parameters:

Figure 8.4 illustrates the results of our proposed segmentation method applied to an original input image (a). Fig.8.4 (b1 and b2) shows the superpixel generating results with two different parameter choices: K = 500, SF = 400= 20x20 (b1); and K =1000, SF = 200 ≅ 14x14 (b2). Figs c1 and d1 correspond to segmentation in Fig a1, and Figs c2 and d2 correspond to segmentation in Fig a2. Fig.8.4 (c1 and c2) show the segmentation results from these two parameter sets, in which each segment is presented in a distinct color. Fig.8.4 (d1 and d2) show the same segmentation results in which the boundaries of segments in purple are overlaid onto the original image. As we expected, when the smallest feature size value (SF) is larger (equivalent to smaller number of superpixels K), our segmentation algorithm produces a small number of large

segments/regions. In other words, the algorithm returns coarser segmentation in which any feature that is smaller than SF will be smoothed out, such that some details of the original image might be ignored (see Fig.3.4.c1). In contrast, with a smaller value of SF (larger value of K), our algorithm produces a large number of small segments/regions and more detail from the original image is retained (see Fig.8.4.c2). In short, the control parameter SF is a means for users to control the coarseness or fineness of the segmentation results. The "optimal" value of SF is dependent on the image features, together with the purpose or application of the segmentation process. With a given input image to be segmented, a user can select the appropriate value of SF based on the smallest feature size (e.g. the number of pixels) in this image that he or she would like to capture. If the segmentation results are too fine or over-segmented (e.g. many smaller segments than expected), one can increase SF and re-run the algorithm, and vice versa.



(a)	
(b1)	(b2)
(c1)	(c2)
(d1)	(d2)

Figure 8.4: Results of our DUHO segmentation method with $\boldsymbol{SF}$ =20x20 (left) and $\boldsymbol{SF}$ =40x40(right). See text for more detail.

8.4 Selecting the best unsupervised metric

In this section, we present six unsupervised metrics in the literature that are commonly used to evaluate segmentation results. Then we propose a framework to find the best comparison metric in the sense that this metric is the most consistent with the ground-truth provided by manual segmentation and, at the same time, is the most sensitive to random segmentation results. We believe a "good" metric should produce a high score on the ground-truth segmentation as well as produce a low score on random segmentation. As discussed earlier, the fundamental difficulty with evaluation of segmentation is that there is no objective, clear definition of good or bad segmentation. Moreover, different observers often do not agree on how to segment the same given image. The issue of variation in the scales of error in ground-truths from human segmentation results will be discussed in the Section 8.5.

8.4.1 Discrepancy evaluation technique (D −metric)

This technique, introduced by Weska and Rosenfeld [70], is a simple technique based on the discrepancy measure (D-metric) between the original and the segmented images. Precisely, discrepancy is computed by the sum of the squares of specified differences between the original image and the segmented image. This measure D is given by:

$$D = \sum_{i=1}^{Ih} \sum_{j=1}^{Iw} \big(G(i,j) - L(i,j)\big)^2 \tag{8.1}$$

where I_h and I_w are, respectively, the height and width of the image; $G(i,j)$ and $L(i,j)$ are the grayscale of the pixel (i,j) of the input image and the segmented image, respectively. Note that in the segmented image, pixels of the same segment have the same value (grayscale or color) that is the average of pixel values (grayscale or color) that belongs to this segment in the original image. In other words, the D-metric is related to the total variation of grayscale in the original image corresponding to all segments. For a good segmentation result, the metric D should be close to zero.

8.4.2 The intra/inter-region visual error ($E-$ metric)

An unsupervised evaluation technique based on the visible color difference [71] is employed to evaluate image segmentation algorithms. Define "intra-region visual error" E_{intra} as:

$$E_{intra} = \frac{1}{N}\sum_{k=1}^{n} \mu(e_k^2 - th) \tag{8.2}$$

where $\mu(a)$ is a step function, given by

$$\mu(a) = \begin{cases} 1 & a > 0 \\ 0 & otherwise \end{cases} \tag{8.3}$$

and e_k^2 is the square of the color error in the R_k image region computed in **L*a*b** color space ($1 \le i \le n$; where n is number of segments), given by

$$e_k^2 = \sum_{p \in R_k} \|\boldsymbol{p} - \overline{\boldsymbol{R}_k}\|^2 \tag{8.4}$$

where $\boldsymbol{p} = [l, a, b]^T$ is the 3D vector corresponding to the pixel $\boldsymbol{p}$; $\overline{\boldsymbol{R}_k} = \left[\bar{l}_k, \bar{a}_k, \bar{b}_k\right]^T$ is the mean vector of the region $\boldsymbol{R}_k$; $\|\quad\|$ is the standard Euclidean norm; N is number of pixels in the input image; th denotes the threshold for visible color difference, with $th = 0.36$ according to [62].

The intra-region visual error is designed to measure the visible color difference within the segmented regions. This measure can be used to estimate the degree of under-segmentation. Intuitively, a properly segmented region should contain as few visible color errors as possible. In other words, the smaller the value of E_{intra}, the better is the segmentation.

On the other hand, another measurement named inter-region visual error is designed to measure the invisible color difference between every adjacent pair of segmented regions. This measure can be used to estimate the degree of over-segmentation. Define "inter-region visual error" E_{inter} as:

$$E_{inter} = \frac{1}{N}\sum_{k=1}^{n}\sum_{j=1,j\neq k}^{n}\frac{L_{kj}}{L_k L_j}\,\mu\left(th - \left\|\overline{\boldsymbol{R}_k} - \overline{\boldsymbol{R}_j}\right\|^2\right) \tag{8.5}$$

where L_k and L_j are the boundary length (in pixels) of regions $\boldsymbol{R}_k$ and $\boldsymbol{R}_j$, respectively. L_{kj} is the "joined length" (or number of pixels in the "shared" boundaries) between the image regions $\boldsymbol{R}_k$ and $\boldsymbol{R}_j$. $L_{ij} = 0$ if $\boldsymbol{R}_k$ and $\boldsymbol{R}_j$ are disjoined. Given a segmentation result, we take into account these boundary pixels with "invisible" color difference (no difference in color) across the boundary. Intuitively, these pixels should not be treated as boundaries. Hence, the smaller the value of $\mathrm{E_{inter}}$, the better is the segmentation.

Based on these two measures, a score or metric E to measure how good is a given segmentation, may be defined by:

$$E \quad = \frac{1}{2}(E_{intra} + E_{inter}) \tag{8.6}$$

Note that, for a segmented image, a large value of intra-region visual error means numerous pixels may be mistakenly merged, such that this image could have been under-segmented. On the other hand, a large value of inter-region visual error means numerous boundary pixels may be mistakenly generated, such that the image could have been over-segmented. Moreover, there is a reciprocal relationship between intra-region error and inter-region error. As we adjust the controlling parameters of a segmentation algorithm to merge more regions together, the inter-region error decreases, while the intra-region error increases. On the contrary, as we segment an image into more regions, the intra-region error decreases while the inter-region error increases. Also note that all pixel color values are normalized to range of [0,1]. Normally n (number of regions) $\ll N$(number of image pixels), such that E_{intra}, E_{inter} and E will all lie in the range [0,1]. For a good segmentation result, the metric E should be close to zero.

8.4.3 Average squared color error ($Q-$ metric)

This metric is empirically defined by Borsotti at el. [72] as:

$$Q = \frac{1}{kN}\sqrt{n}\sum_{k=1}^{n}\left[\frac{e_{\boldsymbol{R}_k}^2}{1+\log A_k} + \left(\frac{n_k}{A_k}\right)^2\right] \tag{8.7}$$

where $k = 10^4$is an empirical number and a normalization factor that takes the size of the image into account, N is the total number of pixels in the image, n is the number of segments, e_{Rk}^2 is the square of the color error in the segment $\boldsymbol{R}_k$ as in (8.4), A_k is the area of the segment $\boldsymbol{R}_k$, and n_k is the number of segments that have the area in the range from $0.98A_k$ to $1.02A_k$.

Note that the $\sqrt{n}$ term penalizes segmentation results having too many regions; the $e_{\boldsymbol{R}_k}^2$ term penalizes results having non-homogeneous regions. The square of the color error will be significantly higher for a large region, such that the adjusted term $(1 + \log A_k)$ is applied. Experiments show that the number of large regions that have a similar area is small, while the number of small regions that have a similar area may be large [42]. Therefore, the Q measure also penalizes the segmentation result having too many small regions that are similar in size. For a good segmentation result, the metric Q should be close to zero.

8.4.4 Entropy based metric ($H-$ metric)

Zhang et al. [73] proposed another unsupervised evaluation metric based on the "region" entropy and the "layout" entropy. Define the entropy for each segment $\boldsymbol{R}_k$ by:

$$H(\boldsymbol{R}_k) = -\sum_{m\in V_k}\frac{L_k(m)}{A_k}\log\frac{L_k(m)}{A_k} \tag{8.8}$$

where V_k is the set of all possible grayscale values of pixels in the region $\boldsymbol{R}_k$ of the original image, $L_k(m)$ is the number of pixels in this region $\boldsymbol{R}_k$ of the original image that have the grayscale value of m.

Define the region entropy H_r of entire image as the sum of entropy across all regions weighted by their areas, given by:

$$H_r = \sqrt{n}\sum_{k=1}^{n}\frac{A_k}{N}\,H(\boldsymbol{R}_k) \tag{8.9}$$

where n, N, A_k, and $H(\boldsymbol{R}_k)$ are defined as before. Note that N is the total number of pixels of the image or the "area" of the image. The first term $\sqrt{n}$ penalizes segmentation result having too many regions. Now define the layout entropy H_l by:

$$H_l = -\sum_{k=1}^{n} \frac{A_k}{N} \log \frac{A_k}{N} \tag{8.10}$$

The H-metric measuring the effectiveness of a segmentation method is the addition of the two entropies, given by:

$$H = H_r + H_l \tag{8.11}$$

For a given dataset, the H-metric can be normalized to a range [0,1], in which a small value of H (close to zero) indicates good segmentation.

8.4.5 Spatial color contrast along the boundaries of segments ($C-$ metric)

This metric introduced in [74] considers the internal and external contrast of the neighbors of each pixel in all segments. Define $W(\boldsymbol{p})$ as the set of pixels that are the 8 neighbors of the pixel $\boldsymbol{p}$. For each segment $\boldsymbol{R}_k$, define the internal contrast I_k and external contrast $E_k$as:

$$I_k = \frac{1}{A_k} \sum_{\boldsymbol{p} \in \boldsymbol{R}_k} \max(\|\boldsymbol{p} - \boldsymbol{q}\|, \forall \boldsymbol{q} \in W(\boldsymbol{p}) \cap \boldsymbol{R}_k) \tag{8.12}$$

$$E_k = \frac{1}{l_k} \sum_{\boldsymbol{p} \in \boldsymbol{R}_k} \max\left(\|\boldsymbol{p} - \boldsymbol{q}\|, \forall \left(\boldsymbol{q} \in W(\boldsymbol{p})\right) AND \ (\boldsymbol{q} \notin \boldsymbol{R}_k)\right) \tag{8.13}$$

where $\boldsymbol{p}$ and $\boldsymbol{q}$ are the 3D vectors corresponding to the pixel $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively; $\|\ \ \|$ is the standard Euclidean norm; and A_k and l_k are the area and the boundary length of the segment $\boldsymbol{R}_k$, respectively.

The contrast $C(\boldsymbol{R}_k)$ of the segment $\boldsymbol{R}_k$ is given by:

$$C(\boldsymbol{R}_k) = \begin{cases} I_k/E_k & if\ I_k \le E_k \\ E_k/I_k & otherwise \end{cases} \tag{8.10}$$

The global contrast, which is used as the measure the effectiveness of segmentation, is defined by:

$$C = \frac{1}{N}\sum_{k=1}^{n} A_k C(\boldsymbol{R}_k) \tag{8.11}$$

For a given dataset, the C-metric can be normalized to a range [0,1], in which a small value of C (close to zero) indicates good segmentation

8.4.6 Global intra-region homogeneity and inter-region disparity (DD −metric)

Rosenberger and Chehdi [75] proposed a metric for segmentation evaluation based on the global intra-region homogeneity and the global inter-region disparity of segments. This metric employs only grayscale levels of image pixels. The global intra-region homogeneity D_1 of segments is the weighted average of the pixel intensity variation of all segments:

$$D_1 = \frac{1}{N}\sum_{k=1}^{n} A_k \sum_{\boldsymbol{p}\in\boldsymbol{R}_k} \left(G(\boldsymbol{p}) - \overline{G(\boldsymbol{R}_k)}\right)^2 \tag{8.12}$$

where $G(\boldsymbol{p})$ is the grayscale level or intensity of the pixel $\boldsymbol{p}$; $\overline{G(\boldsymbol{R}_k)}$ is the average grayscale level of all pixels belong to the segment $\boldsymbol{R}_k$; and other notations are as defined earlier.

Define the disparity of two segments $\boldsymbol{R}_k$ and $\boldsymbol{R}_m$ as:

$$d(\boldsymbol{R}_k, \boldsymbol{R}_m) = \frac{\left|\overline{G(\boldsymbol{R}_k)} - \overline{G(\boldsymbol{R}_m)}\right|}{N_G} \tag{8.13}$$

where N_G is the number of gray levels of all pixels in the entire image. Then, the global inter-region disparity D_2 is defined as the average of all the disparity between any two segments, given by:

$$D_2 = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{m=1}^{n} d(\boldsymbol{R}_k, \boldsymbol{R}_m) \tag{8.14}$$

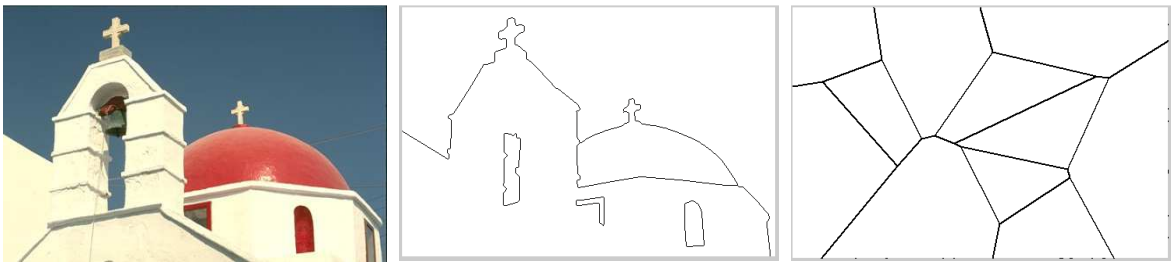
Notations in (8.14) are as defined earlier. The metric DD used as a quantitative measure of a segmentation is:

$$DD = \frac{D_1 - D_2}{2} \tag{8.15}$$

For a given dataset, the DD-metric can be normalized to a range [0,1], in which a small value of DD (close to zero) indicates good segmentation.

8.4.7 Selecting the best unsupervised metric

The key idea of selecting the best evaluation metric out of six metrics introduced above is that a better metric should produce a better score on the ground-truth segmentation (produced by human observers) and produce a worse score on random segmentations at the same time. The 300 ground-truth segmentation results are available together with the DSDB database introduced in Section 8.1. We generated 300 random segmentation results to help evaluate the various metrics. To create a random segmentation, we first generated a bitmap image having the same size as the image to be segmented (e.g. 480 pixels x320 pixels) and employed a random integer number n, $1 \leq n \leq 150$. Next, we randomly initialized n segment "centers" (n pairs of xy-coordinators for n points in the image plane). Then, we employed a K-means clustering technique to divide the bitmap image into n regions, which served as a random segmentation map consisted of random n segments with random size, shape, and location. Figure 8.5 presents an example of a random segmentation of an original image.



(a) (b) (c)

Figure 8.5: The ground-truth segmentation (b) and a random segmentation (c) of an original image (a)

For the i[th] evaluation metric, $1 \leq i \leq 6$, we calculate its 300 metric values (normalized to be in the range from 0 to 1) for the 300 ground-truth segmentations. Call the distribution of these 300 values the f_i^G distribution, and compute its mean μ_i^G and standard deviation σ_i^G. Similarly, we calculated these metric values for 300 random segmentations to form the distribution f_i^R with μ_i^R mean and σ_i^R standard deviation. Since we expect for the "good" evaluation metric, the

f^Gdistribution will be close to zero, and the f^Rdistribution will be close to one, the Fisher's distance [76], was used to measure the dissimilarity or the "distance" between two distributions. The Fisher distance $F_i$then become our quantitative measurement of the goodness of an evaluation metric, with F_i given by:

$$F_i = \frac{2\left(\mu_i^R - \mu_i^G\right)^2}{\left(\sigma_i^R\right)^2 + \left(\sigma_i^G\right)^2} \tag{8.15}$$

Table 8.1: Fisher's distances between the f_i^G distribution on the set of ground-truth segmentations and f_i^R on a single set of random segmentations

Metric score distribution	Method 1 D−metric [70]	Method 2 E−metric [71]	Method 3 Q−metric [72]	Method 4 H−metric [73]	Method 5 C−metric [74]	Method 6 DD−metric [75]
Human ground-truth segmentation	μ_1^G= 0.32	0.14	0.2	0.28	0.18	0.29
	σ_1^G= 0.14	0.08	0.17	0.11	0.12	0.16
A single set of 300 random segmentations	μ_1^R= 0.64	0.71	0.75	0.67	0.63	0.65
	σ_1^R=0.13	0.15	0.12	0.18	0.17	0.14
Fisher's distance between the two distributions	F_1 = 5.61	**22.48**	13.97	6.84	9.35	5.73

The larger the value of the Fisher distance, the more separation there is between the two distributions (ground-truth and random segmentations), and therefore the better is the metric. Note that the distributions of metric values corresponding to ground-truth segmentations are fixed, while the distributions of metric values corresponding to random segmentations vary, due to a different set of 300 random segmentation is generated each time a metric evaluation is computed. Accordingly, we ran the Fisher distance procedure for each metric 50 times (each time with a different set of 300 random segmentations) to obtain reliable statistical measures. Table 8.1

shows an sample result of a single run, in which the E-metric provides the best separation between the two distributions.

The boxplot of Fisher distance distributions corresponding to the 6 metrics after 50 runs is provided in Figure 8.5. In each box, the central mark (in red) is the median, the edges of the box (in blue) are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers, if present, are plotted individually as red crosses. The larger the Fisher's distance, the better is the metric measurement. The results clearly show that E-metric is the best among the six metrics presented herein. Accordingly, it will be used in comparisons that follow.

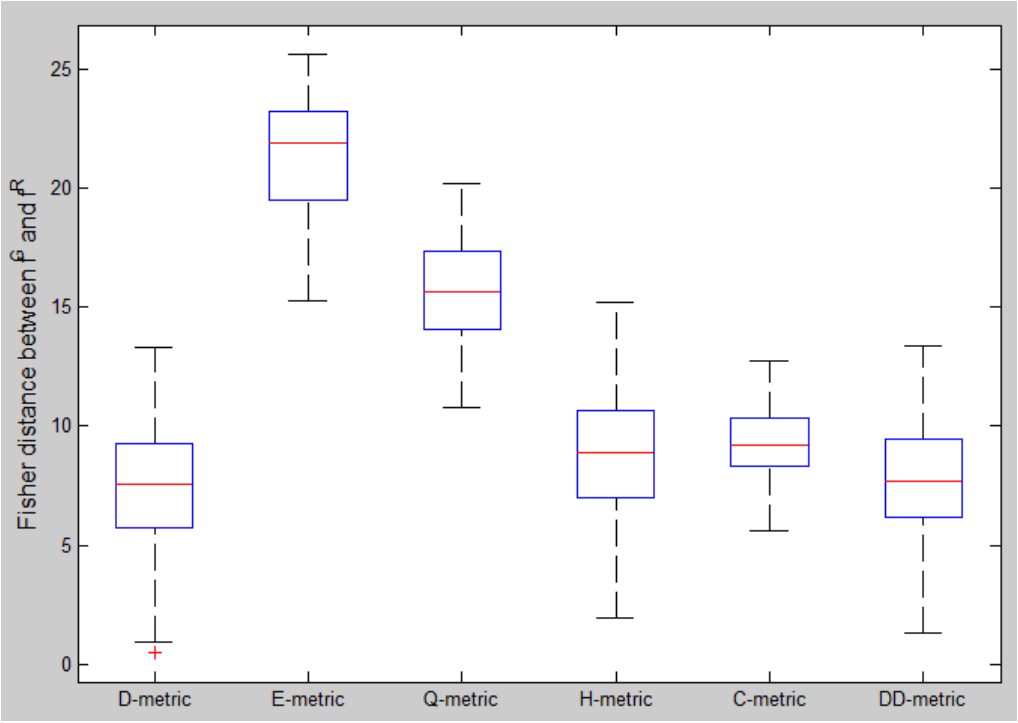


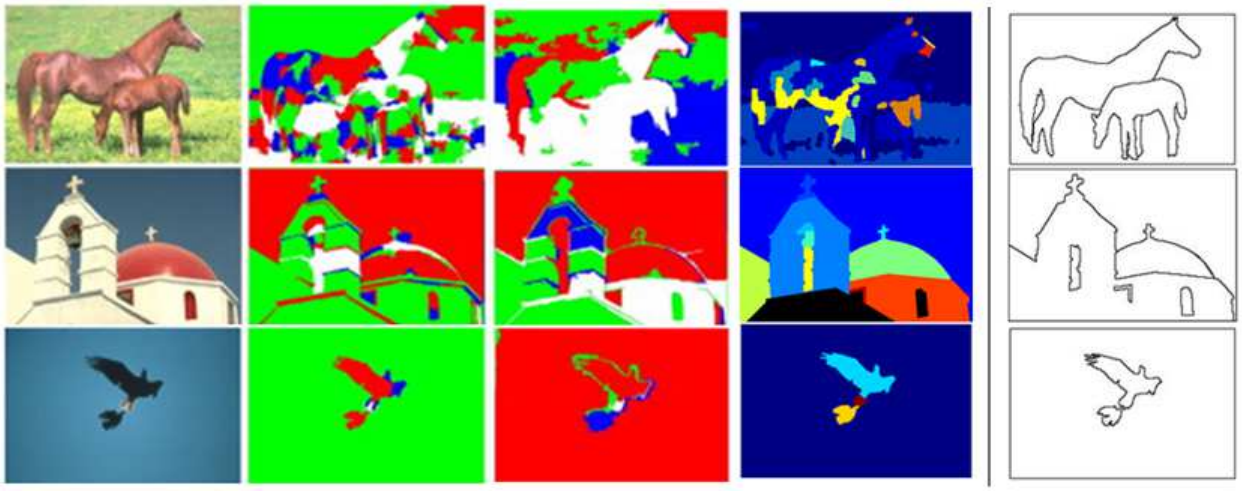
Figure 8.6: Fisher's distances distribution corresponding to 6 metric measurements after 50 randomization runs

8.5 Comparisons with other methods

8.5.1 Comparisons with non-superpixel-based and superpixel-based methods

Figure 8.7 (b)-(d) shows results obtained from the mean-shift [47] with window size $ws = \left[\frac{hi}{50} \times \frac{wi}{50}\right]$; normalized cut method [54], and our DUHO segmentation algorithm, where hi and wi are the height and width of the input image in pixels. Both the mean-shift and normalized cut algorithms segment directly at the pixel level and employ a similarity measurement based on pixel color and spatial information.

a1	b1	c1	d1	e1
a2	b2	c2	d2	e2
a3	b3	c3	d3	e3



(a) (b) (c) (d) (e)

Figure 8.7: Some Comparison: (a) original image; (b, c, and d) segmentation results from mean-shift, normalized cut, and our method, respectively; (e) human hand-label

We observe that the mean-shift method tends to produce over-segmentation, and its results are heavily dependent on the window size [22,46]. The normalized cut algorithm, working at the pixel level, might not preserve the detail edges of the input image, and hence might lead to misclassification of pixels into "incorrect" segments. Both these two methods have high storage requirements and are computationally time consuming. For example total computation times for the results in Fig 8.7, using a PC with Intel dual core 2.2 GHz CPU, 2GB RAM, Matlab©2010b, and Image Processing Toolbox Version 7.1, were as given in Table 8.1. Our DUHO algorithm

performs with significantly lower computation times than the mean-shift and normalized cut algorithms.

Table 8.2: Computational times (seconds) for segmentation of images in Fig 8.7 (Intel dual core 2.2 GHz CPU, 2GB RAM, Matlab©2010b, and Image Processing Toolbox version 7.1)

Image / **Method**	a1 (sec)	a2 (sec)	a3 (sec)
Mean-shift	135	128	117
Normalized-cut	142	140	132
DUHO method	95	92	87

The E-metric evaluation, selected in Section 8.4 as the best, was applied for 300 segmentation results on images from the Berkeley segmentation dataset and benchmark [68] for each of these 3 algorithms. Table 8.3 presents the evaluation results.

Table 8.3: Evaluation of three segmentation algorithms on the dataset

Segmentation algorithm	**Mean-shift**	**Normalized-cut**	**DUHO method**
Performance metric E (small is better): mean ± std	0.31 ± 0.11	0.36 ± 0.24	**0.26** ± 0.12

We see that our DUHO algorithm performs significantly better than the mean-shift and normalized cut algorithms.

8.5.2 Comparisons with superpixel-based methods

In this section, the segmentation results from our DUHO algorithm will be compared with 3 recently published, unsupervised color segmentation algorithms based on superpixels, namely: PSEG, GSEG, and JSEG. The main idea of the PSEG [77] is to scan through a hierarchy of image partitions, from a highly over-segmentation to a highly under-segmentation partition, to find the best partition that maximizes a predefined goodness function. In the PSEG, the pixel colors in RGB color space are used directly. The GSEG [60] is based on the unseeded region growing

technique, in which the initial seeds are found using the color gradient information (in CIE **L*a*b*** color space). After the region growing process, regions with similar characteristics are blended by the multi-resolution region merging to form the final segmentation. During the merging process, new seeds might be added or old seeds discarded .The JSEG [78] includes 2 stages, quantization and spatial segmentation. First, pixel colors (in CIE L*u*v* color space), smoothness of the local area, and texture orientations are quantized into a small number of predefined values. Then, these values are formed into a representation vector of a local region that will be clustered into different groups.

Figure 8.8 presents some segmentation results from our DUHO method and these three methods. Visually, results from all four methods appear close to human segmentation. However, our proposed segmentation process produces finer details.

The E-metric unsupervised evaluation was applied for 300 segmentation results on images from the DBSB dataset [68]. In addition, we employ supervised evaluation techniques, called the boundary recall [63] and boundary precision [79] measurement to evaluate the segmentation results. Boundary recall is the percent of the ground-truth edge pixels that are within two pixels distance from a region boundary. We can express this as:

$$Boundary\ recall = \frac{L(\text{"hit" groundtruth edge})}{L(\text{groundtruth edge})} \tag{8.16}$$

where L(.) is the length in pixels; “hit” means that the current pixel in the ground-truth edge is in the range of 2 pixels from a pixel in a region’s boundary of the segmentation result. Boundary precision is the percent of the region edge pixels (resulted from a segmentation method) that are within two pixels distance from a ground-truth edge boundary. We can express boundary presision as:

$$Boundary\ precision = \frac{L(\text{"hit" segmentation edge})}{L(\text{segmentation edge})} \tag{8.16}$$

Notice that $L(\text{"hit" segmentation edge}) = L(\text{"hit" segmentation edge})$ is the number of “mutual”- or within 2 pixels along the boundary of the ground-truth edge and the segmentation

result edge. Boundary recall and precision measure the matching degree between the ground-truth and the segmentation results. High recall value indicates that most of the "correct" boundaries are discovered in the segmentation results; while high precision value indicates that the segmentation results are more accurate or most of the segmentation boundaries are the "correct" boundaries. For a good segmentation, both of boundary recall and boundary precision values are expected to be high (near 1). Table 8.4 summarizes the evaluation results.

Table 8.4: Evaluation of our DUHO segmentation and three other algorithms on the dataset [68]

Segmentation algorithm	**Human hand-label**	**PSEG**[77]	**GSEG**[60]	**JSEG**[78]	**Proposed DUHO**
Unsupervised performance metric E (small is better)	***0.14***	0.31	0.29	0.31	**0.26**
Boundary recall (higher is better)	--	0.82	0.86	0.77	**0.89**
Boundary precision (higher is better)	--	**0.81**	0.76	0.72	0.79
Average computational time (sec)	--	232.4	162.7	145.1	**93.3**

We see that our DUHO algorithm performs better than the JSEG, GSEG, and PSEG algorithms, based on both the unsupervised metric E and the supervised boundary recall measurement. And our algorithm is the second best (and comparable with the best) among these four algorithms based on the boundary precision measurement. The values of boundary precision in Table 8.4 are not very high due to the fact that hand-label segmentation usually ignores details in the image and hence produces coarser results comparing with results from all four segmentation methods presented herein. The computation times in Table 8.4 were derived using a PC with Intel dual

core 2.2 GHz CPU, 2GB RAM, Matlab©2010b, and Image Processing Toolbox Version 7.1. Our DUHO algorithm performs significantly faster than the JSEG, GSEG, and PSEG algorithms.

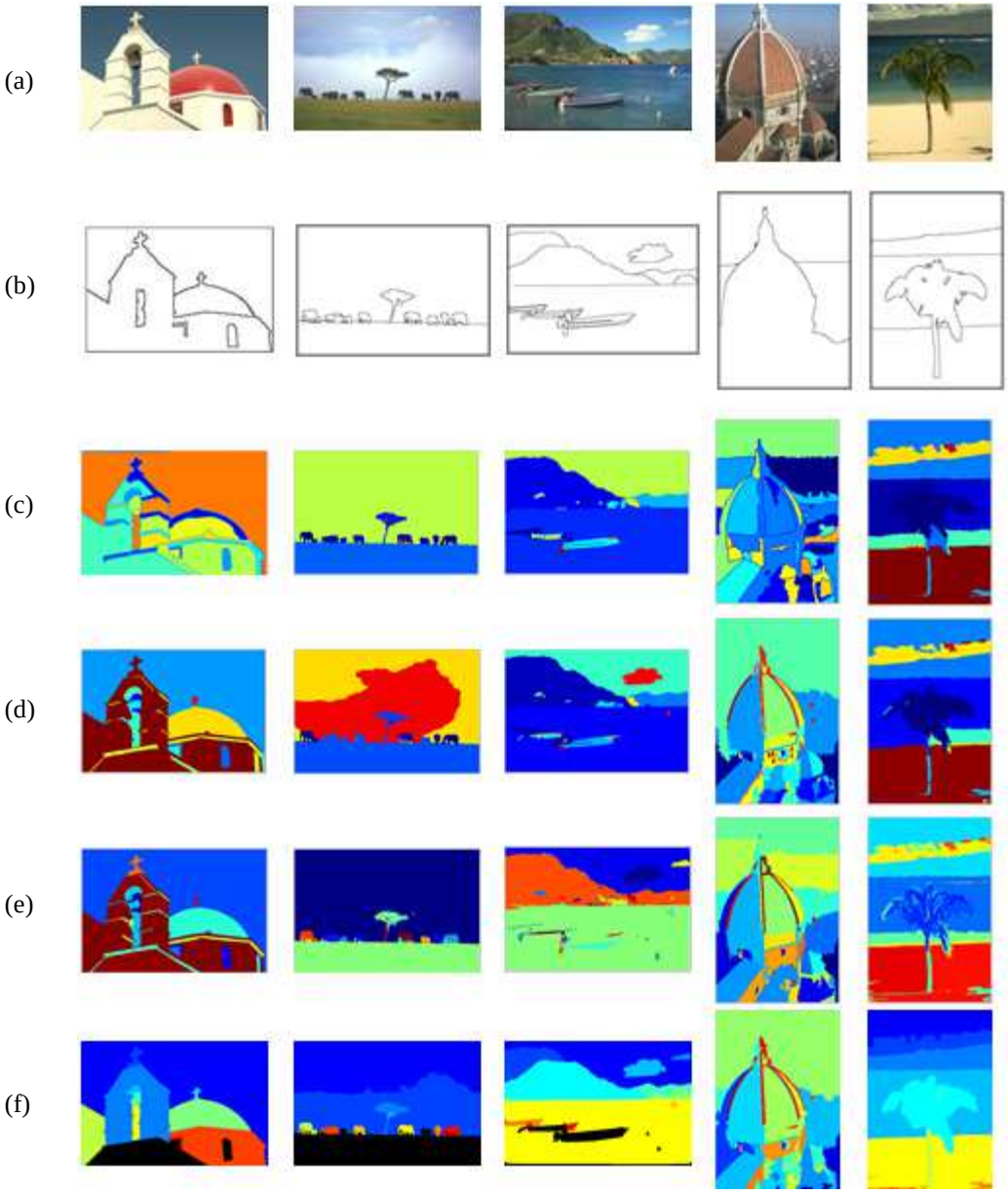


Figure 8.8: Some Comparison: row (a): original image; (b) segmentation results from human hand-label, (c) PSEG, (d) GSEG, (e) JSEG, and (f) DUHO.

Chapter 9

CONCLUSIONS AND RECOMENDATIONS

4.1 Conclusions

The principal original contributions of this work are threefold:

(1) A new automatic data clustering method for convex data sets called M-ART

(2) A new method for evaluating image segmentation metrics, which was used to identify the best available metric, namely the E-metric.

(3) A new algorithm, called the DUHO method, for segmenting color images, which outperforms existing segmentation methods, as measured by the E-metric, and requires substantially less computation time the existing methods.

This work contains two parts, in which new methods for data clustering and image segmentation are presented. In the first part, Chapter 1 to Chapter 5, an automatic data clustering procedure is proposed. First, a pre-processing process, named density-based removal, is applied to produce more distinct clusters. This pre-process is equivalent to removing input vectors near the boundaries of each cluster, which in many cases does not change the data structure or the number of "correct" clusters in the data. Then, we employ the M-ART neural network clustering technique to group similar input vectors into clusters. In the M-ART network, the vigilance ρ

determines the maximum size of clusters, and consequently affects the number of clusters. Conventionally, a trial-and-error approach is used to tune this value of ρ.

We propose a framework to auto-adjust the value of ρ based on a user-selected allowable separation between clusters. Even though one must still select the value of a parameter, choosing the allowed separation factor is intuitively easier than selecting the value of ρ. The appropriate value of ρ is strongly dependent upon on the specific data set, and is therefore very difficult to select a-priori. On the other hand, the allowed separation factor value simply represents how much separation between adjoining clusters a user is willing to accept. Accordingly, the separation factor could be chosen for multiple data sets before running the algorithm. Experiments conducted on different synthetic 2-D, 3-D, 4-D, 5-D, and 10-D Gaussian data sets, some published and some generated by the authors, with varying numbers of vectors, numbers of clusters, and different degrees of separation between clusters, demonstrate the effectiveness and reliability of the proposed clustering method. Two case studies of texture classification and texture segmentation are also presented, showing very good results when compared with those from the well-known K-means method. While the M-ART clustering method is an original contribution of this research, it applies only to convex data sets. As such, it was unsuccessful in applications to image segmentation, a goal of this study. Accordingly, we investigated a different approach to this area in Part II of this work.

In the second part of this dissertation, we introduce a general-purpose segmentation method, which we call the DUHO method, which works for a large variety of natural scene images in color. This DUHO algorithm contains two main steps. First, a superpixel generating algorithm is applied to a given image to build *K* superpixels. Then a new region growing algorithm iteratively groups these superpixels into appropriate regions and forms the final image segmentation result. The proposed method is a type of unseeded region-based segmentation technique that preserves the spatial relationship between pixels in the image, and hence preserves the detailed edges and the image spatial structure. Our DUHO algorithm has three main advantages compared with other region-growing-based segmentation techniques [57-60]. First, it operates at a “superpixel” level,

rather than at the image pixel level, to reduce computational time and depress noise. Second, the proposed method works for color images rather than gray scale image as in [57-60]. Third, the decision of grouping an adjacent superpixel to an existing region is dynamically dependent upon the statistics, or "shape and size", of this region. The segmentation results show significant improvements when compared with results from existing methods using a fixed, global threshold. The control parameter SF, the smallest feature size, in our DUHO algorithm controls the coarseness or fineness of the segmentation results. The "optimal" value of SF is dependent upon the image features and the purpose or application of the segmentation process, and it should be appropriately selected by users or follow the rules of thumb suggested in Chapter 7. A quantitative evaluation method based on square color error is introduced, and experiments with real datasets shows very good results when compared with those from other published, state-of-art segmentation methods, as well as requiring substantially less computational time.

4.2 Recommendations

Recommendations for future work are:

(1) Finding means to automatically tune parameters in the segmentation algorithm, such as $\boldsymbol{W}$ and SF, which must currently be established manually using trial and error. A classical optimization procedure (iterative-based or gradient-based) might be employed to find the optimal control parameters values to maximize the overall segmentation performance based on some score index.

(2) Proving the convergence of the proposed algorithm to the minimum of the performance function.

REFERENCES

[1] M. Ball and S. G. W. H., *Analyzing visual data*. Newbury Park, CA: Sage Publications, 1992.

[2] H. Frank, *et al.*, *Fuzzy Cluster Analysis, Methods for Classification, Data Analysis and Image Recognition*. NY: John Wiley & Sons Ltd, 1999.

[3] A. D. Gordon, *Classification*, 2 ed.: Chapman & Hall/CRC, 1999.

[4] E. Martin, *et al.*, "Density-based clustering in spatial databases: the algorithm GDBSCAN and its application," *Data Mining and Knowledge Discovery*, vol. 2, pp. 169–194, 1998.

[5] A. Mihael, *et al.*, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOID'99 Int. Conf. on Management of Data*, Philadelphia PA, 1999.

[6] M. Markou and S. Singh, "Novelty detection: a review—part 1: statistical," *Signal Processing* vol. 83, pp. 2481 – 2498, 2003.

[7] M. Markou and S. Singh, "Novelty detection: a review—part 2: neural network based approaches," *Signal Processing* vol. 83, pp. 2499 – 2521, 2003.

[8] H.-P. Kriegel, *et al.*, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, pp. 1-58, 2009.

[9] J. C. Bezdek and N. R. Pal, "Some new indexes of cluster validity," *IEEE Trans. Systems Man Cybernet. B Cybernet*, vol. 28, pp. 301-315, 1998.

[10] D. W. Kim, *et al.*, "On cluster validity index for estimation of the optimal number of fuzzy clusters," *Pattern Recognition*, vol. 37, pp. 2009 – 2025, 2004.

[11] C. Böhm, *et al.*, "Computing Clusters of Correlation Connected Objects " in *In Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2004.

[12] D. L. Reilly, *et al.*, "The use of multiple measurements in taxonomic problems," *Annu. Eugenics*, vol. 45, pp. 35–41, 1982.

[13] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, pp. 23-63, 1987.

[14] G. A. Carpenter and S. Grossberg, *Adaptive Resonance Theory, the Handbook of Brain Theory and Neural Networks*, 2 ed.: Cambridge, MA: MIT Press, 2003.

[15] C. J. Lakhmi, *et al.*, *Innovations in ART neural networks*. NY: Physica- Verlag Heidelberg, 2000.

[16] M. T. Hagan, *et al.*, *Neural Network Design*: PWS Publishing, Massachusetts, 2002.

[17] H. Xu, "Mahalanobis Distance-Based ARTMAP Networks," Master, Computer Science, San Diego State University, 2003.

[18] K. Giyadh and C. C. Ka, "Euclidean ART neural network," in *Proceeding of the World Congress on Engineering and Computer Science*, San Francisco, USA, 2008.

[19] K. Giyadh and C. C. Ka, "Euclidean ART neural network," *Proc. of the World Congress on Engineering and Computer Science 2008*, October 22-24, 2008 2008.

[20] P. C. Mahalanobis, "On the generalised distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, pp. 49–55, 1936.

[21] M. I. Vuskovic, *et al.*, "Simplified ARTMAP Network Based on Mahalanobis distance," in *Proceeding of the 2002 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Science*, Las Vegas, Nevada, 2002.

[22] D. S. Watkins, *Fundamentals of matrix computations*: John Wiley& Sons, Inc., 1991.

[23] G. Golub and C. Van Loan, *Matrix computations (3rd ed.)*: Johns Hopkins University Press, 1996.

[24] D. J. Field, "Relations between the statistics of natural images and the response properties of cortical cells " *Journal of The Optical Society of America A,* vol. 4, pp. 2379--2394, 1987.

[25] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*: Prentice Hall, 2002.

[26] R. W. Picard, *et al.*, "Real-time recognition with the entire Brodatz texture database," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, New York, 1993, pp. 638-639.

[27] D. J. Field, "Relations between the statistics of natural images the response properties of cortical cells " *Journal of The Optical Society of America A,* vol. 4, pp. 2379--2394, 1987.

[28] R. W. Picard, *et al.*, "Real-time recognition with the entire Brodatz texture database," in *Proc. of the IEEE Conf. on CVPR*, New York, 1993, pp. 638-639.

[29] A. K. Jain, F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters,"*Pattern Recognition*, vol. 24, no. 12, pp.1167-1186, 1991.

[30] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc., Ser. B*, vol. 39, no. 1, pp. 1-38, 1977.

[31] Mori, G. *Guiding model search using segmentation*. in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. 2005.

[32] Felzenszwalb, P.F. and D.P. Huttenlocher, *Efficient Graph-Based Image Segmentation.* Int. J. Comput. Vision, 2004. **59**(2): p. 167-181.

[33] Stockman, G. and L.G. Shapiro, *Computer Vision*2001: Prentice Hall PTR. 608.

[34] Pham, D.L., C. Xu, and J.L. Prince, *Current methods in medical image segmentation.* Annual Review of Biomedical Engineering, 2000. **2**(1): p. 315-337.

[35] Mueller, M., K. Segl, and H. Kaufmann, *Edge- and region-based segmentation technique for the extraction of large, man-made objects in high-resolution satellite imagery*. Pattern Recognition, 2004. **37**(8): p. 1619-1628.

[36] Segundo, M.P., et al., *Automatic Face Segmentation and Facial Landmark Detection in Range Images.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2010. **40**(5): p. 1319-1330.

[37] Proen, x00E, and H. a, *Iris Recognition: On the Segmentation of Degraded Images Acquired in the Visible Wavelength.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2010. **32**(8): p. 1502-1516.

[38] Akram, M.U., et al. *Improved fingerprint image segmentation using new modified gradient based technique*. in *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*. 2008.

[39] Raut, S., et al. *Image Segmentation – A State-Of-Art Survey for Prediction*. in *Advanced Computer Control, 2009. ICACC '09. International Conference on*. 2009.

[40] Camargo, A. and J.S. Smith, *An image-processing based algorithm to automatically identify plant disease visual symptoms.* Biosystems Engineering, 2009. **102**(1): p. 9-21.

[41] Zhang, H., J.E. Fritts, and S.A. Goldman, *Image segmentation evaluation: A survey of unsupervised methods.* Computer Vision and Image Understanding, 2008. **110**(2): p. 260-280.

[42] Byoung-Ki, J., J. Yun-Beom, and H. Ki-Sang. *Image Segmentation by Unsupervised Sparse Clustering*. in *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*. 2005.

[43] Heiler, M. and C. Schnorr. *Natural image statistics for natural image segmentation*. in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. 2003.

[44] Islam, M., J. Yearwood, and P. Vamplew, *Unsupervised Color Textured Image Segmentation Using Cluster Ensembles and MRF Model Advances in Computer and*

Information Sciences and Engineering, T. Sobh, Editor 2008, Springer Netherlands. p. 323-328.

[45] Lucchi, A., et al., *A Fully Automated Approach to Segmentation of Irregularly Shaped Cellular Structures in EM Images Medical Image Computing and Computer-Assisted Intervention – MICCAI 2010*, T. Jiang, et al., Editors. 2010, Springer Berlin / Heidelberg. p. 463-471.

[46] Martínez-Usó, A., F. Pla, and P. García-Sevilla, *Unsupervised colour image segmentation by low-level perceptual grouping.* Pattern Analysis & Applications, 2011: p. 1-14.

[47] Polak, M., H. Zhang, and M. Pi, *An evaluation metric for image segmentation of multiple objects.* Image and Vision Computing, 2009. **27**(8): p. 1223-1227.

[48] Senthilkumaran, N. and R. Rajesh. *Image Segmentation - A Survey of Soft Computing Approaches*. in *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on*. 2009.

[49] Ni, K., et al., *Local Histogram Based Segmentation Using the Wasserstein Distance.* International Journal of Computer Vision, 2009. **84**(1): p. 97-111.

[50] Gonc, et al., *HAIRIS: A Method for Automatic Image Registration Through Histogram-Based Image Segmentation.* Image Processing, IEEE Transactions on, 2011. **20**(3): p. 776-789.

[51] Hartigan, J.A., *Clustering Algorithms*1975: John Wiley \\& Sons, Inc. 351.

[52] Yizong, C., *Mean shift, mode seeking, and clustering.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1995. **17**(8): p. 790-799.

[53] Lindeberg, T. and M.-X. Li, *Segmentation and Classification of Edges Using Minimum Description Length Approximation and Complementary Junction Cues.* Computer Vision and Image Understanding, 1997. **67**(1): p. 88-98.

[54] Jianbo, S. and J. Malik, *Normalized cuts and image segmentation.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000. **22**(8): p. 888-905.

[55] Miranda, P. and A. Falcão, *Links Between Image Segmentation Based on Optimum-Path Forest and Minimum Cut in Graph.* Journal of Mathematical Imaging and Vision, 2009. **35**(2): p. 128-142.

[56] Jingdong, W., et al. *Normalized tree partitioning for image segmentation*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008.

[57] Adams, R. and L. Bischof, *Seeded region growing.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1994. **16**(6): p. 641-647.

[58] Shah, B.N., S.K. Shah, and Y.P. Kosta. *A seeded region growing algorithm for spot detection in medical image segmentation*. in *Image Information Processing (ICIIP), 2011 International Conference on*. 2011.

[59] Shih, F.Y. and S. Cheng, *Automatic seeded region growing for color image segmentation.* Image and Vision Computing, 2005. **23**(10): p. 877-886.

[60] Garcia Ugarriza, L., et al., *Automatic Image Segmentation by Dynamic Region Growth and Multiresolution Merging.* Image Processing, IEEE Transactions on, 2009. **18**(10): p. 2275-2288.

[61] Bleau, A. and L.J. Leon, *Watershed-Based Segmentation and Region Merging.* Computer Vision and Image Understanding, 2000. **77**(3): p. 317-370.

[62] Lolla, S.V.G. and Hoberock L.L.. *Improved Unsupervised Clustering over Watershed-Based Clustering*. in *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*. 2010.

[63] Radhakrishna, A., et al., *SLIC Superpixels ⋆*, 2010.

[64] Ohta, Y.-I., T. Kanade, and T. Sakai, *Color information for region segmentation.* Computer Graphics and Image Processing, 1980. **13**(3): p. 222-241.

[65] Fairchild, M.D., *Color and Image Appearance Models*2005: John Wiley and Sons.

[66] Jain, A.K., *Data clustering: 50 years beyond K-means.* Pattern Recognition Letters, 2010. **31**(8): p. 651-666.

[67] [69] Felzenszwalb, P. and D. Huttenlocher, *Efficient Graph-Based Image Segmentation.* International Journal of Computer Vision, 2004. **59**(2): p. 167-181.

[68] Martin, D., et al. *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*. in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on.* 2001.

[69] Hsin-Chia, C. and W. Sheng-Jyh. *The use of visible color difference in the quantitative evaluation of color image segmentation*. in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*. 2004.

[70] Weszka, J. S. Threshold Evaluation Techniques, University of Maryland, College Park, 1998.

[71] Hsin-Chia, C. and W. Sheng-Jyh (2004). *The use of visible color difference in the quantitative evaluation of color image segmentation*. Acoustics, Speech, and Signal Processing, (ICASSP '04). IEEE International Conference, 2004.

[72] Borsotti M., Campadelli P., and Schettini R., "*Quantitative evaluation of color image segmentation results*," Pattern Recognition Letters, vol. 19, pp. 741-747, 1998

[73] Zhang, H., Fritts, J. E., & Goldman, S. A. *An entropy-based objective evaluation method for image segmentation*. SPIE, 5307, 38-49. (2005)

[74] Chabrier, S., B. Emile, H. Laurent, C. Rosenberger, and P. Marche. "*Unsupervised evaluation of image segmentation application to multi-spectral images.*" In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, vol. 1, pp. 576-579. IEEE, 2004.

[75] Rosenberger, C., and K. Chehdi. "Genetic fusion: application to multi-components image segmentation." *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. Vol. 6. IEEE, 2000.

[76] Johson, R., Wichem, D. Applied multivariate statistical analysis. Princeton Hall, 1992.

[77] Martínez-Usó, Adolfo, Filiberto Pla, and Pedro García-Sevilla. "Unsupervised colour image segmentation by low-level perceptual grouping." *Pattern Analysis & Applications* (2011): 1-14.

[78] Deng, Yining, and B. S. Manjunath. "Unsupervised segmentation of color-texture regions in images and video." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23.8 (2001): 800-810.

[79] Powers, David M W (2007/2011). "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation". *Journal of Machine Learning Technologies* **2** (1): 37–63.

VITA

Trung Huy Duong

Candidate for the Degree of

Doctor of Philosophy

NEW TECHNIQUES FOR DATA CLUSTERING
AND COLOR IMAGE SEGMENTATION

Major Field: Mechanical and Aerospace Engineering

Biographical:

Personal Data: Born in Amthuong, Phutho Province, Vietnam, on December 10th, 1981, the second son of Duong Huy Lung and Vu Thi Luc. Married to Doan Huong Ly.

Education: Graduated from Vinhphuc Talented High School, Vinhphuc, Vietnam, in 1999. Received a Bachelor degree in Mechatronics, Mechanical Engineering from Talented Engineers Training Program, Center for Talent Training, Hanoi University of Technology, Vietnam, in Jun 2004. Received a Master of Science degree in Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2009. Completed the requirements for the Doctor of Philosophy in Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2013

Experience: Lecture Assistant, Department of Mechanical Engineering, University of Communication and Transports, Vietnam, from 2004 to 2007. Graduate Research Assistant and Research Associate, Department of Mechanical and Aerospace Engineering, Oklahoma State University from January 2008 to Dec 2013.