Multi-Manifold Learning and Voronoï Region-Based Segmentation with an Application in Hand Gesture Recognition

by

Randima Nuwangi De Silva Hettiarachchi

A Thesis submitted to The Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

August 2016

Abstract

A computer vision system consists of many stages, depending on its application. Feature extraction and segmentation are two key stages of a typical computer vision system and hence developments in feature extraction and segmentation are significant in improving the overall performance of a computer vision system. There are many inherent problems associated with feature extraction and segmentation processes of a computer vision system. In this thesis, I propose novel solutions to some of these problems in feature extraction and segmentation.

First, I explore manifold learning, which is a non-linear dimensionality reduction technique for feature extraction in high dimensional data. The classical manifold learning techniques perform dimensionality reduction assuming that original data lie on a single low dimensional manifold. However, in reality, data sets often consist of data belonging to multiple classes, which lie on their own manifolds. Thus, I propose a multi-manifold learning technique to simultaneously learn multiple manifolds present in a data set, which cannot be achieved through classical single manifold learning techniques.

Secondly, in image segmentation, when the number of segments of the image is not known, automatically determining the number of segments becomes a challenging problem. In this thesis, I propose an adaptive unsupervised image segmentation technique based on spatial and feature space Dirichlet tessellation as a solution to this problem. Skin segmentation is an important as well as a challenging problem in

computer vision applications. Thus, thirdly, I propose a novel skin segmentation technique by combining the multi-manifold learning-based feature extraction and Voronoï region-based image segmentation.

Finally, I explore hand gesture recognition, which is a prevalent topic in intelligent human computer interaction and demonstrate that the proposed improvements in the feature extraction and segmentation stages improve the overall recognition rates of the proposed hand gesture recognition framework. I use the proposed skin segmentation technique to segment the hand, the object of interest in hand gesture recognition and manifold learning for feature extraction to automatically extract the salient features. Furthermore, in this thesis, I show that different instances of the same dynamic hand gesture have similar underlying manifolds, which allows manifold-matching based hand gesture recognition.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my PhD advisor Prof. James F. Peters for his tremendous support and guidance given to me throughout the past four years. This thesis and the other contributions of my PhD research work would not have been possible if it wasn't for his constant guidance, encouragement and inspiration. Secondly, I would like to thank my internal committee members Prof. Robert McLeod, Prof. Pradeepa Yahampath and Prof. Robert Thomas for their valuable support, insightful comments and encouragement given throughout. I would also like to thank Prof. Marek Reformat for his insightful comments and encouragement as my external examiner.

I would like to extend my heartfelt gratitude to my husband, Ishan Wickramasingha for always being there to discuss my research ideas and for encouraging and inspiring me, and my mother, Chandani Hettiarachchi for being my pillar of strength throughout my life. Without their love, encouragement, moral support and countless sacrifices, this thesis would never have been possible. I would like to dedicate this thesis to my husband, my mother and my late father, Samanjith Hettiarachchi, who inspired me to pursue my higher studies in Computer Engineering.

My sincere thanks also go to University of Manitoba for providing me with financial support through University of Manitoba Graduate Fellowship and Engineering graduate awards, which helped me to continue my PhD studies and research work smoothly. Finally, I would like to thank my family away from home, my dear friends in Winnipeg and my lab mates for their constant support and encouragement and the staff of Department of Electrical and Computer Engineering, University of Manitoba for proving a peaceful working environment.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AFHA	Ant Colony Fuzzy C-means Hybrid Algorithm
AS	Ant System
CBIR	Content Based Image Retrieval
COIL20	Columbia Object Image Library
DSPF	Discriminative Skin-Presence Feature
FPSS	Fast Propagation-based Skin-region Segmentation
FSD	Face and Skin Detection
HE	Hessian Eigenmaps
HGR	Hand Gesture Recognition
HMM	Hidden Markov Model
ICA	Independent Component Analysis
ISOMAP	Isometric Feature Mapping
JTS	Jochen Triesch Static Hand Posture Database
KCC	k-Connected Components
KNN	k-Nearest Neighbor
LDA	Linear Discriminant Analysis
LE	Laplacian Eigenmaps
LPP	Locality Preserving Projection
LLE	Locally Linear Embedding

MDS	Multidimensional Scaling
M-ISOMAP	Multi-Manifold Isometric Feature Mapping
MMD	Manifold-Manifold Distance
MMDA	Multi-Manifold Discriminant Analysis
MM-LLE	Multi-Manifold Locally Linear Embedding
MMVR	Multi-Manifold Voronoi Region
MSE	Mean Squared Error
PCA	Principal Component Analysis
PMD	Point-to-Manifold Distance
RFHA	Region Splitting and Merging-Fuzzy C-means Hybrid Algorithm
RSM	Region Splitting and Merging
SASS	Self-adaptive Algorithm for Skin-region Segmentation
SIMPLIcity	Semantics-sensitive Integrated Matching for Picture Libraries
SLLE	Supervised Locally Linear Embedding
SVM	Support Vector Machine

Publications

- Journal Publications:

1. **R. Hettiarachchi** and J. Peters, "Voronoï Region-Based Adaptive Unsupervised Color Image Segmentation.," *Pattern Recognition*, Elsevier, In Press, Accepted on 12 December 2016.
2. **R. Hettiarachchi** and J. Peters, "Multi-Manifold-Based Skin Classifier on Feature Space Voronoï Regions for Skin Segmentation.," *Journal of Visual Communication and Image Representation*, Elsevier, vol. 41, pp. 123-139, 2016.
3. **R. Hettiarachchi** and J. Peters, "Multi-Manifold LLE Learning in Pattern Recognition.," *Pattern Recognition*, Elsevier, vol. 48, no.9, pp. 2947–2960, 2015.
4. **R. Hettiarachchi** and J. Peters, "Dimensionality Reduction via Proximal Manifolds.," *General Mathematics Notes*, vol. 22, no.2, pp. 133-142, 2014.
5. J. Peters and **R. Hettiarachchi**, "Proximal Manifold Learning via Descriptive Neighbourhood Selection.," *Applied Mathematical Sciences*, vol.8, no.71 pp. 3513-3517, 2014.
6. **R. Hettiarachchi**, J. Peters, and N. Bruce, "Fence-like Quasi-periodic Texture Detection in Images.," *Theory and Applications of Mathematics*

& Computer Science, vol.4, no.2, pp.123-139, 2014.

7. J. Peters and **R. Hettiarachchi**, "Visual Motif Patterns in Separation Spaces.," *Theory and Applications of Mathematics & Computer Science*, vol.3, no.2, pp. 3658, 2013.

- Conference Publications:

1. C. J. Henry, J. F. Peters, **R. Hettiarachchi**, and S. Ramanna, "Content-Based Image Retrieval Using a Metric Free Nearness Measure," *The 15th IASTED International Conference on Signal and Image Processing*, 2013.

Chapter 1

Introduction

The organization of a computer vision system mainly depends on its application. Figure 1.1 depicts the key stages of a typical computer vision system. The ordering of these stages may change and some stages such as feature extraction may be repeated depending on the application. According to Figure 1.1, feature extraction and segmentation are two key stages of a typical computer vision system. Feature extraction extracts meaningful information (features) from the original data (e.g., images) and segmentation retrieves an object of interest from the original data for further processing. The performance of later stages such as classification and recognition strongly depends on the performance of earlier stages such as feature extraction and segmentation. Thus, developments in feature extraction and segmentation significantly contribute to the overall performance of a computer vision system.

In this thesis, I propose novel solutions for some of the inherent problems in feature extraction and segmentation stages of a computer vision system. To start with, I explore manifold learning for feature extraction in computer vision applications. Manifold learning has been extensively studied over the years as a feature extraction technique and it performs feature extraction through non-linear dimensionality reduc-

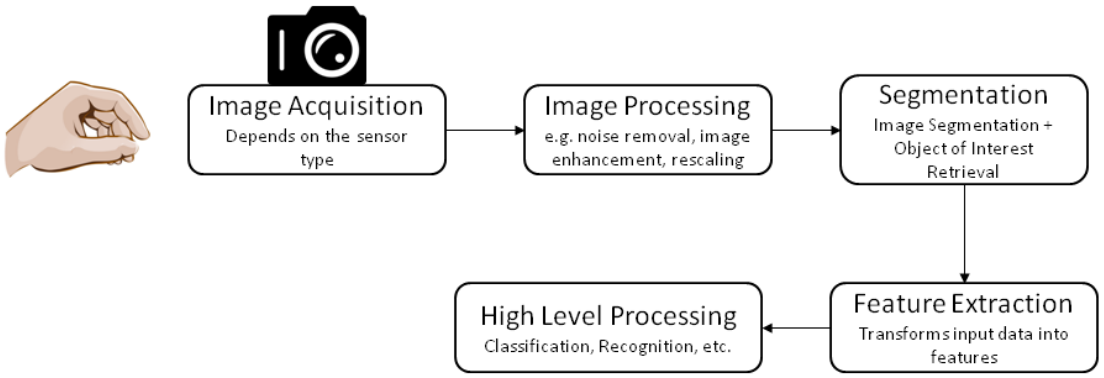


Figure 1.1: Key stages of a typical computer vision system

tion. Thus, in addition to automatic feature extraction, manifold learning addresses the *curse of dimensionality* (challenges in handling very high dimensional data such as images) problem in computer vision applications too. In this thesis, I specifically address the problem of simultaneously learning multiple manifolds present in a data set with a multiple class structure, which cannot be achieved through classical single manifold learning techniques. Furthermore, I introduce point-to-manifold distance-based classification to classify new data samples in the multi-manifold space, which provides better and more consistent recognition rates.

With regard to segmentation, first I address the problem of automatically determining the number of segments of a given image, which is a challenging problem in automatic image segmentation. I use a combination of spatial and feature space Dirichlet tessellation of a given image in order to achieve this. Then, I combine the proposed image segmentation technique and multi-manifold learning based feature extraction technique to propose a solution for the skin segmentation problem, which is an important yet a challenging problem in a wide range of computer vision applications such as human computer interaction and medical image (skin cancer) analysis. In most of these applications, skin segmentation is used to segment the object of

interest (e.g. hand, face, pigmented skin) through detection of skin regions.

Finally, I explore hand gesture recognition, which is a prevalent topic in intelligent human computer interaction and demonstrate how the proposed improvements to the feature extraction and segmentation stages improve the overall recognition rates of the proposed hand gesture recognition system. Hand gesture recognition can be subdivided into static hand gesture (hand posture) recognition and dynamic hand gesture recognition. Segmenting the hand, which is the object of interest in this application, is a challenging task when it comes to both static and dynamic hand gesture recognition.

In this thesis, I use the proposed skin segmentation technique to segment the hand region in hand gesture images and I propose multi-manifold learning for feature extraction in hand posture recognition and single manifold learning for feature extraction in dynamic hand gesture recognition. Furthermore, I show that different instances of the same dynamic hand gesture have similar underlying manifolds, which allows recognition of new hand gestures by comparing their corresponding manifolds with reference manifolds of known gestures. Moreover, I use proximity theory (set proximity and metric proximity) to build the theoretical foundation of the solutions proposed in this thesis.

1.1 Manifold Learning for Feature Extraction

Feature extraction is the process of transforming input data into features that can be useful for machine learning algorithms in higher level processes such as classification and recognition. According to [8], one major goal of feature extraction is to improve the robustness of salient features extracted from the input data, while also potentially removing noise and redundancy from the input. There are various feature extraction

methods proposed in the past literature and Storcheus et al. categorize those into *feature selection* methods, which select a subset from large input feature set, *feature weighting* methods, which aim at finding a best weight for each feature and *feature construction* methods, which involve synthesizing new features or mapping onto new feature spaces from the existing set of features in [8].

Manifold learning has been extensively studied over the past decade as a feature extraction technique. Storcheus et al. [8] categorize dimensionality reduction techniques including manifold learning as feature construction-based feature extraction methods. Manifold learning performs feature extraction through non-linear dimensionality reduction based on the assumption that the intrinsic dimensionality of many high dimensional real world data sets is small and that they lie on low dimensional geometric structures called manifolds.

Briefly, a manifold is a topological space that is locally Euclidean, i.e., around every point in the space, there is a neighborhood that is topologically the same as the open unit ball in $\mathbb{R}^n$. A topological space X is a non-empty set together with a collection of subsets τ that we call *open sets* such that the following properties hold true.

1^o The entire space X is open.

2^o The empty set $\emptyset$ is open.

3^o If open sets A, B are in τ, then $A \cup B$ (union of A and B) is open and is in τ.

4^o If open sets A, B are in τ, then $A \cap B$ (intersection of A and B) is open and is in τ.

In the case of 2-dimensional (planar) manifolds, for every point in the plane, there is neighborhood (denoted by $N_r(x_0)$) which is an open set of radius r and with center

x_0 such that

$$N_r(x_0) \left\{ x \in \mathbb{R}^2 : \|x - x_0\| < r \right\},$$

where $\| \, . \, \|$ is the Euclidean distance.

In $\mathbb{R}^1$, an open ball is an open interval. In $\mathbb{R}^2$ (Euclidean plane), an open ball is an open disk (a plane disk with no boundary points). In $\mathbb{R}^3$ (Euclidean 3-space), an open ball is an open sphere (a spherical region with no boundary or surface points). In all three cases, an open unit ball is a neighborhood $N_1(x_0)$ centered on the origin with radius $r = 1$. For more about manifolds, see [9], [10, §18.3, p. 130], [11, §8.1, p. 89], [12].

Manifold learning differs from other dimensionality reduction techniques such as Principal Component Analysis (PCA) [13] because of its non-linear character and its structure-preserving mapping. There are variations of PCA, e.g., Kernel PCA [14], which is a non-linear form of PCA and Sparse PCA [15], which aims to find principal components that are both sparse and explain as much of the variance in the data as possible. However, these techniques do not focus on structure-preserving mapping during dimensionality reduction.

Let X and Y be topological spaces. A structure-preserving mapping $f : X \to Y$ is a homomorphism, such that

1^o f is one-to-one and onto.

2^o f is continuous.

3^o f^{-1} (inverse) $: Y \to X$ is continuous.

A mapping is one-to-one, provided distinct points mapped onto distinct points, e.g., if $x_1, x_2 \in X, x_1 \neq x_2$, and $f(x_1), f(x_2) \in Y$ then points $f(x_1) \neq f(x_2)$. A mapping $f : X \to Y$ is onto (surjective), provided, for each $y \in Y$, there is a $x \in X$ such that $y = f(x)$. A function $f : X \to Y$ between topological spaces X and Y is continuous,

if and only if the inverse of each open set is also open, i.e., whenever $B \subseteq Y$ is an open set, then the inverse $f^{-1}(B)$ is an open set in X. For more about this, see [16, §1.3, p. 8], [10, §7, p. 44] and [17].

There are numerous applications of manifold learning in pattern recognition and computer vision. For example, in the areas of image classification [18–20] and object recognition [21–23]. In computer vision applications, we are mostly dealing with digital images, which provide a typical example of a real-world high dimensional vector space. Manifold learning is advantageous in computer vision and pattern recognition in two different ways. First, it learns the hidden structures in high dimensional data and it preserves that structure while mapping those high dimensional data onto lower dimensions. Thus, manifold learning performs automatic feature extraction and provides a more meaningful representation of data in lower dimensions. Patterns, which may be hidden in higher dimensions, can be discovered in lower dimensions through manifold learning. Second, manifold learning addresses the problem of *curse of dimensionality* by reducing the dimensionality of data, which improves the computational efficiency of higher level processing stages in a computer vision system.

1.1.1 Definition of a Manifold

Let's first look at the definition of a manifold from another perspective, namely, manifolds that are metric topological spaces. This is implicit in what I have mentioned earlier, since an open unit ball is defined in terms of a Euclidean norm, which is the magnitude of the difference between vectors in the Euclidean space $\mathbb{R}^n$. That is, a manifold is a metric topological space that is locally similar to an Euclidean space.

Definition 1.1. *A* manifold M *is a metric topological space with the following property: if* $x \in M$*, then there is some neighborhood* U *of* x *and some integer* $n \geq 0$ *such*

that U is homeomorphic to $\mathbb{R}^n$. ■

A homeomorphic mapping is a mapping that is continuous, one-to-one and onto with a continuous inverse [24].

1.1.2 Manifold Learning Techniques

Manifold learning serves as a dimensionality reduction technique. Dimensionality reduction can be categorized into linear and non-linear techniques. Examples for linear techniques are Principal Component Analysis (PCA) [13] and Independent Component Analysis (ICA) [25]. Manifold learning falls under the non-linear dimensionality reduction techniques. Some other non-linear dimensionality reduction techniques are Polynomial PCA [26], Kernel PCA [14] and Multidimensional Scaling (MDS) [27].

Manifold learning techniques can be broadly categorized relative to global and local techniques. Isometric feature mapping (ISOMAP) [28] is a well-known global manifold learning algorithm and Locally Linear Embedding (LLE) [29,30], Laplacian Eigenmaps (LE) [31] and Hessian Eigenmaps (HE) [32] are methods that employ local manifold learning techniques. Because of their computational efficiency, local non-linear manifold learning algorithms have gained prominence.

1.1.3 Locally Linear Embedding (LLE)

The LLE manifold learning technique will be frequently discussed in this thesis. Hence, in this section, I briefly introduce the LLE algorithm based on [33] and [21, §2.2]. Basically, LLE recovers each global non-linear structure from locally linear fits. With LLE, each low-dimensional embedding is found, provided the neighbors of each point in a high-dimensional space are mapped to corresponding neighbors of a point in a low-dimensional space. An embedding is a representation of a topological object,

manifold, graph, field, etc. in a certain space in such a way that its connectivity or algebraic properties are preserved [34].

Suppose that a data set contains data points $\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_N \in \mathbb{R}^D$. Let ε be a positive real number and let k be a positive integer. The neighborhood of $\boldsymbol{x}_i$ is denoted by $N(\boldsymbol{x}_i)$, containing the k nearest data points to $\boldsymbol{x}_i$. That is, for each $\boldsymbol{x}_j \in N(\boldsymbol{x}_i)$, the distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is less than some ε. Each data sample $\boldsymbol{x}_i$ is linearly represented by its neighbors in $N(\boldsymbol{x}_i)$. To do this, the weights $\{w_{ij}\}_{\boldsymbol{x}_j \in N(\boldsymbol{x}_i)}$ will be found by minimizing (1.1).

$$\varepsilon^2 = E_1^{(i)}\left(\{w_{ij}\}_{\boldsymbol{x}_j \in N(\boldsymbol{x}_i)}\right) = \left[\boldsymbol{x}_i - \sum_{\boldsymbol{x}_j \in N(\boldsymbol{x}_i)} w_{ij}\boldsymbol{x}_j\right]^2, \tag{1.1}$$

subject to the condition $\sum_{\boldsymbol{x}_j \in N(\boldsymbol{x}_i)} w_{ij} = 1$ and $w_{ij} > 0$. Also, it is important to note that $w_{ij} = 0$, if $\boldsymbol{x}_j \notin N(\boldsymbol{x}_i)$. These optimal weights are invariant to three types of transformations scaling, orthogonal transformation and translation. Equation (1.1) can be written as

$$\varepsilon^2 = \left[\boldsymbol{x}_i - \sum_{j=1}^{k} w_{ij}\boldsymbol{x}_j\right]^2 = \left[\sum_{j=1}^{k} w_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)\right]^2 = \sum_{jk} w_{ij}w_{ik}\mathbb{G}_{jk},$$

where $\mathbb{G}_{jk} = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_k)$ is the local Gram matrix and $\boldsymbol{x}_j, \boldsymbol{x}_k \in N(\boldsymbol{x}_i)$. Reconstruction weights are found using the Lagrange multiplier method [21]. Lagrange multipliers can be used to find the extrema of a multivariate function subject to constraints [35].

$$w_{ij} = \frac{\sum_k \mathbb{G}^{-1}_{jk}}{\sum_{lm} \mathbb{G}^{-1}_{lm}}.$$

These reconstruction weights are used to find the d-dimensional vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N \in \mathbb{R}^d$ that define the d-dimensional embedding of the original data set

in $\mathbb{R}^D$ such that $d < D$. Such vectors will be found by minimizing (1.2).

$$E_2(\mathbb{Y}) = E_2\left(\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N\right) = \sum_{i=1}^{N} \left[\boldsymbol{y}_i - \sum_{\boldsymbol{y}_j \in N(\boldsymbol{y}_i)} w_{ij} \boldsymbol{y}_j \right]^2, \text{ where} \tag{1.2}$$

$$\mathbb{Y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix}_{N \times d} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1d} \\ y_{21} & y_{22} & \cdots & y_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nd} \end{bmatrix}_{N \times d} = \begin{bmatrix} \boldsymbol{Y}_1 & \boldsymbol{Y}_2 & \cdots & \boldsymbol{Y}_d \end{bmatrix}_{N \times d}. \tag{1.3}$$

(1.2) can be rewritten in the following matrix form.

$$\begin{aligned} E_2(\mathbb{Y}) &= \|\mathbb{Y} - \mathbb{W}\mathbb{Y}\|^2 = \sum_{k=1}^{d} \left[\boldsymbol{Y}_k - \mathbb{W}\boldsymbol{Y}_k\right]^2 = \sum_{k=1}^{d} (\boldsymbol{Y}_k - \mathbb{W}\boldsymbol{Y}_k)^T (\boldsymbol{Y}_k - \mathbb{W}\boldsymbol{Y}_k) \\ &= \sum_{k=1}^{d} \boldsymbol{Y}_k^T (I - \mathbb{W})^T (I - \mathbb{W}) \boldsymbol{Y}_k = \sum_{k=1}^{d} \boldsymbol{Y}_k^T \mathbb{M} \boldsymbol{Y}_k, \end{aligned}$$

where $\mathbb{W}$ is an $N \times N$ weight matrix and

$$\mathbb{M}_{N \times N} = (I - \mathbb{W}_{N \times N})^T (I - \mathbb{W}_{N \times N}).$$

Each $\boldsymbol{Y}_i$ is orthogonal and regulated. I.e. $\boldsymbol{Y}_i^T \boldsymbol{Y}_j = 0,\ i \neq j,\ i, j = 1, 2, \ldots, d$ and $E(|\boldsymbol{Y}_i|^2) = \frac{1}{N} \boldsymbol{Y}_i^T \boldsymbol{Y}_i = 1,\ i = 1, 2, \ldots, d$ respectively. The optimal solution is calculated using the Lagrange multiplier method in (1.4).

$$E_2(\mathbb{Y}) = \sum_{k=1}^{d} \boldsymbol{Y}_k^T \mathbb{M} \boldsymbol{Y}_k - \sum_{k=1}^{d} \lambda_k \left(\frac{1}{N} \boldsymbol{Y}_k^T \boldsymbol{Y}_k - 1 \right). \tag{1.4}$$

Next, differentiate (1.4) with respect to $\boldsymbol{Y}_k, k = 1, 2, ...d$. Then $\boldsymbol{Y}_k^T \mathbb{M} - \lambda_k \boldsymbol{Y}_k^T = 0$ implies $\mathbb{M}\boldsymbol{Y}_k = \lambda_k \boldsymbol{Y}_k$. As a result, the $\{\lambda_k\}$ are eigenvalues of $\mathbb{M}$, while $\{\boldsymbol{Y}_k\}$ are the corresponding eigenvectors. Thus, $E_2(\mathbb{Y})$ can be minimized by finding the bottom eigenvalues of $\mathbb{M}$ and their corresponding eigenvectors. $\boldsymbol{Y}_0^T$ has the eigenvalue $\lambda_0 = 0$. Hence, the lowest $d+1$ eigenvalues of $\mathbb{M}$ and their corresponding $d+1$ eigenvectors are found and $\boldsymbol{Y}_0$ will be discarded. The eigenvectors corresponding to the bottom $d+1$ eigenvalues are taken in this case, since this is a minimization problem.

1.2 Image Segmentation

1.2.1 Overview of Image Segmentation

Image segmentation is the process of dividing an image into different regions such that each region is, but the union of any two adjacent regions is not, homogenous [36]. Figure 1.2 shows a typical example of pixel color-based image segmentation. In Figure 1.2, image pixels are partitioned into several segments based on their color. This new representation given by image segmentation makes it easier to further analyze the image. Image segmentation is typically used to detect and segment the objects of interest in an image, which provides the points or regions in the image that are relevant for further processing. For example, in hand gesture recognition, only the region containing the hand is relevant for further processing to recognize the type of hand gesture.

The existing image segmentation techniques can be broadly categorized into threshold-based, clustering-based, region-based, edge-based and physics-based segmentation approaches [36, 37]. There are various hybrid image segmentation tech-

niques, which combine two or more of the aforementioned approaches as well. The choice of image segmentation technique depends on its application.

(a) Original image (b) Segmented image

Figure 1.2: An example for image segmentation

1.2.2 Image Tessellation-Based Image Segmentation

Image tessellation is a tiling of an image surface with one or more geometric shapes and Dirichlet tessellation (also called Voronoï diagram) is one example of image tessellation. Dirichlet [38] introduced polygon-based tessellation in 1850, which was elaborated by Voronoï in 1907 [39]. A Voronoï diagram is the partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other. Thus, a Voronoï region can be defined in the following way.

Definition 1.2. ***Voronoï Region***

Let S and X be finite sets in an n-dimensional Euclidean space. A Voronoï region of $p \in S$ (denoted V_p) is defined by

$$V_p = \{x \in X : \|x - p\| \leq_{\forall q \in S} \|x - q\|\},$$

where S is the set of generating points.

According to Definition 1.2, in order to generate the Voronoï diagram, the generating points or the seed points have to be provided. In our context, these seed points can be *corners, centroids, critical points or scale-invariant feature transform (SIFT) key points [40]* found in images. Figure 1.3 shows an example of Dirichlet tessellation (in green) of an image using image corner points (in blue) as generating points. In this example, the image is spatially tessellated as the (x,y) coordinates of the pixels and the generating points are considered when tessellating the image. Du et al. introduced the technique of Centroidal Voronoï Tessellations (CVT) in 1999 [41], which uses centroids as the generating points for the Voronoï tessellation. In 2006, Du et al. revisits the CVT algorithm in their subsequent article [42], which focuses more on the applications of CVT.

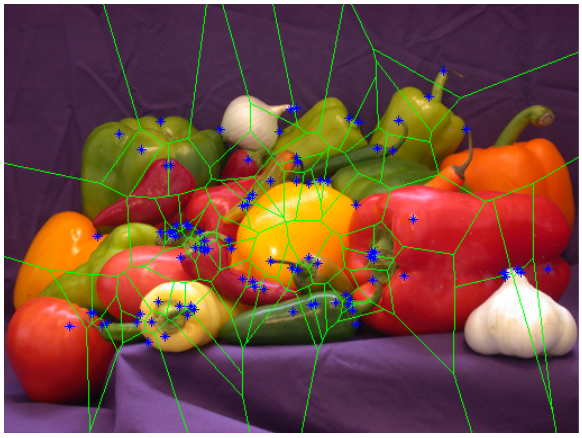


Figure 1.3: An example of Dirichlet tessellated image

Voronoï regions (convex polygons produced by Dirichlet tessellation) have been explored as a solution to the image segmentation problem during the past two decades. An interesting paper on Voronoï based image segmentation is *On Points Geometry for Fast Digital Image Segmentation* [43]. In [43], rather than applying the Voronoï Diagram on the image itself, it is applied on a few selected points by using the image

histogram. Suhail et al. also suggested Voronoï cells for image segmentation in [44]. The articles [41,42] focus on spatial Dirichlet tessellation while [43] focus on Dirichlet tessellation on image histograms.

1.3 Skin Segmentation

Skin segmentation is the classification of an input colored image into skin and non-skin regions based on features such as color and texture. Skin segmentation plays an important role in a wide range of applications in image processing and computer vision. In most of these applications, skin segmentation is used to segment the object of interest through detection of skin regions. For example, skin segmentation is used in hand and face detection and tracking for human computer interaction in [45,46]. In Figure 1.4, skin segmentation is used to segment the hand region in an image. Another popular application of skin segmentation is objectionable content filtering [47–49], which uses skin segmentation to detect the amount of skin present in images or videos to block pornographic content.

(a) Original image (b) Skin segmented image

Figure 1.4: An example for skin segmentation

Skin segmentation is also used for content-based image retrieval in [50, 51] to retrieve skin lesion images similar to a given query image as a diagnostic aid and for

skin lesion segmentation and analysis in medical imaging in [52, 53] to segment skin lesions in dermoscopic images. In [50–53], skin segmentation is used to differentiate pigmented skin from healthy skin in order to segment the pigmented skin patch. Another application of skin segmentation is image coding using region of interest [54, 55], which uses skin segmentation to find the region of interest (e.g. face).

There are numerous skin segmentation techniques proposed in the literature and those can be broadly categorized into threshold-based techniques and model-based techniques. The articles [7, 56, 57] provide a good analysis and comparisons of the existing skin segmentation methods. Although there are various skin segmentation techniques proposed in the past literature, [57] observes that no satisfactory solution has been developed so far. This is mainly due to the challenges in automatic skin segmentation of images with varying backgrounds and background regions having color similar to skin (e.g. tablecloth in Figure 1.4).

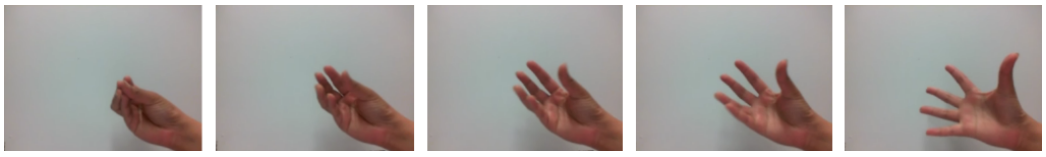
1.4 Hand Gesture Recognition

Vision-based hand gesture analysis and recognition has been a widely discussed topic in intelligent human computer interaction systems in the recent past. It has various applications in virtual reality [58], sign language recognition [59–61], sterile human-machine interfaces [62,63], tele-medicine [64], tele-rehabilitation systems [65,66], hand gesture-controlled games [67] and many more. Vision-based hand gesture recognition can be subdivided into two main categories: model-based approaches and appearance-based approaches. Model-based approaches consider the kinematic parameters which map the 2D projection image to the 3D hand model, which can then be used to recognize new hand gestures. The appearance-based approaches use the appearance of predefined 2D image templates of known hand gestures to recognize new hand

gestures.

In a real world setting, hand gestures can be static or dynamic. Static hand gestures (also called hand postures) do not vary over time while the dynamic hand gestures may vary over time. Static hand gestures are defined as orientation and position of hand in the space during a period of time without any movement [68]. Dynamic hand gestures can be defined as movement of hand during a given amount of time. Thus, in addition to spatial variations, dynamic hand gestures show temporal variations as well. Figure 1.5 shows some examples of hand postures and a dynamic hand gesture.

(a) Static hand gestures (hand postures)



(b) A dynamic hand gesture

Figure 1.5: Examples of static and dynamic hand gestures

1.5 Motivation and Objectives

1.5.1 Motivation

As discussed in the beginning of this chapter, the performance of the high level processing stage (e.g. recognition) of a typical computer vision system significantly depends on the performance of feature extraction and segmentation stages. Feature

extraction plays a major role in extracting the salient features from original data, which are later used by machine learning algorithms and segmentation helps to extract the object of interest, which is relevant for further processing. Thus, my main motivation is the significance of developments in feature extraction and segmentation on improving the overall performance of a computer vision system. In this thesis, I address several inherent problems in feature extraction and segmentation based on the motivations summarized below.

- The classical manifold learning techniques perform dimensionality reduction assuming that points in the data set lie on a single manifold in the low dimensional embedding. In reality, we often come across data sets with data belonging to multiple classes where points in each data class lie on their own manifold. The classical single manifold learning techniques are not capable of learning multiple manifolds present in a data set simultaneously. Hence, multi-manifold learning techniques capable of simultaneously learning multiple manifolds in a data set are essential in such cases. Furthermore, in practical applications, local manifold learning techniques are preferred over global techniques because of their computational efficiency. However, the existing multi-manifold learning techniques are not applicable to local non-linear manifold learning techniques. Thus, a multi-manifold learning technique based on local manifold learning is vital for practical applications of multi-manifold learning.

- In cluster-based image segmentation techniques, determining the number of clusters and cluster centroids is crucial for accurate segmentation of the image. In the case of color images, which are complex data sets by nature, determination of the number of pixel clusters and the cluster centroids becomes very challenging. Thus, adaptive unsupervised image segmentation techniques,

which automatically find the number of clusters and the corresponding cluster centroids are vital for successful image segmentation.

- Distinguishing skin regions from similar color background regions (non-skin regions with color similar to skin) is one of the major challenges when it comes to skin segmentation. The state-of-the-art methods attempt to distinguish skin pixels from similar color background pixels by analyzing their features such as color and texture. However, these techniques still fail to provide a satisfactory result when both the skin and similar color background regions are detected with high skin probability based on the selected feature set. In such cases, verification of the boundaries between skin and similar color background regions through image segmentation will lead to more accurate skin segmentation results. Furthermore, having a balanced training data set and a proper feature extraction technique to extract the salient features further improve the skin classifier accuracy.

- Hand gesture recognition is a prevalent topic in human-computer interaction. The motivations for the proposed improvements in the hand gesture recognition framework proposed in this thesis are listed below.

  - Although manifold learning has been explored in static hand gesture (hand posture) recognition, only the classical single manifold learning techniques have been explored so far. However, since the hand posture data set consists of data belonging to multiple gestures (classes), multi-manifold learning can be expected to provide a more meaningful low dimensional representation of training data for hand posture recognition.

  - In the case of dynamic hand gesture recognition, although different in-

stances of the same dynamic hand gesture performed by different people at different times look different due to noise and personal differences, their underlying manifolds have to be similar as all of them represent the same hand gesture. Thus, it is possible to recognize unknown dynamic hand gestures by comparing their corresponding manifolds with reference manifolds of known dynamic hand gestures.

– In both static and dynamic hand gesture recognition, segmentation of a hand, particularly in complex backgrounds is a challenging task. Also, proper detection of the boundaries of the hand region is important to extract features such as shape descriptors. In such situations, skin segmentation provides a great tool to segment the hand region in gesture recognition systems.

1.5.2 Objectives

The main objective of this research work is to improve the overall performance of a computer vision system by providing solutions to some inherent problems in feature extraction and segmentation stages. Different objectives were set when addressing those problems in feature extraction and segmentation as given below.

1. To develop a locally linear multi-manifold learning technique, which can simultaneously learn multiple manifolds present in a data set with data belonging to multiple classes.

2. To develop an adaptive unsupervised image segmentation technique by using spatial and feature space Dirichlet tessellation, which can automatically segment an image by adaptively finding the number of clusters and cluster centroids in the given image.

3. To develop a more robust skin segmentation technique by using multi-manifold learning for feature extraction and Voronoï region-based image segmentation to verify the skin classification result.

4. To develop a more robust hand gesture recognition framework with the following objectives:

    - To improve hand posture recognition rates by using multi-manifold learning for feature extraction in order to simultaneously learn hand posture manifolds of each type of hand posture.
    - To improve dynamic hand gesture recognition rates by using manifold learning for feature extraction in order to simultaneously analyze global and local motion of hand gestures.
    - To recognize unknown dynamic hand gestures by comparing their corresponding manifolds with reference manifolds of known dynamic hand gestures.
    - To improve the accuracy of hand segmentation process in both static and dynamic hand gesture recognition by using the proposed skin segmentation technique.

1.6 Major Contributions of the Thesis

In this thesis, I first explore multi-manifold learning-based feature extraction and Voronoï region-based adaptive unsupervised image segmentation to improve feature extraction and segmentation processes of a computer vision system. Then, I combine the proposed multi-manifold learning-based feature extraction method and Voronoï region-based image segmentation method to develop a more robust automatic skin

segmentation technique. Finally, I propose a hand gesture recognition framework by using manifold learning for feature extraction and the proposed skin segmentation technique for hand (object of interest) segmentation in hand gesture recognition. The major contributions of this thesis are as listed below.

1. Multi-Manifold LLE Learning

    - A multi-manifold learning algorithm based on LLE to simultaneously learn multiple manifolds present in a data set with data belonging to multiple classes.
    - Reduced computational complexity compared to other multi-manifold learning algorithms.
    - Finds the optimum low-dimensional space that gives high recognition rates by minimizing the nearness of manifolds.
    - Introduces the point-to-manifold distance-based classification to classify new data samples, which provides better and more consistent recognition rates.

2. Voronoï Region-Based Adaptive Unsupervised Color Image Segmentation

    - An adaptive unsupervised image segmentation algorithm, which automatically finds the number of pixel clusters and cluster centroids of a given image.
    - A hybrid of spatial and feature space Dirichlet tessellation to adaptively segment an image.
    - Determines the number of clusters and cluster centroids within each spatial Voronoï region rather than in the whole image, which significantly reduces the complexity of the segmentation problem.

- Introduces proximal cluster merging based on centroid proximity to merge similar clusters within each pair of spatial Voronoï regions to automatically find the number of clusters and cluster centroids of an image.

3. Multi-Manifold and Voronoï Region-Based Skin Segmentation

   - Builds a balanced training data set for the skin classifier, which is favorable for standard learning algorithms.
   - Performs image-segmentation-based verification of skin classifier result, which improves skin segmentation accuracy.
   - Uses multi-manifold-based feature extraction to support non-linear training data with a multiple class structure.

4. Application of Manifold Learning and Skin Segmentation in Hand Gesture Recognition

   - Improves hand posture recognition rates by using multi-manifold learning for feature extraction and point-to-manifold distance-based classification.
   - Simultaneously analyzes global and local motion of dynamic hand gesture by using manifold learning for feature extraction.
   - Proves that different instances of the same dynamic hand gesture have similar underlying manifolds.
   - Performs new dynamic hand gesture recognition by comparing their underlying manifold with reference manifolds of known hand gestures.

1.7 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the multi-manifold learning algorithm based on LLE, which is capable of learning multiple manifolds of a given data set. Chapter 3 introduces the Voronoï region-based image segmentation technique and Chapter 4 presents the proposed skin segmentation technique, which combines multi-manifold learning technique proposed in Chapter 2 and image segmentation technique proposed in Chapter 3. The proposed hand gesture recognition framework is presented in Chapter 5. Finally, chapter 6 discusses the conclusions and future directions of the research work presented in this thesis. Figure 1.6 depicts the road map of this thesis.

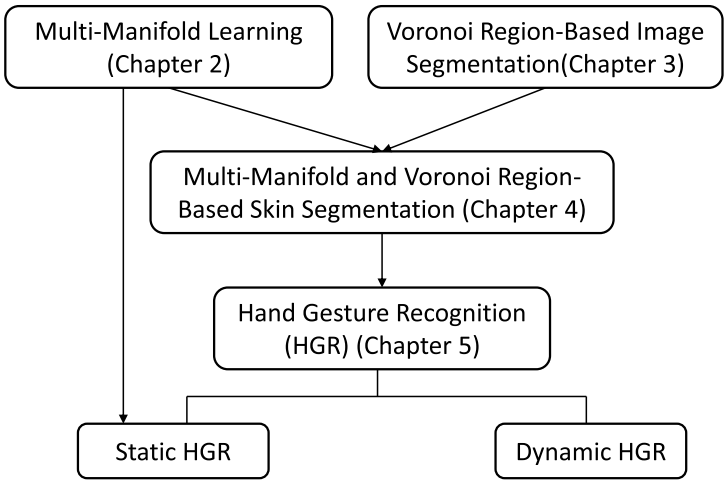


Figure 1.6: Road map of the thesis

Chapter 2

Multi-Manifold LLE Learning in Pattern Recognition

2.1 Introduction

In this chapter, I present a technique to simultaneously learn multiple manifolds present in a data set, which is not possible with classical manifold learning techniques. Manifold learning techniques have been used extensively in solving image classification problems [18–20] and object recognition problems [21–23], which involves data belonging to multiple classes. A *data class* is a grouping of data based on some characterization of the data such as a class of edge points or a class of script points. Most of the manifold based classification methods focus on performing classification on the manifold, assuming that the original data with multiple classes are on a single manifold.

In pattern recognition, we often come across situations where the data belonging to multiple classes do not lie on a single manifold. In other words, if the original data set has data belonging to multiple classes, then the data belonging to each class

lie on a particular manifold. This situation is illustrated in Fig. 2.1, where the data in two neighborhoods of points x, y (denoted by $N(x), N(y)$) belong to two different classes, namely, Class 1 and Class 2. Under such conditions, it is essential for the manifold learning algorithm to preserve the local structure of the individual manifolds belonging to multiple classes of data by performing multi-manifold learning.

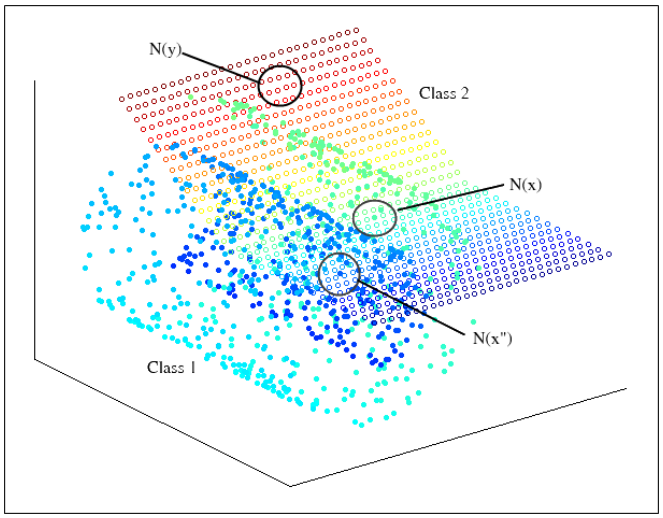


Figure 2.1: Neighborhood selection in classical LLE algorithm

The multi-manifold learning problem is closely related to subspace clustering. *Subspace clustering refers to the task of finding a multi-subspace representation that best fits a collection of points taken from a high dimensional space* [69]. Data belonging to multiple classes in a given data set could have been drawn from multiple subspaces. By performing multi-manifold learning, I expect to find a subspace representation, which represents all data classes in a data set. However, in the case of multi-manifold learning, in addition to the problem of clustering, learning individual manifolds belonging to each data class plays a major role as well.

Fan et al. [70] proposed an algorithm to perform multi-manifold learning based

on the ISOMAP manifold learning algorithm (M-Isomap). Since ISOMAP is a global non-linear manifold learning algorithm, it first learns the individual manifolds separately and learns a skeleton representing the global structure of the data simultaneously. Secondly, the embeddings of the manifolds are relocated to a global coordinate system by referring to the low dimensional representation of the skeleton.

Yang et al. [71] propose Multi-Manifold Discriminant Analysis (MMDA) in performing multi-manifold learning based on Locality Preserving Projection (LPP) [72]. LPP is seen as an alternative to PCA and Multidimensional Scaling (MDS) [27]. Because of the linear nature of LPP, the multi-manifold learning algorithm proposed in [71] finds the optimum projection in order to achieve within-class compactness and between-class separability at the same time with the help of linear discriminant analysis (LDA). Similar linear approaches for multi-manifold learning are also considered in [73, 74]. M-ISOMAP is a global technique and MMDA is a linear technique. Thus, the multi-manifold learning techniques introduced in [70] and [71, 73, 74] are not applicable to local non-linear manifold learning techniques.

LLE is widely used in solving image classification and object recognition problems. However, the classical LLE algorithm is not capable of learning multiple manifolds in a given data set. Thus, finding a multi-manifold learning algorithm based on LLE is significant for the advancement of LLE-based pattern recognition applications. In this chapter, I introduce an LLE-based multiple manifold learning method (briefly, MM-LLE), which is a local non-linear multi-manifold learning technique that preserves the class structure of the data. To achieve this, the proposed MM-LLE algorithm differs from LLE in several ways.

In contrast to LLE, I use a supervised form of neighborhood selection in the first phase of the algorithm. Although the supervised form of LLE and its applications in

classification were discussed in the past literature [75,76], the multi-manifold learning capability of supervised LLE (SLLE) has not been explored. Secondly, I use the nearness of manifolds in the multi-manifold space as a measure to find an optimum low dimensional embedding for a given data set. Such a nearness measure makes it possible to obtain high classification accuracy. Also, my focus is on the topology of the individual manifolds in multi-manifold spaces learnt by using the proposed MM-LLE algorithm. Techniques such as Ensemble Manifold Regularization [77], which combines the automatic intrinsic manifold approximation and semi-supervised classifier learning can be more challenging in the case of multi-manifold learning.

Furthermore, in contrast to conventional classification techniques, I classify an unknown data sample by finding the nearest manifold to the same data sample in the multi-manifold space. Therefore, unlike [70] and [71], which mainly focus on maintaining within-class distances and achieving separability of manifolds during classification, the proposed method focuses on preserving the structure and the remoteness of individual manifolds in the multi-manifold space. The proposed multi-manifold learning algorithm was evaluated by comparing its performance with several single manifold learning techniques as well as multi-manifold learning techniques found in the literature. The experiments were conducted by using synthetic multi-manifold data sets as well as several real world data sets.

The rest of the chapter is organized as follows. The methodology of the proposed LLE-based multi-manifold algorithm is explained in Section 2.3. Section 2.4 presents the experimental results of the proposed MM-LLE algorithm on traditional sample data sets and real world data sets. These experimental results are discussed in detail in Section 2.5. Finally, Section 2.6 provides the conclusions of this chapter.

2.2 Preliminaries

Here, I briefly introduce notation to indicate that one set is near another set, provided the sets have at least one common point. Throughout the thesis, a point in $\mathbb{R}^n$ is denoted by x and the n-dimensional feature vector representation of x is denoted by $\boldsymbol{x}$. Let V be a finite-dimensional linear (vector) space, $A, B \subset V$, $x, y \in V$. The Hausdorff distance from a point to a set $D(x, A)$ is defined by $D(x, A) = inf\ \{d(x, y) : y \in A\}$ and $d(x, y)$ is the Manhattan distance between points x and y defined by equation 2.1. The Čech closure [78] of A (denoted by clA) is defined by $clA = \{x \in V : D(x, A) = 0\}$.

Definition 2.1. ***Near Sets (Set Proximity)*** [79]
Let V be a finite-dimensional linear (vector) space and $A, B \subset V$. The sets A and B are proximal (near) (denoted $A\ \delta\ B$), provided $clA\ \cap\ clB \neq \emptyset$.

Whenever sets A and B have no points in common, the sets are *far* from each other, denoted by $A\ \not\delta\ B$, where $clA \cap\ clB = \emptyset$.

Definition 2.2. ***Far Sets*** *[80] Let V be a finite-dimensional linear (vector) space, $A, B \subset V$. The sets A and B are remote (far) (denoted $A\ \not\delta\ B$), provided $clA \cap clB = \emptyset$*

Keep in mind that each point $x \in X$ is represented by a feature vector $\boldsymbol{x} = \{\phi_1(x), \phi_2(x), \ldots, \phi_D(x)\}$, where $\phi_1, \phi_2, \ldots, \phi_D$ are probe functions for different features. The Manhattan distance $d(x, y)$ between feature vectors is defined by,

$$d(x, y) = \sum_{i=1}^{D} |\phi_i(x) - \phi_i(y)| \text{ (Manhattan distance).} \tag{2.1}$$

Next, I introduce the neighborhood of a point, which will be useful in the rest of this thesis.

Definition 2.3. ***Neighborhood of a Point***

Let V be a finite-dimensional linear (vector) space and $X \subset V$. Choose $\varepsilon > 0$, a positive real number. The neighborhood of a point $x \in X$ is the set of all points in X that are within some ε distance of x (denoted by $N_\varepsilon(x)$), defined by

$$N_\varepsilon(x) = \{y \in X : \|x - y\| < \varepsilon\}.$$

2.3 Multiple Manifold Learning

In the LLE algorithm, for a given data set, the local geometry in the neighborhood of each data point is characterized by the linear coefficients that reconstruct the data point from its neighbors. Each reconstruction is local to its neighborhood and confined to linear subspaces. Once the data in the high dimensional space is mapped to a low dimensional space, the weights w_{ij} that reconstruct the inputs in high dimensions should also reconstruct its embedded manifold coordinates in low dimensions.

Thus, LLE algorithm preserves the local geometry when mapping the data from a high dimensional space to a low dimensional space by making the nearby points in the high dimensional space remain nearby in the low dimensional space.

2.3.1 Single Class - Single Manifold

First consider the case where the data set consists of data belonging to a single class. The classical LLE algorithm is capable of handling this scenario. To apply the LLE algorithm on X, I define the k-nearest neighborhood of a point $x_i \in X$ based on Definition 2.3. Let $N_\varepsilon(x)^c$ denote the complement of neighborhood $N_\varepsilon(x)$, *i.e.*, the set of all points in X that are not in the neighborhood.

Definition 2.4. ***k-nearest neighborhood***

Let $\varepsilon > 0, k > 0$ and let $N_{\varepsilon,k}(x)$ denote the k-nearest neighborhood of a point $x \in X$, defined by

$$N_{\varepsilon,k}(x) \subset N_{\varepsilon}(x) :$$

$$\forall x_i \in N_{\varepsilon,k}(x) \text{ and } \forall x_j \in N_{\varepsilon,k}(x)^c \cap N_{\varepsilon}(x), \|x - x_i\| < \|x - x_j\| \text{ and } |N_{\varepsilon,k}(x)| = k. \quad \blacksquare$$

where $|N_{\varepsilon,k}(x)|$ denotes the cardinality (number of elements) of set $N_{\varepsilon,k}(x)$.

Next, I show that any two points x_i and x_j in the data set with the same feature vectors, have common k-nearest neighbors.

Theorem 2.1. *Let x_i and x_j be feature vectors in $\mathbb{R}^n$. If $x_i = x_j \Rightarrow \|x_i - x_j\| = 0$, then $N_{\varepsilon,k}(x_i) = N_{\varepsilon,k}(x_j)$.*

Proof. For $x_i, x_j \in \mathbb{R}^n$, if $x_i = x_j$, then $N_{\varepsilon}(x_i) = N_{\varepsilon}(x_j)$. From Def. 2.4, $N_{\varepsilon,k}(x_i) \subset N_{\varepsilon}(x_i)$ and $N_{\varepsilon,k}(x_j) \subset N_{\varepsilon}(x_j)$. Also, if $x_i = x_j$, $\forall x' \in N_{\varepsilon,k}(x_i), \|x_i - x'\| = \|x_j - x'\|$ and $\forall x'' \in N_{\varepsilon,k}(x_j), \|x_i - x''\| = \|x_j - x''\|$. Hence, $N_{\varepsilon,k}(x_i) = N_{\varepsilon,k}(x_j)$. $\square$

2.3.2 Multiple Classes - Multiple Manifolds

Next, consider a data set that consists of data belonging to multiple classes. For simplicity, I consider data belonging to two classes. Let X be the data set consisting of $N+P$ feature vectors describing points sampled from a high dimensional data space such that first N points belong to class 1 and the last P points belong to class 2. Let $c \in \{1, 2\}, L \in \{N, P\}$. Each feature vector $\boldsymbol{x}_i^c$ of a point x_i^c in X is a D-dimensional feature vector denoted by, $\boldsymbol{x}_i^c = (x_{1,i}^c, x_{2,i}^c, ..., x_{D,i}^c)^T, 1 \leq i \leq L$, *i.e.*,

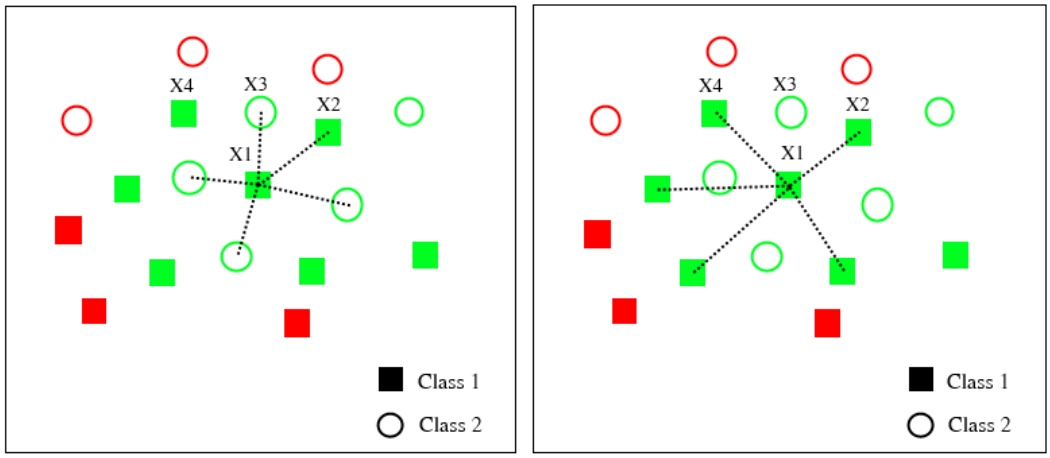
$$
\begin{aligned}
X &= \begin{bmatrix} \boldsymbol{x}_1^1 & \boldsymbol{x}_2^1 & \cdots & \boldsymbol{x}_N^1 & \boldsymbol{x}_1^2 & \boldsymbol{x}_2^2 & \cdots & \boldsymbol{x}_P^2 \end{bmatrix} \\
&= \begin{bmatrix} x_{1,1}^1 & x_{1,2}^1 & \cdots & x_{1,N}^1 & x_{1,1}^2 & x_{1,2}^2 & \cdots & x_{1,P}^2 \\ x_{2,1}^1 & x_{2,2}^1 & \cdots & x_{2,N}^1 & x_{2,1}^2 & x_{2,2}^2 & \cdots & x_{2,P}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{D,1}^1 & x_{D,2}^1 & \cdots & x_{D,N}^1 & x_{D,1}^2 & x_{D,2}^2 & \cdots & x_{D,P}^2 \end{bmatrix}
\end{aligned}
$$

The classical LLE algorithm is not capable of learning multiple manifolds simultaneously, because in the classical algorithm, the neighborhood of a given point is determined regardless of the class label. As proven in theorem 2.1, any two points having the same feature vector, but belonging to two different classes, will share common neighbors and hence will be reconstructed by the same set of neighbors in the low dimensional space. Also in the classical LLE algorithm, there is no guarantee that the neighborhood of a point in class 1 will consist of points in class 1 itself. For example, in Figure 2.1, the neighborhoods of points x and x'' contain more points belonging to class 2 than points belonging to their own class (class 1). Thus, x and x'' will be reconstructed from neighbors in class 2 rather than from its own neighbors in the low dimensional space. For this reason, when there are data belonging to two different classes sampled from two different underlying manifolds, the existing algorithm fails to preserve the structure of individual manifolds during manifold learning.

As a solution to this problem, I propose a variation in the neighborhood selection phase of the classical algorithm. In this method, I take into account the class label of the point x_i, when selecting its neighborhood. In other words, the neighborhood of point x_i will consist of points that belong to the same class as x_i. Since I take into account class labels in the selection of a neighborhood, I call this supervised k-nearest neighborhood selection. The proposed neighborhood selection method is illustrated

in Fig. 2.2.

As shown in Fig. 2.2, when finding the k-nearest neighbors of the point x_1 of class 1, in the classical method, points from class 2 will also be considered. Therefore, x_3, which does not belong to class 1 will become a neighbor of x_1, but x_2 and x_4, which belong to class 1 will not become its neighbors. However, in the proposed method, x_3 will no longer be a neighbor of x_1, but x_2 and x_4 will become its neighbors leading to successful reconstruction of manifolds belonging to both class 1 and class 2.



(a) Unsupervised neighborhood selection (b) Supervised neighborhood selection

Figure 2.2: Two forms of neighborhood selection

Definition 2.5. ***Supervised k-nearest neighborhood***

Let C be a data class present in X. The supervised k-nearest neighborhood of a point $x_i \in C \subset X$ is $N_{\varepsilon,k}(x_i)$: if $x_j \in N_{\varepsilon,k}(x_i)$, then $x_j \in C$. ■

LLE constructs a *neighborhood preserving mapping* based on the idea that the same weights that reconstruct a point from its neighbors in high dimensional space should reconstruct its embedded manifold coordinates from its neighbors in the low dimensional space [29]. In other words, the embedding computed by the LLE al-

gorithm is optimized to preserve the local configurations of nearest neighbors. A neighborhood preserving mapping can be defined as follows,

Definition 2.6. ***Neighborhood Preserving Mapping***.
Let X, Y be metric linear spaces and let $N_\varepsilon(x)$ be a neighborhood of point $x \in X$. A neighborhood preserving mapping $f : X \to Y$ maps the neighborhood of x, $N_\varepsilon(x) \subset X$ to a neighborhood of $y = f(x)$ in Y, denoted by $N_{\varepsilon'}(y) \subset Y$, where $\varepsilon' \leq \varepsilon$. That is f is defined by $N_{\varepsilon'}(y) = f(N_\varepsilon(x))$. Thus, $\forall x_i, x_j \in X$, if $\|x_i - x_j\| < \varepsilon$, then $\|y_i - y_j\| < \varepsilon'$ and if $\|x_i - x_j\| > \varepsilon$, then $\|y_i - y_j\| > \varepsilon'$, where $y_i = f(x_i)$ and $y_j = f(x_j)$.

The neighborhood preserving map f is a continuous, one-to-one and onto map. I deduce the following Lemma to show how the supervised neighborhood selection helps to preserve the local geometry of manifolds corresponding to multiple data classes in the data set.

Lemma 2.1. *Let X be a metric linear space, C_i a data class present in X and $f : X \to M$, where f is a neighborhood preserving mapping and $M_i \subset M$ is the low dimensional manifold embedding of points in data class $C_i \subset X$. For any point $x_i \in X$,*

1º if $x_i \in C_i \subset X$, then $N_{\varepsilon,k}(x_i) \subset C_i$.

2º if $x_i \in C_i \subset X$ is mapped to point $y_i \in M_i \subset M$, then $N_{\varepsilon,k}(x_i) \subset C_i \mapsto N_{\varepsilon',k}(y_i) \subset M_i$.

Proof. 1º Immediate from Def. 2.5.

2º From Definition 2.6, $N_{\varepsilon,k}(x_i) \subset X$ maps to $N_{\varepsilon',k}(y_i) \subset M$. From 1, if $x_i \in C_i$, then $N_{\varepsilon,k}(x_i) \subset C_i$. Since all $x' \in C_i \mapsto y' \in M_i$, $\forall y_j \in N_{\varepsilon',k}(y_i), y_j \in M_i$. Thus, $N_{\varepsilon,k}(x_i) \subset C_i \mapsto N_{\varepsilon',k}(y_i) \subset M_i$.

□

Remark The result of Lemma 2.1 proves that through supervised neighborhood selection in the high dimensional space, each point in manifold M_i corresponding to data class C_i can be reconstructed by neighbors belonging to its own class in the low dimensional space and hence preserve the local geometry of M_i. This is true for all data classes $C_i \subset X$ allowing multiple manifold learning.

Next, I prove that the proposed neighborhood selection method preserves the nearness and remoteness between manifolds in the reduced space.

Lemma 2.2. *Let x_i^1 and x_j^2 be any two points in a finite-dimensional linear space X, C_1, C_2 data classes, where $x_i^1 \in C_1 \subset X$ and $x_j^2 \in C_2 \subset X$. If $x_i^1 = x_j^2 \Rightarrow \left\|x_i^1 - x_j^2\right\| = 0$, then $N_{\varepsilon,k}(x_i^1)\ \delta\ N_{\varepsilon,k}(x_j^2)$.*

Proof. If $x_i^1 = x_j^2$, then $N_{\varepsilon,k}(x_i^1) \cap N_{\varepsilon,k}(x_j^1) \neq \emptyset$. Thus, $N_{\varepsilon,k}(x_i^1)\ \delta\ N_{\varepsilon,k}(x_j^2)$. □

Theorem 2.2. *Let $C_1, C_2 \subset X$ such that $X = C_1 \cup C_2$ and let M_1 and M_2 be the manifolds such that $f : C_1 \to M_1$ and $f : C_2 \to M_2$, where f is a neighborhood preserving mapping. If C_1 is near C_2, then M_1 is near M_2.*

Proof. If $C_1\ \delta\ C_2$, then there is at least one point $x_i^1 \in clC_1$ and one point $x_j^2 \in clC_2$ such that $x_i^1 = x_j^2$. Let $y_i = f(x_i^1) \in clM_1$ and $y_j = f(x_j^2) \in clM_2$. From Lemma 2.1, $N_{\varepsilon,k}(x_i^1) \subset clC_1$ maps to $N_{\varepsilon',k}(y_i) \subset clM_1$ and $N_{\varepsilon,k}(x_j^2) \subset clC_2$ maps to $N_{\varepsilon',k}(y_j) \subset clM_2$. From Lemma 2.2, $N_{\varepsilon,k}(x_i^1)\ \delta\ N_{\varepsilon,k}(x_j^2)$. From Definition 2.6, $x_i^1 = x_j^2 \Rightarrow \left\|x_i^1 - x_j^2\right\| < \varepsilon \Rightarrow \|y_i - y_j\| < \varepsilon'$. Thus, $N_{\varepsilon',k}(y_i) \cap N_{\varepsilon',k}(y_j) \neq \emptyset \Rightarrow M_1\ \delta\ M_2$. □

Theorem 2.3. *Let $C_1, C_2 \subset X$ such that $X = C_1 \cup C_2$ and let M_1 and M_2 be the manifolds such that $f : C_1 \to M_1$ and $f : C_2 \to M_2$, where f is a neighborhood preserving mapping. If C_1 is far from C_2, then M_1 is far from M_2.*

Proof. If $C_1\ \not\delta\ C_2$, then $clC_1 \cap clC_2 = \emptyset$. Hence, $\forall x_i^1 \in clC_1$ and $\forall x_j^2 \in clC_2$, $x_i^1 \neq x_i^2$ and $N_{\varepsilon,k}(x_i^1) \cap N_{\varepsilon,k}(x_j^1) = \emptyset \Rightarrow N_{\varepsilon,k}(x_i^1)\ \not\delta\ N_{\varepsilon,k}(x_j^2) \Rightarrow \left\|x_i^1 - x_j^2\right\| > \varepsilon$.

From Lemma 2.1, $N_{\varepsilon,k}(x_i^1) \subset clC_1$ maps to $N_{\varepsilon',k}(y_i) \subset clM_1$ and $N_{\varepsilon,k}(x_j^2) \subset clC_2$ maps to $N_{\varepsilon',k}(y_j) \subset clM_1$, where $y_i = f(x_i^1) \in clM_1$ and $y_j = f(x_j^2) \in clM_2$. From Definition 2.6, $\left\|x_i^1 - x_j^2\right\| > \varepsilon \Rightarrow \|y_i - y_j\| > \varepsilon'$. Thus, $\forall y_i \in clM_1$ and $\forall y_j \in clM_2$, $N_{\varepsilon',k}(y_i) \cap N_{\varepsilon',k}(y_j) = \emptyset \Rightarrow M_1 \not\delta M_2$. □

Remark Theorem 2.2 proves that if two data classes are near each other in the high dimensional space, their corresponding manifolds will be near each other in the low dimensional space as well. Also, Theorem 2.3 proves that if two data classes are far from each other in the high dimensional space, their corresponding manifolds will be far from each other in the low dimensional space as well. That is, the proposed supervised neighborhood selection method does not affect the nearness or remoteness of data belonging to different classes when they are mapped onto the low dimensional space.

The following properties of the proposed multiple manifold learning method can be summarized based on the findings given above,

- If M_i is the manifold corresponding to data class C_i and if a data point $x_i \in C_i \mapsto y_i \in M_i$, the neighbors that reconstruct y_i in the low dimensional space also belong to M_i preserving the local geometry of M_i (from Lemma 2.1).

- The nearness or remoteness between data classes will be preserved while mapping them onto the low dimensional multi-manifold space through the proposed method (from Theorems 2.2 and 2.3).

That is, the original class structure of the data set will be preserved in the low dimensional multi-manifold space as well. Thus, these properties of the proposed method are important in classifying new data in the multi-manifold space.

Finally, I define two distance metrics, which will be useful in the implementation of the proposed method. First, consider the distance from a point to a manifold [81]. This distance measure is important in classifying an unknown data sample in a manifold space.

Definition 2.7. ***Point to Manifold Distance (PMD)***.
Let x be a point in a finite-dimensional vector space V, M a manifold. The distance from the point x to manifold M (denoted by $d(x, M)$) is defined by

$$d(x, M) = \min_{S_i \in M} d(x, S_i) = \min_{S_i \subset M} \min_{y \in S_i} \|x - y\|,$$

where S_i are local linear subspaces in M and $\|\cdot\|$ is the Euclidean distance. ■

Next, it is important to define a distance measure to compare two manifolds in the multi-manifold space. Since the focus of MM-LLE algorithm is on preserving the structure of the individual manifolds belonging to each class, any two manifolds present in the reduced space can be compared by measuring their proximity in the multi-manifold space. To achieve this, the manifold-manifold distance, which is a variant of the Hausdorff metric given in [82], is given in Def. 2.8.

Definition 2.8. ***Manifold-Manifold Distance (MMD)***
Let M_1, M_2 be manifolds.

$$d(M_1, M_2) = \frac{1}{T_{M_1}} \sum_{i=1}^{T_{M_1}} \min_{1 \le j \le T_{M_2}} \|M_1(i) - M_2(j)\|,$$

where T_{M_1} and T_{M_2} are the number of data points of M_1 and M_2 respectively, and $M_1(i)$ is the i^{th} point on the manifold M_1. Since the $d(M_1, M_2)$ is directional, the

distance between M_1 and M_2 will be obtained as shown below.

$$D(M_1, M_2) = \max\{d(M_1, M_2), d((M_2, M_1)\}. \quad \blacksquare$$

2.3.3 Proposed Multi-Manifold LLE Algorithm

In this section, I present the proposed multi-manifold learning algorithm based on LLE. The pseudo code of the algorithm is given in Algorithm 1. The proposed algorithm differs from LLE in several ways. The main steps of the MM-LLE algorithm are as follows.

1. **k-Nearest Neighbor Search**

   MM-LLE uses supervised neighborhood selection in the neighborhood selection phase. During this phase, neighborhood search is performed within individual classes of smaller sizes compared to the size of the whole data set, which reduces the computational complexity of the nearest neighbor search in higher dimensions. It is also important to note that I allow the number of nearest neighbors (k) to vary in each class to learn accurately the manifold corresponding to each data class.

2. **Weight Matrix Construction**

   In contrast to the classical LLE algorithm, weight matrix construction of MM-LLE algorithm is performed at individual class level. This becomes beneficial in terms of computational complexity of the algorithm, which will be discussed in detail in Section 2.3.4.

3. **Manifold-Manifold Nearness Minimization**

   The algorithm runs iteratively by increasing d until the minimum manifold-

Algorithm 1 LLE-Based Multi-Manifold Learning Algorithm

Read data set $X = \{X_1, X_2, \ldots, X_N\}$
for each data class C_q in X **do**
  for each data point x_i in C_q **do**
    Find k_q nearest neighbors $N_{\varepsilon,k_q}(x_i)$ *%k_q can be different for each class C_q*
    Compute the weights w_{ij} that best reconstruct x_i from $N_{\varepsilon,k_q}(x_i)$ by using equation (1.1)
  end for
  Find weight matrix W with weights of each C_q
end for
Find $N \times N$ sparse matrix $M = (I - W)^T(I - W)$
Set $d = 1$ and set $MinDist = 0$
% Run iteratively until the nearness between manifolds belonging to different classes is minimum
% assuming that different classes have feature vectors sufficiently far (not near) from each other
while ($MinDist < \varepsilon_{mmd}$ AND $d < N$) **do**
  Find $d+1$ minimum eigenvectors Y of M and discard the smallest eigenvector

  % Find manifold embedding coordinates Y_q from Y
  Set initialLength = 0
  for each data class C_q in X **do**
    $Y_q = Y((initialLength + 1) : (initialLength + length(X_q)))$
    $initialLength = initialLength + length(X_q)$
  end for
  % Find manifold-manifold distance
  for each Y_q, Y_r in Y **do**
    Find $D(Y_q, Y_r)$ with Definition 2.8
  end for
  % Find minimum distance between manifolds
  Find $MinDist = \min(D(Y_q, Y_r))$
  $d = d + 1$
end while

manifold distance of the multi-manifold space reaches a predefined threshold ε_{mmd} to obtain the most successful manifold embedding during classification. From Theorem 2.2, it is evident that if the data classes in the data set are not near each other, then their manifolds will not be near each other as well. Assuming different classes are not near each other, a successful manifold embedding can be found by finding the embedding, which gives a sufficiently high minimum manifold-manifold distance. Thus, the algorithm finds an optimum embedding by minimizing the nearness of manifolds in the multi-manifold space. The advantage of using minimum manifold-manifold distance to determine the best d and embedding will be discussed later in Section 2.4.2. Manifold nearness minimization can be further divided as follows.

(a) **Partial Eigenvalue Decomposition**

This step is similar to that of LLE, in which the $d+1$ smallest eigenvectors will be found and the smallest eigenvector with eigenvalue closest to zero will be discarded.

(b) **Minimum Manifold-Manifold Distance Calculation**

The manifold-manifold distance will be found by using Definition 2.8. The minimum manifold-manifold distance for a given embedding will be found by finding the manifold-manifold distance between each pair of manifolds in that multi-manifold space.

2.3.4 Computational Complexity of the MM-LLE Algorithm

Computational complexity of an algorithm plays a major role, when it comes to practical applications of the proposed method. For this reason, a detailed analysis of the computational complexity of MM-LLE is given in this section. Assume a D-

dimensional data set of size N and C data classes is reduced to d dimensions and the number of neighbors is given by k. An analysis of the computational complexity of the classical LLE algorithm is given in [83, §3.1]. Based on that, I analyze the computational complexity of MM-LLE.

As given in Section 2.3.3, MM-LLE consists of three main steps. The first phase is the k-nearest neighbor search for which the average cost would be $O[D\,log(k)\,N\,log(N)]$. Since MM-LLE performs k-nearest neighborhood search at individual class level, the average cost would rather be $O[\sum_i^C D\,log(k_i)\,N_i\,log(N_i)]$. Thus, at higher dimensions this modification will be very beneficial in terms of computational complexity. Likewise, at the second step, the cost of weight matrix construction will be $O[\sum_i^C D\,N_i\,K_i^3]$, which is more advantageous than the cost of weight matrix construction $O[D\,N\,K^3]$ in LLE.

The third step of MM-LLE consists of two sub-steps. Out of these, eigenvalue decomposition has a cost of $O[d\,N^2]$, which is similar to LLE. The cost of the second sub-step of minimizing the manifold-manifold nearness would be $O[\sum_{i<j}^C d\,N_i\,N_j]$. The third step runs iteratively until the minimum manifold-manifold distance reaches the predefined threshold. Assuming the algorithm runs for m number of iterations before it reaches the predefined threshold ($1 \leq m \leq N-1$), the computational complexity of the third step is $O[m(d\,N^2 + \sum_{i<j}^C d\,N_i\,N_j)]$.

In summary, the overall computational complexity of the MM-LLE algorithm is

$$O\left[\sum_i^C D\,log(k_i)\,N_i\,log(N_i)\right] + O\left[\sum_i^C D\,N_i\,K_i^3\right] + O\left[m\,d\,N^2 + \sum_{i<j}^C m\,d\,N_i\,N_j\right].$$

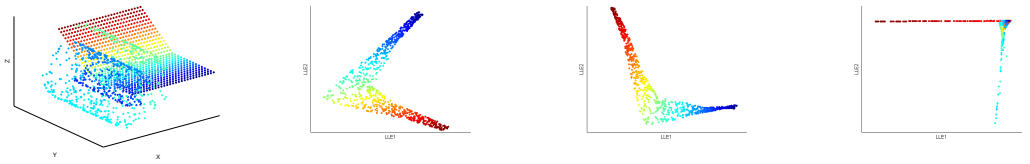
2.4 Experiments and Results

The experiments of the proposed algorithm were conducted with traditional sample test data sets as well as real world image data sets. For comparisons, the two major multi-manifold learning algorithms found in past literature, M-ISOMAP and MMDA were implemented. Moreover, several single manifold learning algorithms such as PCA, ISOMAP, LLE and LE were also implemented for comparisons.

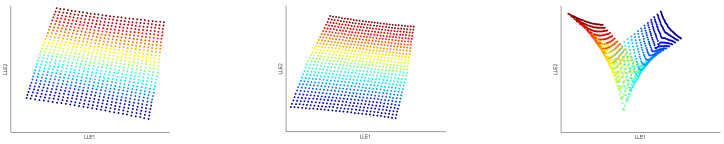
2.4.1 Traditional Sample Data sets

The proposed multi-manifold LLE learning algorithm has been tested on some synthetic multi-manifold data sets generated by using the traditional single manifold data sets. Two synthetic multi-manifold data sets were prepared for the initial experiments. The first multi-manifold data set was prepared by combining the *Swiss Roll* and *Corner Planes* single manifold data sets (Fig. 2.3-(a)) and the second data set was prepared by combining the *Punctured Sphere* and *Twin Peaks* single manifold data sets (Fig. 2.3-(h)). All of these sample data sets are three-dimensional, mapped to two-dimensional spaces by the MM-LLE method.

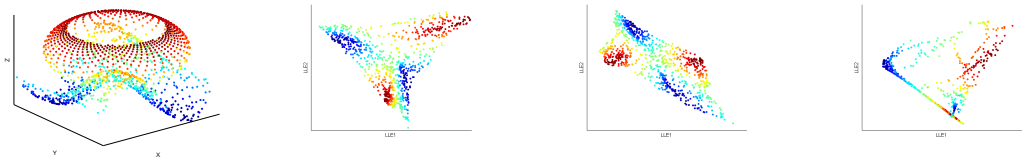
First, the classical LLE algorithm was performed on two synthetic multi-manifold data sets, namely, *Swiss Roll-Corner Planes* and *Punctured Sphere-Twin Peaks*. Next, the proposed MM-LLE algorithm was performed on the two synthetic multi-manifold data sets. Supervised k-nearest neighborhoods with k=8, 4, 4, 8, were used for *Swiss Roll, Corner Planes, Punctured Sphere and Twin Peaks* data sets, respectively. The neighborhood sizes were selected based on the success of learning the underlying manifold for each data set. To assess the multi-manifold learning capability of MM-LLE against LLE, the individual manifolds of single manifold data sets (*Swiss Roll, Corner Planes, Punctured Sphere and Twin Peaks*) were obtained by applying LLE



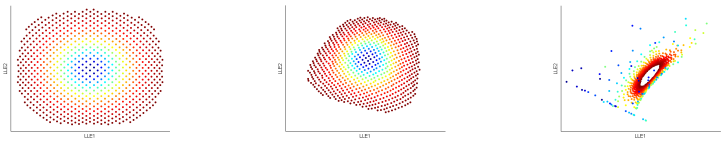
(a) Swiss Roll-Corner Planes (b) Swiss Roll Manifold (c) MM-LLE (d) LLE



(e) Corner Planes Manifold (f) MM-LLE (g) LLE



(h) Twin Peaks-Punctured Sphere (i) Twin Peaks Manifold (j) MM-LLE (k) LLE



(l) Punctured Sphere Manifold (m) MM-LLE (n) LLE

Figure 2.3: Comparison of Multi-Manifold learning capability of MM-LLE and LLE for synthetic multi-manifold test data sets

on each single manifold data set ((b), (e), (i) and (l) in Fig. 2.3).

Fig. 2.3 compares the results obtained on the synthetic multi-manifold data sets for classical LLE algorithm and the proposed MM-LLE algorithm. After the man-

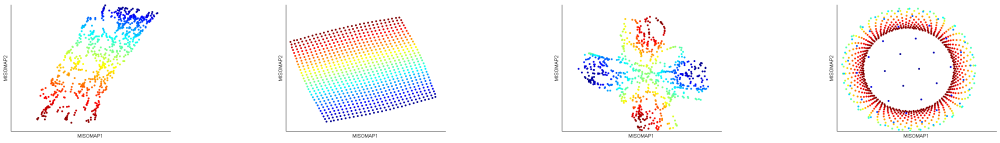
ifolds were learnt by LLE and MM-LLE for the synthetic multi-manifold data sets, manifolds belonging to each class in the synthetic multi-manifold data set were plotted separately to be compared with their corresponding single manifolds given by (b), (e), (i) and (l) in Fig. 2.3. For example, once MM-LLE was applied on the *Swiss Roll-Corner Planes* synthetic multi-manifold data set and when the data belonging to each class (*Swiss Roll and Corner Planes*) were separately plotted in the manifold space, the result is given by (c) and (f) in Figure 2.3. Likewise, the result for LLE is given by (d) and (g) in Fig. 2.3.

The multi-manifold learning capability of MM-LLE was then compared with M-ISOMAP and MMDA algorithms for the same two synthetic multi-manifold data sets. Fig. 2.4 shows the results of these comparisons.

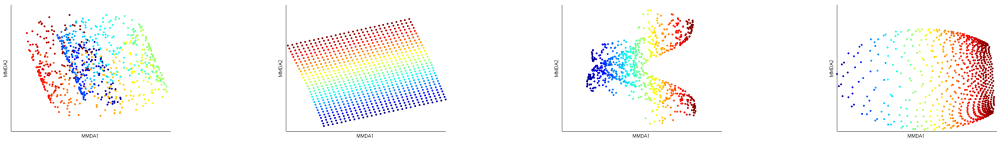
By observing the results given in Fig. 2.3, it can be clearly seen that MM-LLE successfully reconstructs the individual manifolds of the synthetic multi-manifold data sets, while LLE fails to reconstruct the manifolds accurately. For example, in the case of Swiss Roll-Corner Planes synthetic multi-manifold data set, after the manifold learning process, I separate the manifolds belonging to each class (Swiss Roll and Corner Planes) and plot them separately. In Fig. 2.3, (c) and (f) show the manifold learnt by MM-LLE for Swiss Roll and Corner Planes data respectively, while (d) and (g) show the manifold learnt by LLE for the same data. It is evident that (c) and (f) resemble (b) (2D manifold of Swiss Roll) and (e) (2D manifold of Corner Planes) respectively, while (d) and (g) are very much different from (b) and (e), respectively.

The results given in Fig. 2.4 demonstrate that MM-LLE is more successful in learning multiple manifolds present in each synthetic multi-manifold data set compared to M-ISOMAP and MMDA. For example, MM-LLE successfully learns *Twin Peaks* and *Punctured Sphere* manifolds, while M-ISOMAP and MMDA fails to do

(a) MM-LLE - Swiss Roll (b) MM-LLE - Corner Planes (c) MM-LLE - Twin Peaks (d) MM-LLE - Punctured Sphere



(e) M-ISOMAP - Swiss Roll (f) M-ISOMAP - Corner Planes (g) M-ISOMAP - Twin Peaks (h) M-ISOMAP -Punctured Sphere



(i) MMDA - Swiss Roll (j) MMDA - Corner Planes (k) MMDA - Twin Peaks (l) MMDA - Punctured Sphere

Figure 2.4: Comparison of Multi-Manifold learning algorithms for synthetic multi-manifold test data sets

so. All three algorithms learn *Corner Planes* manifold successfully. The classical LLE algorithm itself is incapable of unrolling (learning) the *Swiss Roll* manifold completely. However, the classical ISOMAP algorithm successfully unrolls the *Swiss Roll*. Thus, M-ISOMAP provides a satisfactory result for the *Swiss Roll* manifold, while MM-LLE provides a result similar to the classical LLE and MMDA fails to unroll it at all.

These results provide an indicator that the proposed supervised neighborhood selection method provides a basis for successful multi-manifold learning. Furthermore, since MM-LLE allows varying neighborhood size to optimize the manifold learnt for

each class, the manifold of each data class could be reconstructed successfully. Thus, I next move on to real world data sets to explore the application of the proposed multi-manifold learning algorithm in image pattern recognition.

2.4.2 COIL-20 Database

I experimented with COIL-20 (Columbia Object Image Library), a man-made object database consisting of 1440 images of 20 objects (or 20 classes) [1]. Each class consists of 72 images. Samples from each class of COIL-20 database are shown in Fig. 2.5. The dimensionality of the data set is 16384.

Figure 2.5: Samples of 20 objects in COIL-20 data set [1]

First, the data set was randomly divided into two sets as training set and test set. From each class, an equal percentage of images were selected as training data and the rest were selected as test data. The experiments were conducted under two different training/test ratios, 80/20% and 50/50% and five different manifold learning algorithms PCA, ISOMAP, LLE, LE and MM-LLE.

Once the manifolds were learnt for training data via each manifold learning algorithm, the test samples were projected on to the low dimensional manifold obtained

from training data by using the out-of-sample extension techniques available for each manifold learning algorithm. For an example for both LLE and MM-LLE, test data were projected on to a manifold by considering the low dimensional (manifold) co-ordinates of their k-nearest neighbors and their corresponding weights in the high dimensional space. This is possible in the case of LLE and MM-LLE, because as mentioned before, the neighbors, which reconstruct a point in high dimensional space, reconstruct the point when it is mapped to low dimensional space as well.

After test data were projected onto the manifold space, these new samples were classified based on the k-nearest neighbor (KNN) classification method on the low dimensional manifold space. In the case of MM-LLE, the resulting low dimensional embedding consists of multiple manifolds. Hence, in addition to KNN, it is possible to classify a test sample by finding the distance from the projection of that sample in the manifold space onto each manifold as shown in Figure 2.6. The sample will be classified based on the manifold that is nearest to it. The point-to-manifold distance defined by Definition 2.7 can be used as a distance measure to find the nearest manifold to a given test sample (x_{ts}) as shown below.

Let $d(x_{ts}, M_q) = \min_{M_i \subset M} d(x_{ts}, M_i)$, where M is the multi-manifold space learned by MM-LLE, containing the manifolds $M_1, M_2, ...M_q, ...M_N$ and M_q is the manifold corresponding to class C_q.

$$Class(x_{ts}) = \begin{cases} C_q, & \text{if } d(x_{ts}, M_q) < \varepsilon_{pmd}, \\ undetermined, & otherwise. \end{cases}$$

In the example given in Fig. 2.6, if ε_{pmd} is selected carefully, Y_1 will be classified as class 1, Y_2 will be classified as class 2 and the class of Y_3 will be undetermined (noise), because it lies far from both manifolds.

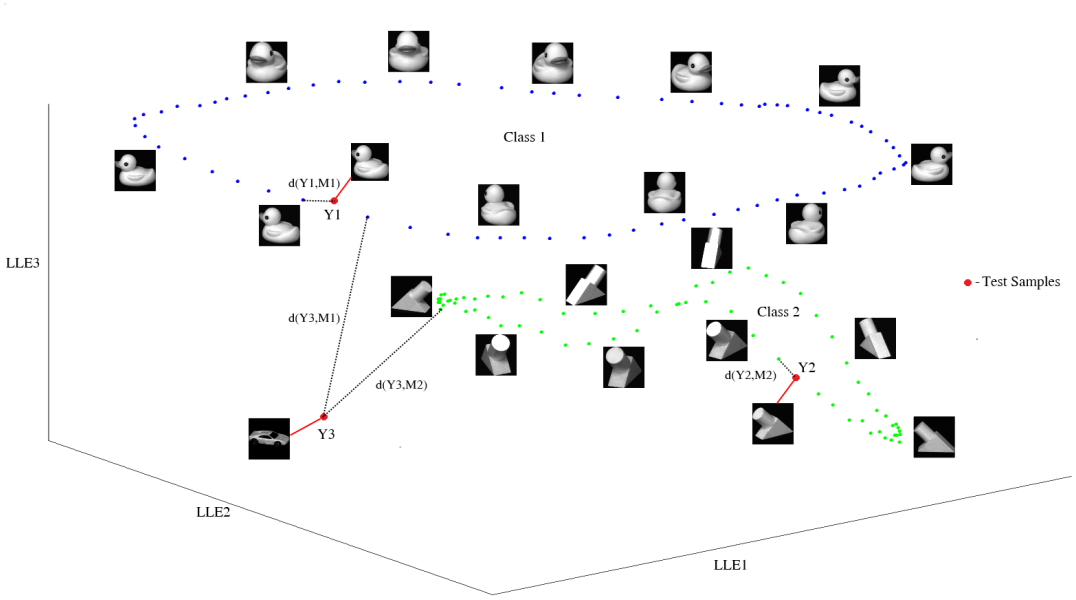
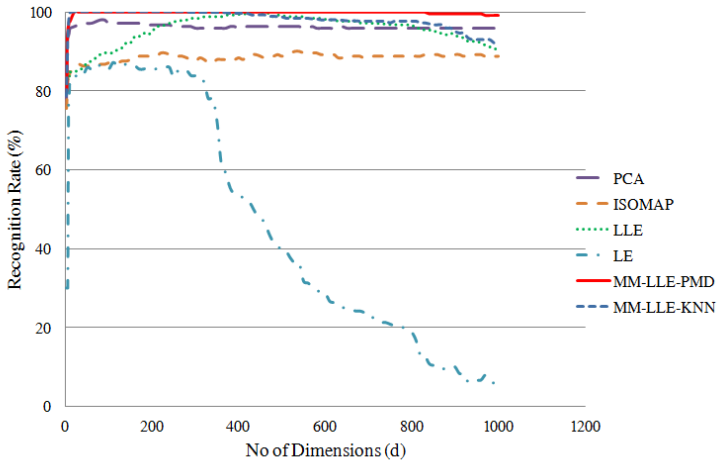


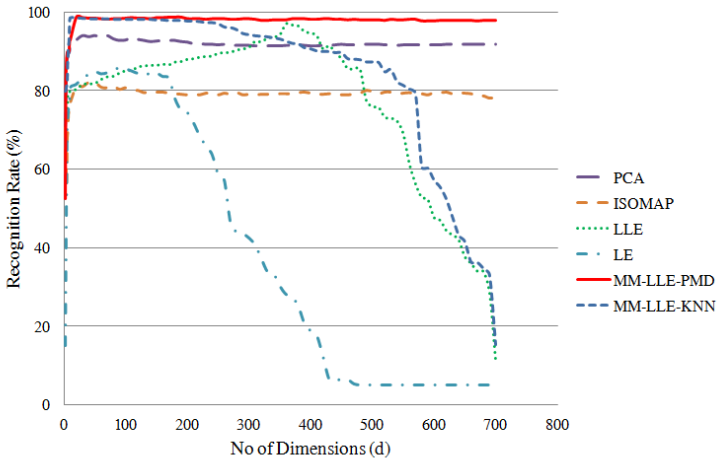
Figure 2.6: Test sample classification based on point to manifold distance

MM-LLE experiments with both KNN (MM-LLE-KNN) and point-to-manifold distance (MM-LLE-PMD), were carried out. The recognition rate was tested by varying the number of embedding (manifold) dimensions (d). The results are given in Fig. 2.7.

Next, the proposed MM-LLE algorithm was compared with the other two multi-manifold learning algorithms, M-ISOMAP and MMDA on the COIL20 data set (Fig. 2.8). There is a practical limitation in applying M-ISOMAP on the wide range of values for d as it was done in the case of comparisons with single manifold learning algorithms (given in Fig. 2.7). In the M-ISOMAP algorithm, first the k-connected components (where k is the number of neighbors) in the data set are found and then the classical ISOMAP algorithm is performed on each of these connected components individually. As mentioned in [70], these connected components can be different from the original class structure of the data set. Thus, in the case of M-ISOMAP, the maximum number of dimensions to which the original space can be reduced to



(a) COIL-20 (Training/Test=80/20%)



(b) COIL-20 (Training/Test=50/50%)

Figure 2.7: Recognition rate by varying the number of manifold dimensions (d) for COIL-20 data set

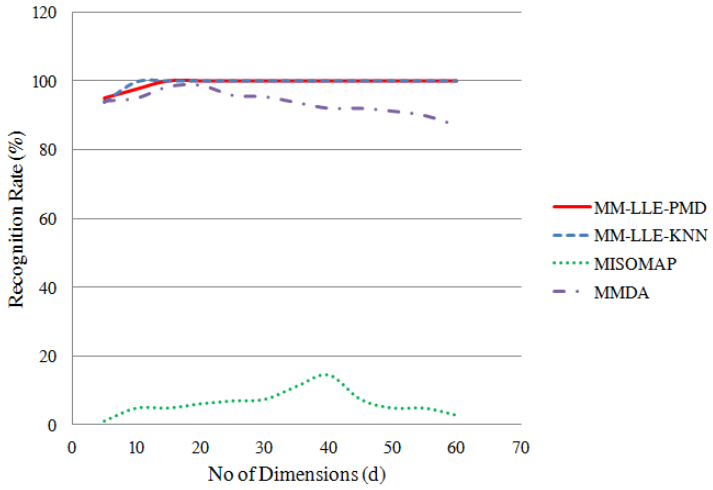
depends on the size of the smallest connected component. In the case of 80/20%, the smallest connected component was 60 and in the case of 50/50%, it was 35. Hence, the recognition rates were obtained for M-ISOMAP, MMDA and MM-LLE for this restricted range of d only. For the comparisons with single manifold learning algorithms, the full range of values for d was covered.

LLE is considered to be a non-parametric algorithm. The only heuristic involved is neighborhood size. However, this neighborhood size (k) plays a major role in learning the underlying manifold accurately. Thus, the classification accuracy of the two variants of MM-LLE was compared with LLE by varying k, to analyze the performance of MM-LLE based on the number of neighbors used in manifold learning (see Fig. 2.9).

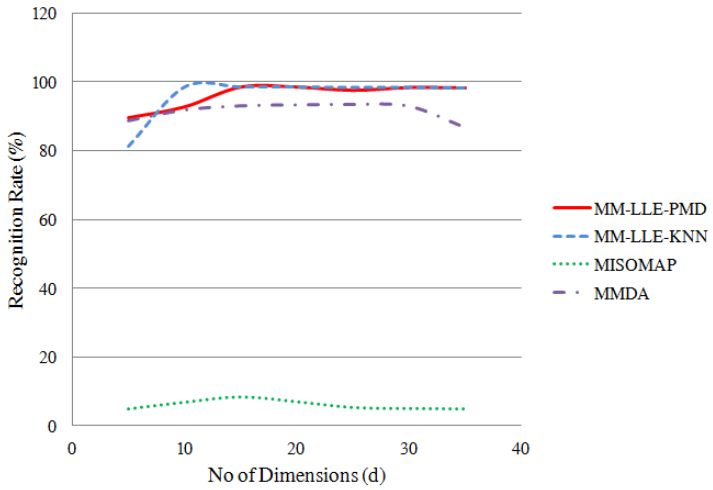
Furthermore, to analyze how closely the manifolds are embedded in the multi-manifold space obtained by the MM-LLE algorithm, the minimum manifold-manifold distance (MMD) together with recognition rate (RR) was measured by varying d for MM-LLE-PMD. Table 2.1 shows the results obtained for $d = 1 : 20$ up to two decimal points.

2.4.3 Face Data Sets

Another set of experiments of the proposed algorithm was conducted on two face data sets, namely, the ORL Database of Faces [84, 85] and the Sheffield (previously UMIST) Face Database [86]. ORL database consists of ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). The Sheffield data set consists of 564 images of 20 individuals (mixed race/gender/appearance). From the Sheffield data set, a



(a) COIL-20 (Training/Test=80/20%)



(b) COIL-20 (Training/Test=50/50%)

Figure 2.8: Recognition rate comparison of multi-manifold learning algorithms for COIL-20 data set

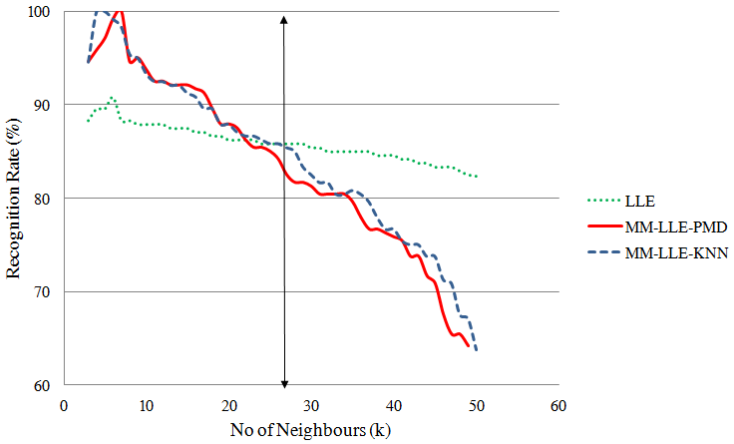


Figure 2.9: Recognition rate by varying the number of neighbors (k) for COIL-20 data set

d	**COIL20 (80/20%)**		**COIL20 (50/50%)**	
	min(MMD)	**Recognition Rate**	**min(MMD)**	**Recognition Rate**
2	1.42E-17	79.92%	4.90E-17	65.28%
3	4.07E-17	81.25%	1.15E-16	65.69%
4	5.18E-17	89.17%	2.75E-16	75.28%
5	8.70E-17	89.17%	2.78E-16	79.86%
6	3.32E-16	90.00%	3.76E-16	80.56%
7	5.71E-16	93.75%	3.91E-16	83.06%
8	6.27E-16	94.17%	8.66E-16	84.17%
9	6.33E-16	94.17%	1.02E-15	85.28%
10	1.34E-15	94.58%	1.22E-15	85.42%
11	2.24E-15	95.00%	1.35E-15	86.53%
12	3.09E-15	96.25%	2.09E-15	86.94%
13	3.66E-15	96.67%	2.16E-15	87.36%
14	4.57E-15	97.58%	2.75E-15	88.47%
15	5.16E-15	97.58%	2.76E-15	89.03%
16	5.21E-15	98.47%	9.26E-15	90.69%
17	6.84E-15	98.61%	2.61E-14	92.22%
18	3.59E-14	99.17%	4.17E-14	95.83%
19	8.94	100.00%	8.94	97.64%
20	12.65	100.00%	12.65	98.611%

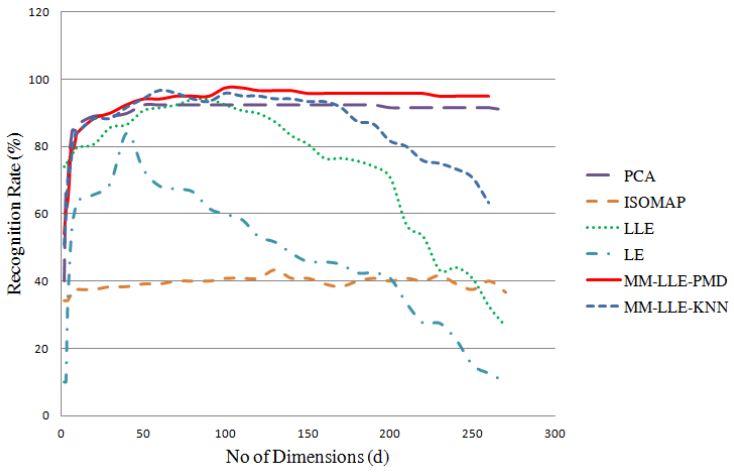
Table 2.1: Minimum MMD and Recognition Rate by varying the number of manifold dimensions (d)

subset of 10 subjects and the first 20 images per each subject were selected for the experiments. In both data sets, the size of a test image is 92x112 pixels, leading to 10304 dimensional feature vectors. A few samples from the ORL face data set and the Sheffield face data set are shown in Fig. 2.10 and Fig. 2.11, respectively.

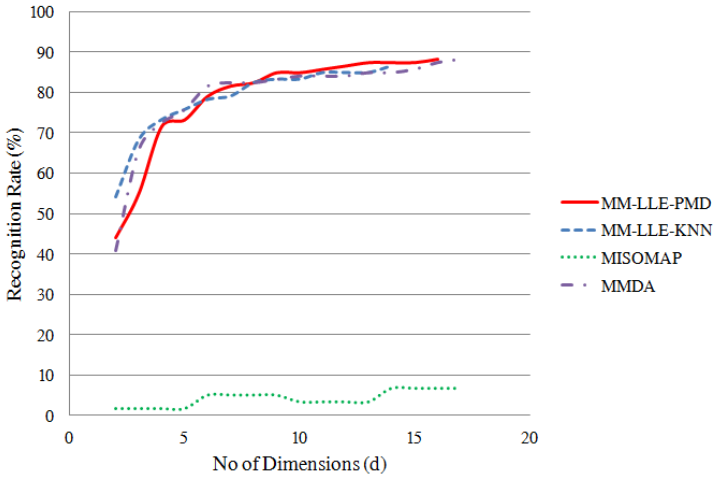
Figure 2.10: Some samples in ORL data set

Figure 2.11: Some samples in Sheffield data set

The experiments for face data sets were conducted following the same method as in the case of COIL20 data set. Experimental results for ORL and Sheffield data sets are illustrated in Fig. 2.12 and Fig. 2.13 respectively. Results of the comparison of MM-LLE with the single manifold learning techniques are shown in sub figure (a) and the results of the comparison of MM-LLE with the multi-manifold learning techniques are shown in sub figure (b) in both Fig. 2.12 and Fig. 2.13. The results of the multi-manifold learning algorithms were limited to a smaller range of d the same as in the case of COIL20 data set due to the limitation in M-ISOMAP algorithm, which was explained in Section 2.4.2.

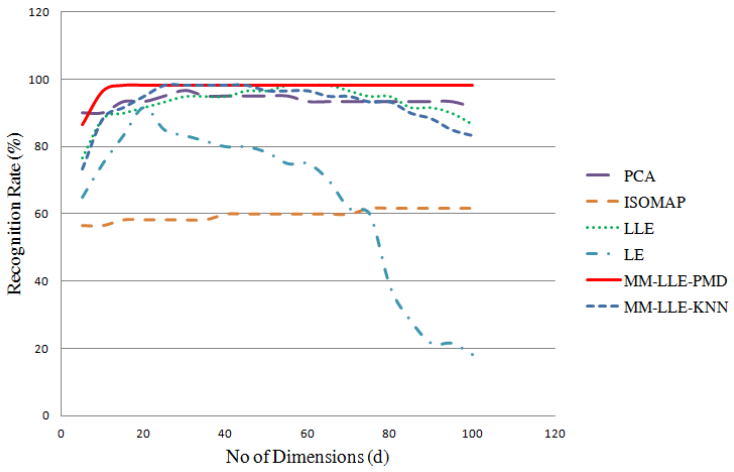


(a) Recognition rate comparison of single manifold learning algorithms with MM-LLE

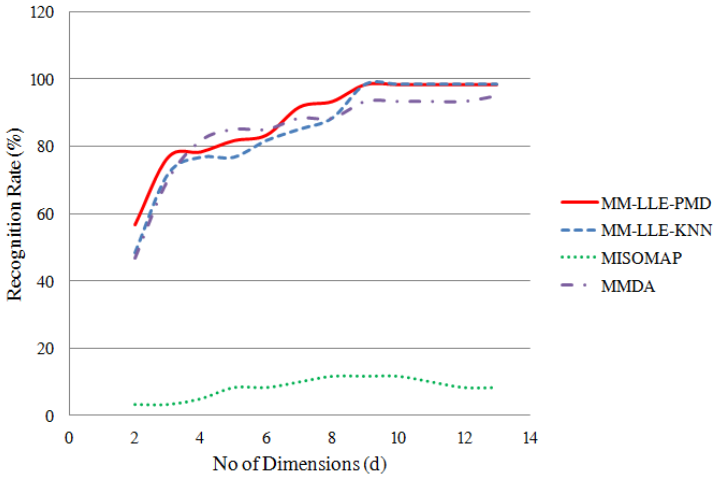


(b) Recognition rate comparison of multi-manifold learning algorithms with MM-LLE

Figure 2.12: Recognition rate comparison for ORL data set



(a) Recognition rate comparison of single manifold learning algorithms with MM-LLE



(b) Recognition rate comparison of multi-manifold learning algorithms with MM-LLE

Figure 2.13: Recognition rate comparison for Sheffield data set

2.4.4 SIMPLIcity Database

Finally, a set of experiments were conducted to explore the manifold-based image classification capability of MM-LLE for Content Based Image Retrieval (CBIR) by using the SIMPLIcity (Semantics-sensitive Integrated Matching for Picture Libraries) database [2,3]. This data set consists of 1000 images belonging to 10 classes. Sample images from each class of the data set are given in Fig. 2.14. Each class of images in the data set was randomly divided in to training and test sets. 70/30%, 50/50% and 30/70% training/test ratios were used for the experiments. Four feature types, namely, Color Correlogram, Color Moment, RGB Color Histogram and Gradient Orientation Histogram were used to represent the images.

In earlier data sets, only single view features (intensity) were used. However, in the case of SIMPLIcity data set, the features used were multiview (color, texture). In these experiments, since my focus was on performing manifold learning by preserving the class structure of data, the different features were concatenated to form a single feature vector for manifold learning. Thus, each image sample was represented by a 646 dimensional feature vector. Other than concatenating the features into a single feature vector, multiview feature learning techniques such as [87, 88] can be used as well.

Figure 2.14: Sample images of SIMPLIcity database [2, 3]

Once the manifold learning phase was complete for the training data, test (query)

data were projected on to the manifold space by using out-of-sample extension and 8-nearest neighbor classification was used to classify the test (query) data samples. The classification accuracies were computed by varying the number of dimensions as it was done for COIL20 database. The results of the two variants of MM-LLE at different training/test ratios were compared with the results of single manifold learning techniques as well as MMDA under the same experimental settings. The maximum classification accuracy achieved by each method rounded up to two decimal places is given in Table 2.2.

During these experiments, it was not possible to apply M-ISOMAP on SIMPLIcity database, since the k-connected components (KCC) algorithm of M-ISOMAP method returned some connected components with a very few members. For example, at the training/test ratio 70/30%, which consists of 700 training images belonging to 10 classes, the KCC algorithm was able to find 15 connected components, when $k = 8$. Out of these 15 connected components, the first component contained 685 samples, while the remaining 14 components contained only 1 or 2 samples in each. The result remained the same for different values of k as well. Hence, it was not possible to proceed with the M-ISOMAP algorithm having just one or two samples in some connected components.

Method	**Training/Test Ratio**		
	70/30%	50/50%	30/70%
PCA	61.33%	61.20%	57.43%
ISOMAP	63.00%	57.20%	46.71%
LLE	61.33%	57.60%	55.43%
LE	59.67%	58.40%	52.43%
MMDA	44.33%	41.2%	38.28%
MM-LLE-KNN	**64.33%**	**62.2%**	**58.43%**
MM-LLE-PMD	62.67%	60.6%	**58.43%**

Table 2.2: Classification accuracy comparison for SIMPLIcity database

2.4.5 Experiments on Computational Complexity of MM-LLE

Some experiments were conducted in order to compare the computational complexity of MM-LLE with other multi-manifold learning algorithms. Table 2.3 shows the execution time of each algorithm run on a Intel Xeon E3-1280 V2 @ 3.60 GHz processor. The execution time of the single manifold learning algorithms ISOMAP, PCA and LLE were recorded in order to be compare with the multi-manifold learning algorithms built upon those, M-ISOMAP, MMDA and MM-LLE, respectively. The execution times were obtained for learning the training set manifold of COIL20 data set at training/test=80/20%, which has 1200 training images of 20 classes with 60 images in each class. The dimensionality of the data set is 16384. If we recall the parameters used in expressing the computational complexity of MM-LLE in Section 2.3.4, the values of the parameters in these experiments would be $N = 1200$, $D = 16384$, $N_i = 60$, $C = 20$, $k = 8$ and $d = 20$.

Method	M-ISOMAP	MMDA	MM-LLE	ISOMAP	PCA	LLE
Exec. Time (sec)	204.499	2.431	**2.332**	7.875	1.340	2.754

Table 2.3: Execution time of multi-manifold learning algorithms

Since these experiments reduce the original data set of $D = 16384$ to a predetermined number of dimensions $d = 20$, the execution time of MM-LLE was obtained for $d = 20$ (one iteration) only. In this setting, MM-LLE completes execution in 2.332 seconds. Next, a set of experiments were conducted to obtain the execution time and the number of iterations of MM-LLE to achieve the predefined threshold $\varepsilon_{mmd} = 2$ given in Algorithm 1. Table 2.4 illustrates the results of these experiments for all the data sets reported in this chapter. For example, for COIL20 (80/20%) and for $\varepsilon_{mmd} = 2$, MM-LLE ran for 20 iterations ($d = 20$) and 27.864 seconds until the minimum manifold-manifold distance reached the predefined ε_{mmd}. At this

$d = 20$, the recognition rate was 100%. Likewise, for all other data sets, the provided embedding (d) resulted in recognition rates or classification accuracies closer to the maximum rates given for the whole range of d in these experiments. In Table 2.4, no. of iterations is denoted by I, execution time is denoted by ET and recognition rate is denoted by RR.

Data Set	**I**	**ET**	N	N_i	C	D	d	k	**RR**
COIL20 (80/20%)	20	27.864 sec	1200	60	20	16384	20	8	100.00%
COIL20 (50/50%)	20	16.394 sec	720	36	20	16384	20	8	98.61%
ORL	40	14.274 sec	280	7	40	10304	40	4	92.50%
Sheffield	10	1.441 sec	140	14	10	10304	10	8	98.33%
Simplicity(70/30%)	10	1.441 sec	700	70	10	646	10	8	61.00%
Simplicity(50/50%)	10	1.364 sec	500	50	10	646	10	8	58.60%
Simplicity(30/70%)	10	0.827 sec	300	30	10	646	10	8	56.57%

Table 2.4: Execution time of MM-LLE for different data sets and predefined $\varepsilon_{mmd} = 2$

2.5 Discussion

The results given in Figure 2.7 indicate that MM-LLE algorithm outperforms the single manifold learning algorithms in terms of the recognition rate under both training/test ratios, 80/20% and 50/50%. At training/test=80/20%, both MM-LLE-KNN and MM-LLE-PMD achieve 100% recognition rate at dimensions as low as 20. Other methods do not reach 100% recognition rate except for LLE, which reaches 100% at $d = 430$. At training/test=50/50%, the maximum recognition rate achieved by MM-LLE-KNN is 98.61% at $d = 10$ and by MM-LLE-PMD is 98.61% at $d = 20$.

In Figure 2.8, it is evident that MM-LLE outperforms other two multi-manifold learning algorithms, M-ISOMAP and MMDA in terms of recognition rate. Out of M-ISOMA and MMDA, MMDA provides better results. As mentioned earlier in Section 2.4.2, M-ISOMAP has a limitation on being applied to a wide range of values for

d. The maximum number of dimensions that M-ISOMAP can reduce the data set to depends on the size of the smallest connected component found by k-connected component (KCC) algorithm used by it. Also, the connected components found by KCC may not be aligned with the class structure of the data set. The two limitations of M-ISOMAP mentioned above may hinder its performance during classification.

Overall, MM-LLE-PMD outperforms MM-LLE-KNN, since MM-LLE-PMD is more consistent in terms of high recognition rates throughout the range for the number of manifold dimensions (d) compared with MM-LLE-KNN and the other manifold learning algorithms considered in this study.

It is evident from Figure 2.7 that for all manifold learning algorithms, recognition rate is dependent on the number of dimensions of the manifold space. The recognition rate increases as d increases and then, after a certain point, it decreases with the further increase of d. Therefore, in order to obtain a good recognition rate, the value for d has to be selected carefully. According to [20], if d is too high, the mapping from high dimensional space to low dimensional space will enhance noise, because of the constraint $\frac{1}{n}YY^T = I$ and if it is too low, distinct parts of data set might be mapped on top of each other. Hence, reducing the data set to the right value of d is crucial for the accuracy of new sample classification on the manifold space.

By analyzing the results given in Table 2.1, it can be clearly seen that the minimum manifold-manifold distance (MMD) increases as d increases. There is a sudden and a significant change in MMD at $d = 19$ and at this d, the recognition rate is also comparatively very high for COIL20 data set at both training/test ratios. We already know that the recognition rate increases as d increases up to some point. Thus, in this chapter, I propose manifold-manifold distance defined by Definition 2.8 as a metric to determine the best multi-manifold embedding or the best d to achieve a good

classification accuracy in practical applications.

Next, if we look at Figure 2.9, it can be clearly seen that for small values of k, MM-LLE provides better recognition rates than the classical LLE algorithm. However, when k is increased beyond 27, the recognition rate of LLE exceeds that of MM-LLE. The reason for this is the number of images belonging to each class in the training data set. This can be explained by analyzing the effect of k on the performance of the local manifold learning algorithms.

In local manifold learning algorithms like LLE and MM-LLE, when k is too small, the mapping fails to preserve the global structure of the manifold and when k is too high, the mapping loses its non-linear character and its performance is similar to PCA, which is globally linear [20]. This explains why the recognition rate for all three algorithms decreases as k increases. Figure 2.9 was produced at training/test=50/50%. Thus, the number of images in each class in training data set is $72/2 = 36$. For MM-LLE, when learning the individual manifolds corresponding to each class, nearest neighbors need to be selected from 36 images, but for LLE, this has to be from the whole data set $36 \times 20(classes) = 720$. Consequently, $k = 27$ (out of 36), which is nearly the size of a single class for MM-LLE, leads to loss of non-linear character in the algorithm. This justifies the rapid reduction of the recognition rate of MM-LLE compared to LLE, when $k > 27$.

By observing the results of the ORL and Sheffield face data sets shown in Figure 2.12 and Figure 2.13 respectively, it is evident that they are almost consistent with the results of COIL20 data set. In the case of face data sets also the two variants of MM-LLE outperform other single manifold as well as multi-manifold learning algorithms. For the ORL data set, MM-LLE-KNN achieves the highest recognition rate of 96.67% at $d = 60$ and MM-LLE-PMD achieves 97.50% at $d = 100$. The highest recognition

rates achieved by PCA, ISOMAP, LLE and LE are 92.50% at $d = 50$, 40.83% at $d = 100$, 94.17% at $d = 80$ and 84.17% at $d = 40$ respectively. For the Sheffield data set, the highest recognition rate achieved by MM-LLE-KNN is 98.33% at $d = 25$ and for MM-LLE-PMD it is 98.33% at $d = 15$. For PCA, ISOMAP, LLE and LE, the highest recognition rates are 96.67% at $d = 30$, 61.67% at $d = 75$, 98.33% at $d = 60$ and 91.67% at $d = 20$ respectively. For the multi-manifold learning algorithms, in the case of the ORL data set, MMDA provides very competitive results at lower dimensions, while M-ISOMAP consistently gives poor results. In the case of the Sheffield data set, MM-LLE outperforms MMDA and M-ISOMAP. Results of M-ISOMAP are not satisfactory for face data sets as well.

It was not possible to apply M-ISOMAP on the SIMPLIcity data set due to the problem of connected components returned as mentioned in Section 2.4.4. Among the other algorithms, MM-LLE-KNN provided the highest classification accuracies at all training/test ratios (see Table 2.2). For example, MM-LLE-KNN achieves 64.33% at training/test=70/30%, which is the highest among all cases. PCA and ISOMAP techniques give competitive results as well. However, although ISOMAP gives good results, when the training set is large, 63.00% at training/test=70/30%, it gives poor results for a smaller training set, 46.71% at training/test=30/70%. The results of the experiments conducted on SIMPLIcity database can be further improved by fine tuning the feature vectors. In this chapter, my focus is on comparing the classification accuracy of MM-LLE with other manifold learning algorithms under the same experimental settings.

When comparing the execution time of each algorithm given in Table 2.3, it is evident that MM-LLE has the lowest execution time compared to the other multi-manifold learning algorithms, M-ISOMAP and MMDA for finding an embedding for a

predefined number of dimensions. M-ISOMAP is the most computationally complex algorithm out of the manifold learning algorithms reported in this chapter. MM-LLE takes 2.332 seconds to execute, while M-ISOMAP and MMDA take 204.499 seconds and 2.431 seconds, respectively. Table 2.4 provides the execution time for different data sets with a predefined threshold for manifold-manifold distance. For example, for COIL20 the algorithm converges after 20 iterations, when $d = 20$ and the execution time is 27.864 seconds. Thus, even with 20 iterations, MM-LLE is about 7 times faster than M-ISOMAP algorithm at one iteration. It can be seen that at the time the MM-LLE algorithm converges, the resulting manifold embedding provides fairly high classification accuracy as well.

2.6 Conclusions

In this chapter, I introduced a multi-manifold learning algorithm based on LLE. Unlike the classical LLE algorithm, the proposed algorithm learns multiple manifolds corresponding to multiple classes in a data set. The proposed algorithm was compared with four other well-known single manifold learning algorithms as well as two multi-manifold learning algorithms. I have shown that the proposed MM-LLE algorithm outperforms the single as well as the other multiple manifold learning algorithms in terms of the recognition rate. In the classification phase, the proposed algorithm provided a basis for experiments using two different classification techniques: k-nearest neighbor and point-to-manifold distance. It was found that point-to-manifold distance-based classification provides better and more consistent recognition rates in most of the cases. The results of the computational complexity through execution time show that MM-LLE runs faster than other algorithms, except for PCA, which is a linear dimensionality reduction technique. Especially, MM-LLE

is much faster than M-ISOMAP.

Also, I have shown that the minimum manifold-manifold distance can be used as a metric to find the optimum d, which provides a high recognition rate. The algorithm runs iteratively by increasing d until it achieves the desired remoteness between the manifolds in the multi-manifold space. Thus, the algorithm is capable of automatically determining an optimum low dimensional embedding. Since the multi-manifold space is only calculated once and new data samples are later projected on to the same space, the overhead on iteratively finding the embedding coordinates by increasing d does not affect the performance of the new data classification process.

Chapter 3

Voronoï Region-Based Adaptive Unsupervised Color Image Segmentation

3.1 Introduction

In this chapter, I introduce a novel image segmentation technique based on spatial and feature space Dirichlet tessellation, which can automatically find the number of clusters and cluster centroids for a given image. Image segmentation plays a major role in many computer vision and pattern recognition applications. According to [36], image segmentation is the process of dividing an image into different regions such that each region is, but the union of any two adjacent regions is not, homogenous. Clustering techniques have been widely used to cluster image pixels based on the similarity of their features (e.g. color, texture, etc.). k-means [89] and Fuzzy C-means (FCM) [90] are two of the most popular clustering techniques used for image segmentation.

In the case of color images, which are complex data sets by nature, determination of the number of pixel clusters and the cluster centroids becomes very challenging. Thus, adaptive unsupervised clustering techniques, which automatically find the number of clusters and the corresponding cluster centroids are vital for successful image segmentation via clustering techniques. For this reason, during the recent past, the focus of many researchers has turned towards adaptive unsupervised clustering techniques for image segmentation. Mean-Shift [91], Ant Colony Fuzzy C-means Hybrid Algorithm (AFHA) [92] and Region Splitting and Merging-Fuzzy C-means Hybrid Algorithm (RFHA) [93] are some of such adaptive unsupervised techniques proposed in the past literature.

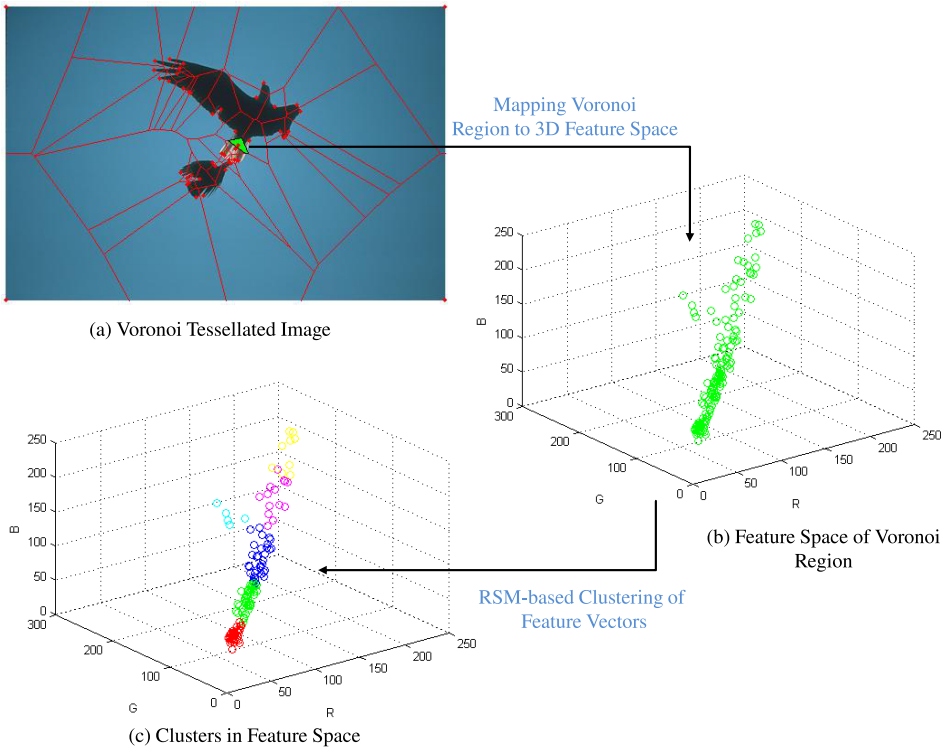


Figure 3.1: Intra-Voronoï region clustering process

In this chapter, I propose a Voronoï region-based adaptive unsupervised algorithm

to automatically find the number of clusters and the cluster centroids in a given image. The main contributions of this chapter are listed as follows: (1) I propose to use a hybrid of spatial and feature space Dirichlet tessellation to adaptively cluster pixels in an image. (2) I propose to perform feature space Dirichlet tessellation within each spatial Voronoï region to cluster pixels into small homogenous clusters. Region splitting and merging will be used to adaptively find the seed points for the feature space Dirichlet tessellation. (3) Finally, I introduce proximal cluster merging based on centroid proximity to merge similar clusters within each pair of spatial Voronoï regions to automatically find the number of clusters and cluster centroids of an image.

The Voronoï region wise clustering in the proposed algorithm reduces the complexity of the segmentation problem significantly, which will be discussed in detail in Section 3.3.2. Furthermore, since the number of possible clusters within a single Voronoï region is usually lower compared to the number of clusters in the whole image, estimating the number of clusters and cluster centroids becomes more efficient and precise.

The rest of the chapter is organized as follows. Section 3.2 presents the work related to the method proposed in this chapter. The methodology of the propsoed Voronoï region-based adaptive unsupervised algorithm is presented in Section 3.3. Section 3.4 discusses the results of experiments conducted on the proposed method and two other adaptive unsupervised cluster-based image segmentation algorithms. Finally, Section 3.5 concludes the research work presented in this chapter.

3.2 Related Work

3.2.1 Clustering-Based Image Segmentation

Clustering is an unsupervised learning process, which does not require class labeled data set as training data to cluster an unknown set of data into clusters. In traditional clustering techniques such as k-means and FCM, the number of clusters has to be predefined to initiate the algorithm. Also, the initial cluster centers (or centroids) are not known. Mean-Shift algorithm introduced by Fukunaga and Hostetler [91] is a non-parametric clustering algorithm, which does not require the number of clusters to be predefined. However, Mean-Shift is very much slower compared to the k-means algorithm. The time complexity of classical k-means is $O(knT)$ while the time complexity of Mean-Shift is $O(Tn^2)$, where k is the number of clusters and n is the total number of data points.

Recently, Yu et al. in [92] proposed an adaptive unsupervised algorithm called Ant Colony Fuzzy C-means Hybrid Algorithm (AFHA), which is a combination of Ant System and Fuzzy C-mean techniques. In [92], Ant System (AS) [94] is used to determine the number of clusters and the cluster centroids. It is said in [92] that AFHA outperforms other state-of-the-art segmentation technique, such as X-means [95], Mean-Shift and Normalized Cut [96]. Another unsupervised adaptive clustering approach for image segmentation named Region Splitting and Merging-Fuzzy C-means Hybrid Algorithm (RFHA) is proposed in [93] by Tan et al. RFHA algorithm uses region splitting and merging scheme to determine the number of clusters and cluster centroids. Both of these algorithms use the Fuzzy C-means algorithm to find the final segmented image. Some other adaptive clustering approaches for image segmentation can be found in [97–102].

The computational complexity of the AFHA algorithm is very high due to the complicated nature of the AS module that it uses. Thus, Yu et al. proposed a modified version of AFHA algorithm named improved AFHA (IAFHA) in [92], which finds the number of clusters and cluster centroids via AS by taking only a small proportion (about 30%) of the total number of pixels into account. This modification in IAFHA significantly reduces the computational complexity of the original AFHA algorithm, but it affects the performance of the algorithm at the same time. RFHA is faster compared to AFHA. However, both of these algorithms suffer from the high computational complexity of the FCM algorithm.

3.2.2 Image Segmentation Evaluation

A formal definition of image segmentation given in [103] is as follows.

Definition 3.1. ***Image Segmentation*** *[103]*

If F is the set of all pixels and $P()$ is a uniformity (homogeneity) predicate defined on group of connected pixels, then segmentation is a partitioning of the set F into a set of connected subsets of regions $(S_1, S_2, \ldots, S_n)$ such that

$$\bigcup_{i=1}^{n} S_i = F \text{ with } S_i \cap S_j = \emptyset, i \neq j \text{ and } P(S_i) = true \ \forall S_i$$
$$\text{and } P(S_i \cup S_j) = false, \text{ when } S_i \text{ is adjacent to } S_j$$

Based on this definition, a segmented image can be evaluated by measuring the homogeneity within each segment and by measuring the overlap between each pair of segments. Haralick et al. proposed four criteria to define a good image segmentation in [104] as follows.

1. Regions should be uniform and homogeneous with respect to some characteristic(s).

2. Adjacent regions should have significant differences with respect to the characteristic on which they are uniform.

3. Region interiors should be simple and without holes.

4. Boundaries should be simple, not ragged, and be spatially accurate.

The first two criteria can be directly derived from the Definition 3.1. Most of the segmentation evaluation methods are based on the first two criteria, jointly called the characteristic criteria. The first criterion measures the intra-region uniformity while the second criterion measures the inter-region disparity. Zhang et al. [105] provides a good summary of unsupervised evaluation methods, which fall under both of these criteria.

3.3 Proposed Method

As explained in Section 1.2, Dirichlet tessellation divides an image into polygonal regions based on the spatial locations (X and Y coordinates) of the seed points (generating points). These polygonal Voronoï regions are much simpler to analyze and process compared to the whole image. Also, since the seed points can be derived from a given image, a tessellation is tailored to the structure of the image itself. Since, these image features are found spatially in an image, I call this process *spatial Dirichlet tessellation*, which divides an image into N Voronoï regions, where N is the number of seed points.

$$V = \{V_1, V_2, \ldots, V_N\}\,.$$

Once a Dirichlet tessellation of a given image is found, the next step is to further divide each Voronoï region V_p until the whole image is grouped into small similar

regions. Thus, next I consider the feature space of each Voronoï region. If each pixel within a given Voronoï region V_p is represented by its d-dimensional feature vector, then each pixel in V_p can be mapped to a point on a d-dimensional feature space.

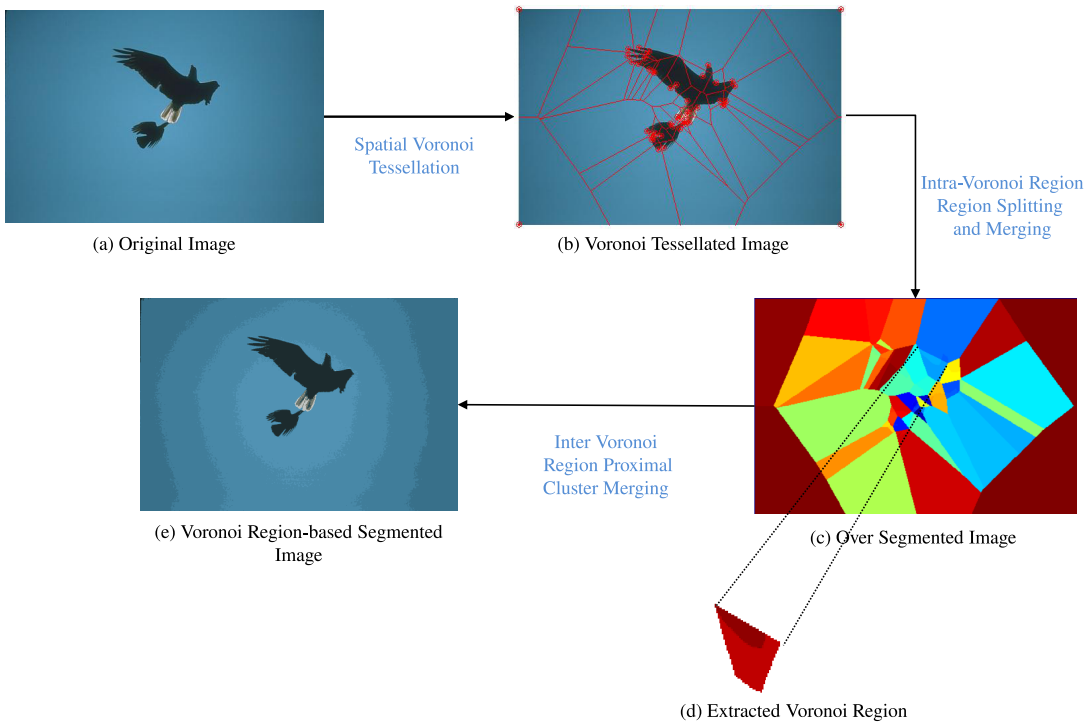


Figure 3.2: Stages of the proposed method

$$V_p \to X_p, \text{ where } X_p \subset \mathbb{R}^d \text{ and } \Phi : \mathbb{R}^2 \to \mathbb{R}^d. \tag{3.1}$$

$$\text{defined by } \Phi(x) = \{\Phi_1(x), \Phi_2(x), \ldots \Phi_d(x)\} = \boldsymbol{x},$$

where Φ is a set of probe functions representing features of a given pixel x. $\Phi(x) = \boldsymbol{x}$ is the feature vector representing the pixel x.

Then, Dirichlet tessellation can be performed on the feature space $\mathbb{R}^d$, where

$d = 3$ in this case as I use R, G and B color channel values of each pixel. In order to find the seed points for the feature space Dirichlet tessellation, I propose to use the region splitting and merging (RSM) scheme given in [93, §2.1]. Through RSM, we can adaptively determine the number of clusters and cluster centroids for each Voronoï region. These cluster centroids can be used as seed points for the feature space Dirichlet tessellation as follows.

First, the set of pixels within V_p will be mapped onto its feature space X_p with the help of equation (3.1). Then, region splitting and merging will be applied on X_p in order to find the adaptive cluster centroids for X_p. Let k_p be the number of clusters and C_p be the cluster centroids found by applying RSM on the set of pixel feature points X_p within a given Voronoï region V_p. After feature space Dirichlet tessellation, X_p can be represented as a set of clusters as given below. Figure 3.1 depicts the process of intra-Voronoï region clustering using RSM and feature space Dirichlet tessellation.

$$X_p = \{s_1, s_2, \ldots, s_{k_p}\}.$$

$$s_i = \left\{ \boldsymbol{x} \in X_p : \|\boldsymbol{x} - \boldsymbol{\mu}_i\| \leq_{\forall \boldsymbol{\mu}_j \in C_p} \|\boldsymbol{x} - \boldsymbol{\mu}_j\| \right\},$$

where $\boldsymbol{\mu}_i$ is the centroid of cluster s_i defined by Definition 3.2.

Once the feature space Dirichlet tessellation is completed for all the Voronoï regions in the image, the image can said to be at an over-segmented state. Let the total set of clusters after the feature space Dirichlet tessellation be $S = \{s_1, s_2, \ldots, s_P\}$. Since, the clustering was performed within each Voronoï region, there can be clusters with similar features belonging to two different Voronoï regions. Thus in the next step, I find such similar clusters and merge them together to find the final number of

clusters and cluster centroids.

In the proposed method, the concept of nearness (proximity) ([80, §1.19] and [106, §1.4]) will be used to find clusters, which are very much similar (proximal) to each other. These proximal clusters can then be merged together to form a single cluster. I extend the notion of nearness of clusters to nearness of their respective centroids to compare two clusters, which is also called the *Centroid* method. In the centroid method, *the resemblance between two clusters is equal to the resemblance between their centroids* [107, §9.5]. Thus, in this method, clusters whose centroids are closest together are merged. The centroid of a merged cluster is the weighted combination of the centroids of the two individual clusters, where the weights are proportional to the number of members in the clusters [108, pp. 373].

Let's start with the definition of a cluster centroid. The centroid of cluster s can be defined as follows,

Definition 3.2. ***Cluster Centroid***

Let s be a cluster, then the centroid $\boldsymbol{\mu}$ of s, can be defined as,

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{\boldsymbol{x}_i \in s} \boldsymbol{x}_i,$$

where $\boldsymbol{x}_i$ are the feature vectors representing the members of s and n is the number of members in s.

In the context of this chapter, the proximal centroids and the proximal clusters will be defined as follows,

Definition 3.3. ***Proximal Centroids***

Let s_i, s_j be two clusters in S and $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ be their cluster centroids respectively. $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ will be proximal to each other (denoted by $\boldsymbol{\mu}_i \ \delta_d \ \boldsymbol{\mu}_j$) if, $d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) < \varepsilon$, where

$d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ is the Manhattan distance (proximity measure) between the two centroids $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ given by,

$$d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \sum_{k=1}^{n} \left| \Phi_k(\mu_i) - \Phi_k(\mu_j) \right|.$$

Definition 3.4. ***Proximal Clusters***

Let s_i, s_j be two clusters in S. The clusters s_i and s_j are called proximal clusters (denoted by $s_i \ \delta_{d_s} \ s_j$) if, $d_s(s_i, s_j) < \varepsilon$, where ε is a predefined threshold and $d_s(s_i, s_j)$ is the proximity measure between the two clusters s_i and s_j given by,

$$d_s(s_i, s_j) = \left| \frac{1}{n_i} \sum_i \frac{1}{n_j} \sum_j (\boldsymbol{x_i} - \boldsymbol{x_j}) \right|,$$

where n_i and n_j are number of members in s_i and s_j respectively and $|\ .\ |$ is the l_1 norm.

Let s_i, s_j be two clusters and $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ be their cluster centroids respectively. Then Lemma 3.1 can be derived to prove that the proximity between two clusters is equal to the proximity between their cluster centroids,

Lemma 3.1. *Let s_i, s_j be two clusters in S and $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ be their cluster centroids respectively. Then,*

$$proximity(s_i, s_j) = proximity(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j).$$

Proof.

$$\begin{aligned} proximity(s_i, s_j) &= d_s(s_i, s_j) \\ &= \left| \tfrac{1}{n_i} \textstyle\sum_i \tfrac{1}{n_j} \sum_j (\boldsymbol{x_i} - \boldsymbol{x_j}) \right| \\ &= \left| \tfrac{1}{n_i n_j} \textstyle\sum_i \sum_j \boldsymbol{x_i} - \tfrac{1}{n_i n_j} \sum_i \sum_j \boldsymbol{x_j} \right| \\ &= \left| \tfrac{1}{n_i} \textstyle\sum_i \boldsymbol{x_i} - \tfrac{1}{n_j} \sum_j \boldsymbol{x_j} \right| \\ &= |\boldsymbol{\mu}_i - \boldsymbol{\mu}_j| = d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) \\ &= proximity(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) \end{aligned}$$

□

Theorem 3.1. *Let s_i, s_j be two clusters in S and μ_i, μ_j be their cluster centroids respectively.*

$$If\ \boldsymbol{\mu}_i\ \delta_d\ \boldsymbol{\mu}_j\ then\ \ s_i\ \delta_{d_s}\ s_j,$$

where d is the Manhattan distance metric leading to metric proximity δ_d and d_s is the proximity measure between two clusters leading to δ_{d_s}.

Proof. From Lemma 3.1, $proximity(s_i, s_j) = proximity(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ and from Definition 3.3, $\boldsymbol{\mu}_i\ \delta_d\ \boldsymbol{\mu}_j \Leftrightarrow d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) < \varepsilon$. Thus, from Definition 3.4, $d_s(s_i, s_j) < \varepsilon \Rightarrow s_i\ \delta_{d_s}\ s_j$. □

Thus, two clusters s_i, s_j can be merged if $d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) < \varepsilon$ and the centroid of the merged cluster can be calculated by using equation (3.2),

$$\boldsymbol{\mu} = \frac{n_i}{n_i + n_j}\boldsymbol{\mu}_i + \frac{n_j}{n_i + n_j}\boldsymbol{\mu}_j. \tag{3.2}$$

where n_i and n_j are the number of elements in s_i and s_j respectively.

In the proposed method, the following merge condition derived based on Theorem 3.1 will be used to iteratively merge proximal clusters belonging to different Voronoï regions.

Merge Condition : Let s_i, s_j be two clusters in S such that $s_i \subset X_i, s_j \subset X_j$, where $V_i \to X_i$ and $V_j \to X_j, V_i \neq Vj$. Let $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ be cluster centroids of s_i and s_j respectively. s_i and s_j will be merged if $d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) < \varepsilon$, where ε is a predefined threshold.

All proximal clusters, which satisfy the above merge condition will be found and merged together iteratively until the number of proximal clusters becomes zero. This process yields the final number of clusters k and the cluster centroids C. Finally, k-means clustering on the feature vectors of the total set of pixels $X = \bigcup_{p=1}^{N} X_p$ of the image will be performed initiated with k and C to find the final segmented image.

3.3.1 Implementation of the Proposed Algorithm

In the implementation of the proposed algorithm, corner points of an image will be used as generating points of the spatial Voronoï tessellation. The proposed algorithm consists of the following major steps:

1. Find the (X,Y) coordinates of the set of the most prominent corners in the image I. The corner detector proposed by Shi and Tomasi [109] available in Matlab was used to extract image corners and the most prominent corners of the image were selected by specifying the minimum accepted quality of corners in Matlab as 0.1. Thus, the number of corners N varies based on the image.

2. Generate the Voronoï diagram $V = \{V_1, V_2, \ldots, V_N\}$ by taking the set of corner points found in step 1 as the seed points.

3. Within each Voronï region V_p,

   (a) Apply RSM on the feature vector set X_p of each Voronoï region V_p to find the number of clusters k_p, cluster centroids C_p

(b) Perform feature space Dirichlet tessellation by taking C_p as the seed points to find the set of clusters $X_p = \{s_1, s_2, \ldots, s_{k_p}\}$.

(c) Let the total set of clusters belonging to all Voronoï regions be $S = \{s_1, s_2, \ldots, s_P\}$.

4. Iteratively merge the inter Voronoï region proximal clusters until the number of proximal clusters becomes zero.

   (a) Find a pair of clusters $s_i, s_j \subset S$, which satisfy the merge condition and $d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = inf_{\forall s_x, s_y \subset S} \left(d(\boldsymbol{\mu_x}, \boldsymbol{\mu_y})\right)$. where $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are the centroids of clusters s_i and s_j respectively.

   (b) Merge s_i and s_j together so that $s_k = s_i \cup s_j$.

   (c) Find new cluster centroid μ_k of cluster s_k using the equation (3.2).

   (d) Repeat steps 4.(a) to 4.(c) until the number of proximal clusters becomes zero.

The pseudo code of the Voronoï-based segmentation algorithm is given in Algorithm 2 and the stages of the proposed method are depicted in Figure 3.2.

3.3.2 Analysis of the Computational Complexity

The main components, which contribute to the computational complexity of the AFHA algorithm are AS and FCM algorithms. The computational complexity of AS is $O(ncdi)$ and the computational complexity of FCM is $O(nc^2di)$ [110], where n is the number of data points, c is the number of clusters, d is the number of dimensions, i is the number of iterations. Thus, the computational complexity of AFHA algorithm can be given as $O(ncdi) + O(nc^2di)$. Similarly in the case of RFHA algorithm, the computational complexities of region splitting and merging phases are

Algorithm 2 Voronoï-based image segmentation algorithm

Input: Image I, Proximity Threshold ε
Output: Voronoï Segmented Image I_{seg}

function Voronoï_segmentation (I, ε)
Find N corners of I, $\{c_1, c_2, \ldots, c_N\}$
%Spatial Dirichlet tessellation
Find $V = \{V_1, V_2, \ldots, V_N\} : d(p, c_i) < d(p, c_j) \forall$ pixels $p \in V_i, i \neq j$
%Feature space Dirichlet tessellation
for each $V_p \subset V$ **do**
  Find set of feature vectors $X_p = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ % $\boldsymbol{x}_i$ = *feature vector of pixel* $p_i \in V_p$
  Find the number of clusters k_p, cluster centroids C_p and set of clusters of X_p using RSM module
  Find $X_p = \{s_1, s_2, \ldots, s_{k_p}\} : d(x, \mu_i) < d(x, \mu_j) \forall\ x \in X_p, i \neq j$
end for
%Inter Voronoï region proximal cluster merging
$S = \{s_1, s_2, \ldots, s_P\} : S = \bigcup_{\forall X_p} s_p \subset X_p$
$minProximity = inf_{\forall s_x, s_y \subset S} (d(\boldsymbol{\mu_x}, \boldsymbol{\mu_y}))$
while $minProximity < \varepsilon$ **do**
  Find the clusters $s_i, s_j \subset S$ that satisfy the merge condition and $\boldsymbol{\mu}_j) = inf_{\forall s_x, s_y \subset S} (d(\boldsymbol{\mu_x}, \boldsymbol{\mu_y}))$
  $s_k = s_i \cup s_j$
  Find new cluster centroid
  $minProximity = inf_{\forall s_x, s_y \subset S} (d(\boldsymbol{\mu_x}, \boldsymbol{\mu_y}))$
end while
$I_{seg} = \{s'_1, s'_2, \ldots, s'_M\}$
return I_{seg}
end function

$O(n^r n^g n^b)$ and $O(idc^2)$ respectively and the complexity of the FCM algorithm is $O(nc^2di)$, where n^r, n^g and n^b are the number of valleys in the histograms of R,G and B channels respectively. Thus, the computational complexity of RFHA algorithm is $O(n^r n^g n^b) + O(idc^2) + O(nc^2di)$.

In the proposed algorithm, the regions splitting and merging happens within each Voronoï region leading to $O(\sum_{k=1}^{N} n_k^r n_k^g n_k^b)$ and $O(i_k dc_k^2)$ complexities in region splitting and merging stages respectively, where N is the number of Voronoï regions. It is important to note that the parameters n_k^r, n_k^g, n_k^b and c_k of the proposed method are much less compared to n^r, n^g, n^b and c of the RFHA algorithm as the region splitting and merging happen within small Voronoï regions in the proposed method rather than on the whole image as in RFHA. Also, the computational complexity of the k-means algorithm is $O(ncdi)$ [110], which is much less compared to the complexity of FCM $O(nc^2di)$. These factors contribute to lower the computational complexity of the proposed method. The computational complexity of the inter Voronoï region proximal cluster merging is $O(idc^2)$. Thus, the computational complexity of the proposed method is $O(\sum_{k=1}^{N} n_k^r n_k^g n_k^b + i_k dc_k^2) + O(idc^2) + O(ncdi)$.

The computational complexities of each algorithm reported in this chapter are summarized in Table 3.1. Computational complexity of Mean-Shift algorithm is given as a reference.

Method	**Computational Complexity**
Mean-Shift	$O(n^2di)$
AFHA	$O(ncdi) + O(nc^2di)$
RFHA	$O(n^r n^g n^b) + O(idc^2) + O(nc^2di)$
Proposed	$O(\sum_{k=1}^{N} n_k^r n_k^g n_k^b + i_k dc_k^2) + O(idc^2) + O(ncdi)$

Table 3.1: Comparison of computational complexity

n = number of data points, c = number of clusters, d = number of dimensions, i

= number of iterations, N = number of Voronoï regions, n^r, n^g and n^b = number of valleys in the histograms of R,G and B channels respectively.

3.4 Experimental Results and Analysis

The proposed algorithm was implemented in Matlab and its results were compared with the results of Matlab implementations of AFHA and RFHA algorithms. For the experiments, the latest version of the Berkeley Segmentation Dataset and Benchmark (BSDS500) [4, 5] was selected. Comparison of results of each algorithm for some sample images from BSDS500 data set is given in Figures 3.3 and 3.4. During the implementation of the algorithm, ε is set to 71, which is said to be effective in detecting perceptually near clusters as given in [93].

3.4.1 Qualitative Analysis of the Segmentation Results

By observing Figures 3.3 and 3.4, it is evident that the proposed method outperforms the other two methods in terms of the segmentation results for the given sample images. For example, for the Coral image in Figure 3.3, both AFHA and RFHA result in non-uniform background (so many isolated pixels in the background) while the proposed method preserves the homogeneity of the segment representing the background. This fact can be clearly seen by zooming in the segments given in false color images (row 2 of Coral image results). *False color image* in the context of this thesis refers to an image depicted in colors other than the real or original colors of that image. False color representation of an image is often helpful in identifying objects or features in that image. Also, both AFHA and RFHA result in some misclassified pixels in the background and on the body of the horses in the Horses image. For the Bird image, AFHA produces an under-segmented image while RFHA produces a

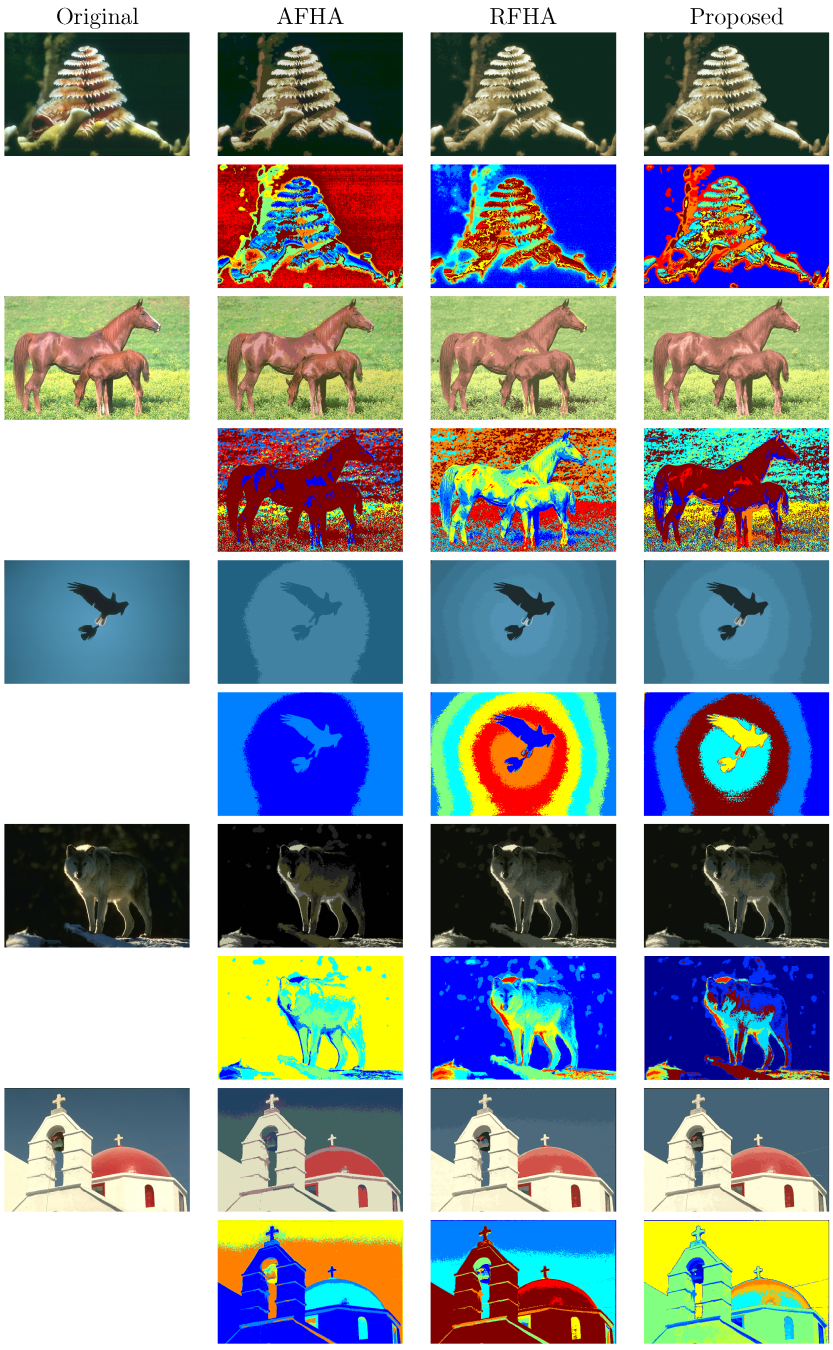


Figure 3.3: Comparison of results for sample images set 1 from BSD500 data set [4,5]

segmented image with too many segments in the region of the sky (over-segmented image). The proposed method provides the best segmentation result for both these

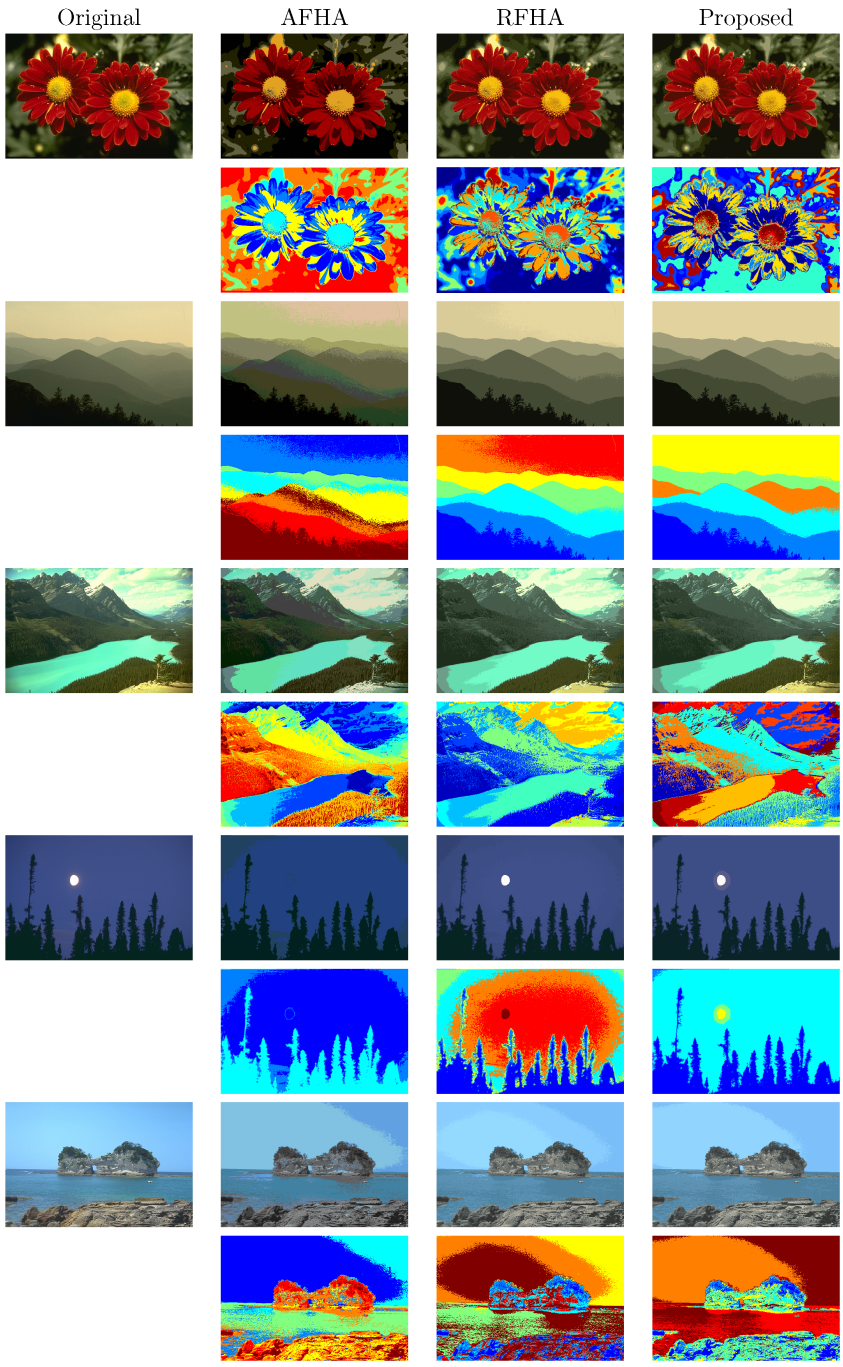


Figure 3.4: Comparison of results for sample images set 2 from BSD500 data set [4,5]

images.

AFHA fails to provide a satisfactory segmentation for the Wolf image as some of

the pixels on the Wolf's body are misclassified. The result of RFHA and the proposed method are almost the same except for the fact that the RFHA segmented image has more noise in the background. The segmentation results of the proposed method and the RFHA algorithm are very much alike for the River image and Flowers image given in Figure 3.4. However, the segmentation of the tree line at the bottom right hand side of the River image segmented by the proposed method is clearer compared to the RFHA segmented image. Furthermore, the proposed method provides the best segmentation result for the Moon image. Both AFHA and RFHA result in over-segmentation of the sky area and AFHA fails to segment the moon in the Moon image.

Similarly, both AFHA and RFHA result in over-segmentation of the sky region in the Church image and Mountains image and RFHA results in over-segmentation of the sky region in the Sea image. Also, AFHA results in misclassified pixels in the segmented Mountains, Church and Sea images. The proposed method provides more uniform segmentation results for all three images.

By observing the results given in Figures 3.3 and 3.4, it is evident that the proposed method is more robust in segmenting large homogenous regions such as the sky region in all sample images. Segmentation results of AFHA mostly include under-segmented images and they commonly contain misclassified pixels while the segmentation results of RFHA are mostly over-segmented. These facts become evident, when comparing the segmentation results given in false color in Figures 3.3 and 3.4.

Furthermore, the segmentation results of the proposed method can be further improved by varying the predefined threshold ε depending on the application. For example, Figure 3.5 depicts how the segmentation result of the sample image 2 can be improved by increasing the ε from 71 to 150. It is important to note that similar

change of the predefined threshold (dc) in RFHA does not result in such improved segmentation, when the predefined threshold is increased to 150. RFHA results in an under-segmented image for $dc = 150$.

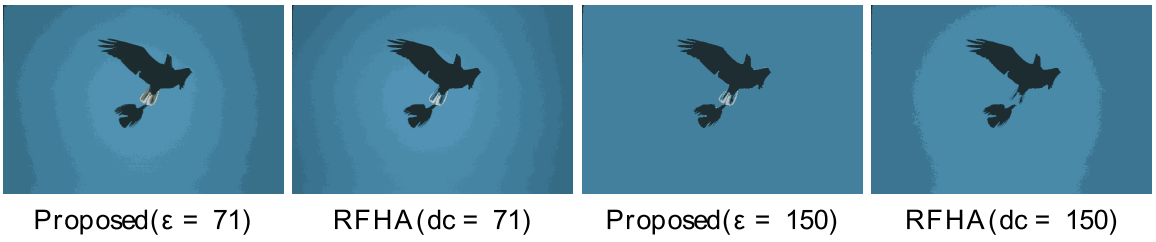


Figure 3.5: Comparison of results for variations of ε

3.4.2 Quantitative Analysis of the Segmentation Results

There are multiple benchmarks reported in the past literature to evaluate the image segmentation results. Zhang et al. [105] broadly categorize these evaluation methods into supervised evaluation methods, which evaluate segmentation algorithms by comparing the resulting segmented image against a manually segmented reference image or ground truth and unsupervised evaluation methods, which evaluate a segmented image based on how well it matches a broad set of characteristics of segmented images as desired by humans. Supervised evaluation is subjective and time consuming while the unsupervised evaluation is quantitative and objective. In this chapter, I will be using the unsupervised evaluation methods.

I first started with the *Mean Squared Error (MSE)*, which is one of the most fundamental benchmarks used to evaluate cluster quality. MSE can be calculated by using the equation (3.3). The number of clusters and cluster quality for the ten sample images given in Figures 3.3 and 3.4 are reported in Table 3.2.

$$MSE = \frac{1}{N} \sum_{j=1}^{M} \sum_{i \in S_i} \|x_i - c_j\|^2 \tag{3.3}$$

where N is the total number of pixels in the images, M is the number of clusters produced during the clustering process, S_j is the set of pixels belonging to j^{th} cluster, c_j is the feature vector of the j^{th} cluster centroid and x_i is the feature vector of the i^{th} pixel belonging to j^{th} cluster. Thus, MSE measures the average deviation of the pixels from the cluster centroids.

Image	No. of Clusters			MSE (*1.0e+3)		
	AFHA	RFHA	Proposed	AFHA	RFHA	Proposed
Coral	9	10	9	1.1727	**0.2886**	0.2964
Horses	23	8	13	1.0874	0.6335	**0.3755**
Bird	2	8	8	1.3135	**0.0479**	0.0561
Wolf	5	7	7	0.8897	0.1760	**0.1704**
Church	6	11	12	1.6880	**0.1567**	0.1862
Flowers	7	15	18	1.9896	0.2946	**0.2372**
Mountains	12	7	6	1.0085	**0.0943**	0.1156
River	16	11	16	1.2827	0.4535	**0.2911**
Moon	3	8	5	1.2717	**0.0413**	0.0970
Sea	8	11	10	1.4449	**0.2283**	0.2374

Table 3.2: Comparison of no. of clusters and MSE of different algorithms

The results given in Table 3.2 show that the proposed method results in a very low MSE for all the sample images. AFHA results in the highest MSE for all sample images. RFHA results in the lowest MSE for six images namely, Coral, Bird, Church, Mountains, Moon and Sea and the proposed method results in the lowest MSE for four images namely, Horses, Wolf, Flowers and River. Overall, for the 200 images that were used in the experiments, the proposed method results in 47% of the images with the lowest MSE and RFHA results in 53% of the images with the lowest MSE. Thus, with respect to MSE alone, RFHA seems to perform better than the proposed method.

However, MSE alone has not proven to be a very reliable benchmark for evaluation of image segmentation results. There is always a trade-off between preserving details and suppressing noise. If there are too many segments in the segmented image, then the difference between pixels belonging to a cluster and the cluster centroid may be lower leading to a smaller MSE. But, since many small clusters are formed and the number of clusters is large, the segmented image may not be a satisfactory one. Therefore, further analysis with regard to the number of clusters and homogeneity of clusters is essential for successful evaluation of the proposed segmentation algorithm.

Liu and Yang proposed an image segmentation evaluation function $F(I)$ in [111], which penalizes the over-segmentation in segmented images. $F(I)$ can be calculated by using equation (3.4).

$$F(I) = \frac{\sqrt{M}}{1000 \times N} \sum_{j=1}^{M} \frac{e_j^2}{\sqrt{N_j}} \tag{3.4}$$

where I is the image to be segmented, M is the number of segments in the segmented image, N_j is the number of pixels in the j^{th} segment and e_j is the color error of region j. e_j is defined as the sum of the Euclidean distances of the feature vectors between the original image and the segmented image of each pixel region.

The term $\sqrt{M}$ in $F(I)$ penalizes the segmentation which forms too many segments. e_j indicates whether or not a region is assigned an appropriate feature (color). If the resulting image is over-segmented, the color error of each segment may be smaller, but since the number of segments is large, the value of F will be large indicating that the segmentation result is not good. On the other hand, if the resulting image is under-segmented, then the number of segments will be reduced, but the color error of each segment will be large leading to a large F.

A modified version of $F(I)$ named $F'(I)$ was proposed by Borsotti et al. in [112].

Borsotti et al. propose to modify the global penalization measure $\sqrt{M}$ used in $F(I)$ to make it more robust for noisy images. Borsotti et al. also proposed another evaluation function named $Q(I)$ in [112], which is said to be more sensitive to small segmentation differences. $Q(I)$ uses a stronger penalization factor to penalize non-homogeneous regions. The equations to calculate $F'(I)$ and $Q(I)$ are given in equations (3.5) and (3.6).

$$F'(I) = \frac{\sum_{j=1}^{M} e_j^2 \sqrt{\sum_{a=1}^{MaxArea} [S(a)]^{1+(1/a)}}}{(1000 \times N)\sqrt{N_j}} \tag{3.5}$$

$$Q(I) = \frac{1}{1000 \times N} \sqrt{M} \sum_{j=1}^{M} \left[\frac{e_j^2}{1 + \log N_j} + \left(\frac{S(N_j)}{N_j} \right)^2 \right] \tag{3.6}$$

where N is the total number of pixels in image I, M is the number of segments, N_j is the number of pixels in the j^{th} segment, $S(a)$ denotes the number of regions in image I that has an area of exactly a and $MaxArea$ denotes the largest region in the segmented image.

Thus, next $F(I)$, $F'(I)$ and $Q(I)$ were measured in order to perform a more comprehensive evaluation of the proposed method. Since the number of small segments in the segmented images produced by the algorithms reported in this chapter was very low, the experimental results for $F(I)$ and $F'(I)$ were the same. As the number of small regions reduces, $\sqrt{\sum_{a=1}^{MaxArea} [S(a)]^{1+(1/a)}}$ approximates to $\sqrt{M}$. Therefore, only the values of $F'(I)$ and $Q(I)$ will be reported in this section. Table 3.3 provides a comparison of the results for $F'(I)$ and $Q(I)$ evaluation functions for different algorithms.

By observing the results given in Table 3.3, it is evident that the proposed method results in the lowest $F'(I)$ and $Q(I)$ for majority of the sample images. The results of

Image	$F'(I)$			$Q(I)$		
	AFHA	RFHA	Proposed	AFHA	RFHA	Proposed
Coral	0.0280	0.0101	**0.0093**	0.3329	0.0891	**0.0875**
Horses	0.0628	0.0136	**0.0135**	0.5503	0.1657	**0.1332**
Bird	0.0067	**0.0016**	0.0020	0.1514	**0.0144**	0.0200
Wolf	0.0140	0.0052	**0.0048**	0.1775	0.0454	**0.0443**
Church	0.0246	**0.0065**	0.0067	0.3926	**0.0532**	0.0668
Flowers	0.0370	0.0148	**0.0142**	0.4788	0.1124	**0.1077**
Mountains	0.0276	**0.0017**	0.0018	0.3730	**0.0229**	0.0256
River	0.0537	0.0143	**0.0131**	0.5149	0.1442	**0.1175**
Moon	0.0086	**0.0011**	0.0015	0.1982	**0.0113**	0.0246
Sea	0.0274	0.0072	**0.0071**	0.3988	0.0726	**0.0721**

Table 3.3: Comparison of $F'(I)$ and $Q(I)$ evaluation functions of different algorithms

$F'(I)$ and $Q(I)$ are consistent for all sample images. The proposed method results in the lowest $F'(I)$ and $Q(I)$ for Coral, Horses, Wolf, Flowers, River and Sea images while RFHA results in the lowest $F'(I)$ and $Q(I)$ for Bird, Church, Mountains and Moon images. It is important note that the proposed method had higher MSE compared to RFHA for Coral and Sea images, but the $F'(I)$ and $Q(I)$ values for the same images are lower compared to the RFHA method. This is due to the penalization of over-segmentation available in $F'(I)$ and $Q(I)$. AFHA results in the highest values for $F'(I)$ and $Q(I)$ for all the sample images. Overall, the proposed method results in the lowest $F'(I)$ and $Q(I)$ for 63% of the 200 images used in experiments. RFHA results in the lowest $F'(I)$ and $Q(I)$ for only 37% of the 200 images in the data set.

The image segmentation evaluation functions F proposed in [111] and F' and Q proposed in [112] fall under the first evaluation criterion given in [104], which measure the intra-cluster uniformity. It is said in [105] that F, F' and Q are biased towards under-segmentation because they use a weighting factor to penalize against over-segmentation. Also, F, F' and Q do not measure the inter-region disparity (the second criterion in [104]), which is vital for fair evaluation of segmentation results.

Thus, next I use the evaluation function F_{RC} proposed by Rosenberger and Chehdi in [113] to cover both evaluation criteria. F_{RC} has two evaluation functions to measure both the intra-region uniformity and inter-region disparity. The first function $\underline{D}(I^j)$ measures the global intra-region uniformity, which quantifies the homogeneity of each region in the segmented image I^j and the second function $\overline{D}(I^j)$ measures the global inter-region disparity between regions.

$$\underline{D}(I^j) = \frac{1}{M}\sum_{j=1}^{M}\frac{N_j}{N}\underline{D}(R_j) \tag{3.7}$$

$$\overline{D}(I^j) = \frac{1}{M}\sum_{j=1}^{M}\frac{N_j}{N}\overline{D}(R_j) \tag{3.8}$$

M is the number of segments, N_j is the number of pixels in j^{th} segment R_j, N is the total number of pixels in image I_j. According to [105], the $\underline{D}(R_j)$ in the case of color images is computed as the average squared color error of region R_j. The inter-region disparity between two regions is calculated as:

$$D(R_i, R_j) = \frac{|E(R_i) - E(R_j)|}{NG} \tag{3.9}$$

where $E(R_i)$ is the average gray-level in the region R_i and NG is the number of gray levels in the image. In the case color images, I will be using the average color difference between the cluster centers to measure $D(R_i, R_j)$.

The intra-region and inter-region metrics were combined in order to find the F_{RC} in [113] as follows.

$$F_{RC} = F(\underline{D}(I^j), \overline{D}(I^j)) = \frac{\overline{D}(I^j) - \underline{D}(I^j)}{2} \tag{3.10}$$

A comparison of the values for $\underline{D}(I^j)$, $\overline{D}(I^j)$ and F_{RC} evaluation functions of

Image	$\underline{D}(I^j)$			$\overline{D}(I^j)$			F_{RC}		
	AFHA	RFHA	Proposed	AFHA	RFHA	Proposed	AFHA	RFHA	Proposed
Coral	19.3304	**2.2217**	3.9567	28.0100	28.0764	**32.7834**	4.3398	12.9273	**14.4133**
Horses	2.6996	9.5929	**2.3926**	6.2871	**19.1335**	12.0146	1.7938	4.7703	**4.8110**
Bird	323.2662	**0.6789**	1.3421	**48.0000**	14.0926	16.4910	-137.6331	6.7069	**7.5744**
Wolf	66.8444	**2.6871**	3.6571	**50.9932**	36.4489	39.0588	-7.9256	16.8809	**17.7008**
Church	72.7586	**1.4397**	3.8182	**40.7715**	23.9536	21.5476	-15.9936	**11.2569**	8.8647
Flowers	43.7067	1.2160	**0.8677**	**24.7662**	12.3263	11.9862	-9.4702	5.55515	**5.55925**
Mountains	11.0262	**1.8963**	3.6470	19.4786	37.0116	**43.2205**	4.2262	17.5576	**19.7868**
River	5.2477	3.5570	**1.2531**	15.3580	**21.6964**	18.7902	5.0552	**9.0697**	8.76855
Moon	218.7566	**0.8104**	8.5069	35.4797	20.7144	**45.2829**	-91.6385	9.9520	**18.3880**
Sea	37.7216	**1.8485**	2.5008	**23.8559**	17.4988	19.6872	-6.9329	7.8252	**8.5932**

Table 3.4: Comparison of intra-region and inter-region metrics and F_{RC} evaluation functions of different algorithms

the proposed method and AFHA and RFHA algorithms is given in Table 3.4. It is said in [105] that F_{RC} is more balanced with respect to under-segmentation and over-segmentation with only slight or negligible biases one way or the other. This fact becomes evident by observing the results given in Table 3.4. In the results given in Table 3.4, RFHA produces the lowest intra-region uniformity for majority of the sample images and AFHA produces the highest inter-region disparity for the majority of the sample images. However, it is important to note that the proposed method produces the best values for the combined metric F_{RC} for a majority of the sample images, which means the proposed method produced a balanced result that preserves both the intra-region uniformity and inter-region disparity at the same time.

Overall, the proposed method results in the best F_{RC} for 71% of the 200 images used in experiments. RFHA results in the best F_{RC} for 29% of the 200 images in the data set while AFHA results in the worst F_{RC} for all 200 images. Thus, we can conclude that the proposed method outperforms both AFHA and RFHA in terms of the image segmentation quality.

3.4.3 Analysis of the Execution Time

Some experiments were conducted in order to compare the execution times of the proposed method with AFHA and RFHA. Table 3.5 shows the execution time of each algorithm run on an Intel Xeon E3-1280 V2 @ 3.60 GHz processor for the 10 sample images reported in the experiments section. Also, the average execution time per image of each algorithm was measured by segmenting the 200 training images in the BSDS500 data set. The results are given in Table 3.6.

Image	Image Size	Execution Time (seconds)		
		AFHA	RFHA	Proposed
Coral	481×321	38.1181	509.9087	**30.0898**
Horses	481×321	60.8689	**14.7592**	32.8365
Bird	481×321	31.1224	47.4568	**3.7746**
Wolf	481×321	37.9469	64.9920	**11.6247**
Church	481×321	35.5790	27.9902	**9.6971**
Flowers	481×321	45.5225	92.5119	**21.3567**
Mountains	481×321	39.5727	**6.1321**	7.1426
River	481×321	73.1063	40.8520	**20.6950**
Moon	481×321	28.7676	58.6831	**9.2689**
Sea	481×321	46.7545	27.3802	**21.9857**

Table 3.5: Comparison of execution times of different algorithms for sample images

By observing the results given in Tables 3.5 and 3.6, we can conclude that the proposed method outperforms AFHA and RFHA in terms of the computational complexity. The average execution time per image of the proposed method is roughly 22 seconds, which is the lowest compared to the other two methods. The average execution time per image of the AFHA method is roughly 65 seconds, which is the highest and the average execution time of the RFHA method is roughly 52 seconds. Thus, the average execution time per image of the proposed method is reduced by 57.69% compared to the RFHA method. Thus, the proposed method is proven to be more suitable for real-time image segmentation applications.

Method	**Avg. Execution Time (seconds)**
AFHA	65.3978
RFHA	52.3573
Proposed	**21.8905**

Table 3.6: Comparison of average execution time per image for 200 images of BSDS500 data set

3.5 Conclusion

In this chapter, a new adaptive unsupervised algorithm based on Voronoï regions was proposed to solve the image segmentation problem. The proposed algorithm is capable of automatically determining the number of clusters and the cluster centroids in a given set of pixels. In the proposed method, an image is adaptively divided into Voronoï regions through spatial Dirichlet tessellation and then feature space Dirichlet tessellation is performed within each of these spatial Vornoï regions to find intra-Voronoï region clusters. These intra-Voronoï region clusters are merged based on the centroid proximity to find the final number of clusters and cluster centroids. The Voronoï region wise clustering in the proposed approach leads to significant reduction in the computational complexity of the algorithm. Furthermore, since the number of possible clusters within a single Voronoï region is usually lower compared to the number of clusters in the whole image, estimating the number of clusters and cluster centroids becomes more efficient and precise.

The experimental results reported in this chapter confirm that the proposed method outperforms two other adaptive unsupervised cluster-based image segmentation algorithms, AFHA and RFHA in terms of the image segmentation quality based on three different unsupervised image segmentation evaluation benchmarks. Also, the results of the experiments on average execution time per image prove that the proposed method is much faster compared to the other two algorithms reported

in this chapter, which makes the proposed method more suitable for real-time image segmentation applications.

Chapter 4

Multi-Manifold-Based Skin Classifier on Feature Space Voronoï Regions for Skin Segmentation

4.1 Introduction

In this chapter, I introduce a novel skin segmentation technique by using the multi-manifold learning technique introduced in Chapter 2 for feature extraction and the automatic image segmentation technique introduced in Chapter 3 to segment skin candidate regions. Skin segmentation is the classification of an input colored image into skin and non-skin regions based on features such as color and texture. Although there are various skin segmentation techniques proposed in the past literature, [57] observes that no satisfactory solution has been developed so far. This is mainly due to the challenges in automatic skin segmentation of images with varying backgrounds and background regions having color similar to skin. Most of the skin segmentation techniques proposed in the past literature fall under the supervised skin segmentation

techniques, which use a training data set to train the skin classifier. These skin classifiers use color and/or textural features to discriminate between skin and non-skin pixels in an image. Bayesian skin classifier in the RGB color space proposed by Jones and Rehg [114] is a popular example of color-based skin classifier and [115–117] are texture-based skin classifiers.

However, the skin detection rates of some of these texture-based approaches were significantly worse compared to the color-based approaches [115]. Kawulok et al. claim that non-skin regions with color similar to skin color often appear as texture patterns on the Bayesian skin probability maps generated by Jones and Rehg [114] method, which cannot be observed for the real skin regions [118]. Thus, more recent approaches [118, 119] focus on spatial analysis of textural features extracted from Bayesian skin probability maps and they show significant improvement compared to the other methods proposed in the past literature. The state-of-the-art methods given in [118, 119] provide better skin segmentation results with low false alarm rates and they are successful in distinguishing skin from similar color background in most of the cases. However, they fail to distinguish skin regions from similar color background regions when they overlap in the Bayesian skin probability map. Also, since the methods given in [118,119] propagate skinness starting from skin seeds, they may lead to high false positive rates if the seeds are erroneously detected in the background.

In this chapter I propose a novel skin segmentation technique, which introduces several improvements to overcome the drawbacks of the state-of-the-art skin segmentation techniques. The state-of-the-art methods such as [118, 119] focus on the discrimination between textural features of the skin and non-skin classes. The within class variation of non-skin class is very high compared to the skin class and the real world representation of training data set is of an imbalanced nature. I.e. the number

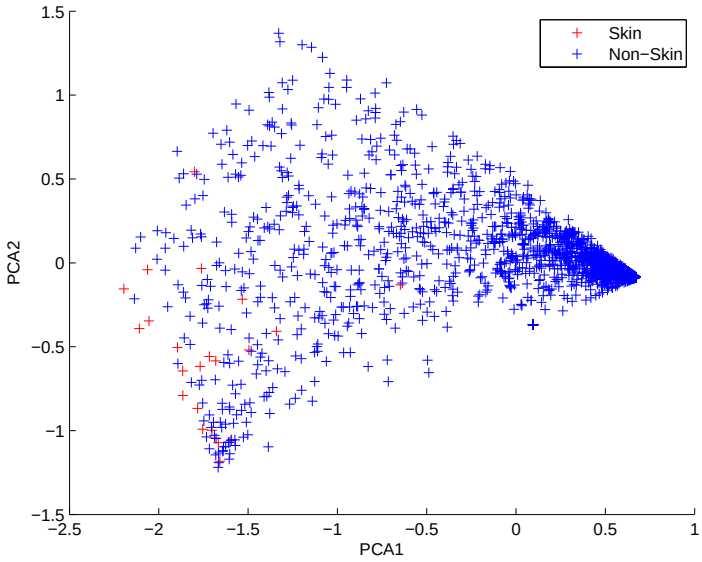


Figure 4.1: Unbalanced class distribution of Skin and Non-Skin classes in training data

of samples in the non-skin class is significantly higher than that in the skin class (see Figure 4.1). Since most of the standard learning algorithms consider a balanced training data set, an unbalanced data set may affect the classifier performance [120–122]. Thus, I believe that the skin classifier performance can be improved if the samples taken to build the non-skin class are restricted to samples with *skin-like* appearance (color) to build a balanced training data set during the classifier training phase. The skin-like non-skin class will be referred to as skin-like class from this point onwards.

It is also important to note that although the feature space of the training data set is high dimensional, skin and skin-like samples vary mainly based on a few features such as color, illumination and texture. Thus, they lie on a low dimensional manifold in the high dimensional feature space. Also, the training data set contains two classes: skin and skin-like, which means data points belonging to each class lie on their own manifold and there is no guarantee that the training data is linear as real

world data sets are often non-linear. Multi-manifold learning non-linearly reduces the dimensionality of data with multiple class structure (skin and skin-like classes in this context) while preserving the intrinsic structure of each data class. Hence, I believe that the skin classification accuracy can be further improved by using multi-manifold learning to build the skin classifier instead of linear techniques such as LDA used in [118, 119].

Furthermore, although skin segmentation can be achieved based on discrimination of textural features extracted from the skin probability map, it becomes challenging, when the background is also detected with high skin probability in the Bayesian skin probability map and when background pixels are spatially proximal to skin pixels. For example, in Figure 4.2, although the hand and background are very much different in color in the original image, it is challenging to distinguish pixels belonging to hand from background pixels solely through spatial analysis of the skin probability map. In this case, both hand and some part of the background fall into skin candidate regions, regions with very high skin probability in the skin probability map.

However, if the skin candidate regions in the original image are segmented based on color information (rightmost image in Figure 4.2) prior to skin classification, it will be possible to distinguish between regions belonging to hand and background during skin segmentation. Thus, I propose to incorporate color-based image segmentation of skin candidate regions through spatial and feature space Dirichlet tessellation proposed in Chapter 3 to solve this problem.

The main contributions of this chapter are listed as follows: (1) I propose to build the skin classifier training data set with skin and skin-like classes rather than skin and non-skin classes in order to obtain a more balanced class distribution in the training data set. (2) I propose to use Voronoï Region-based image segmentation technique

Original Image Bayesian Skin Map Segmented Image

Figure 4.2: A sample image with overlapping skin and background in skin candidate regions

proposed in Chapter 3 to initially segment the skin candidate regions in an image based on color information. Each segment in the resulting segmented image can then be classified as skin or skin-like regions. (3) Finally for skin classification, I propose a multi-manifold-based skin classifier based on the multi-manifold learning algorithm introduced in Chpater 2.

The rest of the chapter is organized as follows. Section 4.2 provides an overview of the work related to the research work presented in this chapter. Section 4.3 discusses the proposed method in detail and elaborates the steps of the proposed algorithm. Experimental evaluation of the proposed method is presented in Section 4.4 and finally the conclusions of the research work presented in this chapter are given in Section 4.5.

4.2 Related Work

4.2.1 Skin Segmentation Techniques

The earlier attempts on skin segmentation focused on color-based skin segmentation. These included simple threshold-based techniques on different color spaces such as RGB, HSV and YCbCr [123–125] as well as model based techniques such as Bayesian,

Gaussian Mixtures and Artificial Neural Networks [114, 126, 127]. Later approaches incorporated textural features such as contrast and homogeneity obtained from a gray-level co-occurrence matrix in an attempt to improve the results of color-based skin segmentation [115]. Although [115] was capable of handling complex background, it is stated in [118] that the skin detection rates of [115] were significantly worse compared to the color-based approaches.

Kawulok et al. further analyzed spatial analysis in skin segmentation [118, 119, 128]. These techniques work on propagating the skinness starting with highly probable skin seeds. A distance transform to propagate the skinness from skin seeds across the image is introduced in Kawulok [128]. In [118] Kawulok et al. perform spatial analysis on texture-based discriminative skin-presence features. It is shown in [118] that the textural differences between skin and non-skin regions on the Bayesian skin probability map provide better discrimination than the textural differences between skin and non-skin regions on gray scale images during skin segmentation.

Thus, to train the skin classifier, they extract basic textural features from the skin probability map for both the skin and non-skin classes and subsequently apply Linear Discriminant Analysis (LDA) [129] on the feature vectors to obtain the discriminative skin-presence feature (DSPF) space. The LDA projection matrix obtained from training data is later used to transform test images onto DSPF space. This process transforms the skin probability map into a DSPF map, which is said to provide better separation between skin and non-skin pixels. Finally, they perform the spatial analysis on DSPF map by using the skinness propagation technique introduced in [128] to detect skin regions in DSPF space.

Kawulok et al. [119] further extended the work done in [118] by introducing self-adaptive seeds during the skin segmentation process. They first obtain the Bayesian

skin probability map to extract the initial set of skin seeds and then the initial seeds are expanded using distance transform to include more skin pixels. A local skin color model is learned from the expanded seeds to find the final set of seeds to propagate skinness and obtain the final skin probability map. The experimental results reported in [118, 119] show that spatial analysis techniques provide better skin segmentation accuracy compared to other skin segmentation techniques.

Overall, textural features are useful in distinguishing skin regions from similar color background regions as skin regions are usually smooth in texture. However, the scale of the textural features should be selected carefully depending on the application as textural features are mostly scale variant. If the textural features are extracted at a scale which is inappropriate to the scale of the skin and background regions in the image, skin segmentation result may be affected.

4.2.2 Bayesian Skin Classifier

Since the Bayesian skin classifier will be used to obtain the initial skin probability maps of the proposed method, let's look at the implementation of the Bayesian skin classifier in detail. At first, based on a training data set, histograms for the skin (C_s) and non-skin (C_{ns}) classes are built. The probability of observing a given color value (v) in the class C_x can be computed as follows,

$$P(v|C_x) = C_x(v)/N_x,$$

where $C_x(v)$ is the number of v-colored pixels in the class x and N_x is the total number of pixels in that class.

For a given image, the probability that a given pixel value belongs to the skin class is computed using the Bayes rule given by eqation 4.1, which will result in the

Bayesian skin probability map of that image.

$$P(C_s|v) = \frac{P(v|C_s)P(C_s)}{P(v|C_s)P(C_s) + P(v|C_{ns})P(C_{ns})}, \tag{4.1}$$

where $P(C_s)$ and $P(C_{ns})$ may be estimated based on the number of pixels in both classes.

4.3 Proposed Method

This section is organized as follows. Section 4.3.1 to Section 4.3.3 presents the major contributions of this chapter in detail. Section 4.3.4 presents the implementation of the proposed skin segmentation algorithm based on the theoretical foundation built in Sections 4.3.1 to 4.3.3. Section 4.3.4 explains the major steps of the proposed skin segmentation algorithm and gives the pseudo code of Multi-Manifold-based skin classification algorithm.

4.3.1 Building a Balanced Training Data Set for the Skin Classifier

In the past literature it is often assumed that all classes are equally present in training and operational data [130]. However, in reality, the training data sets are often imbalanced, which means the number of observations from one class is considerably greater (majority class) than the number of observations from the other class (minority class). According to [120, 121], since most of the standard learning algorithms consider a balanced training data set, the learning algorithms are often biased towards the majority class and this may result in a higher misclassification rate for the minority class instances. Furthermore, Mollineda et al. [122] claim that the class imbalance may produce a deterioration of the classifier performance and Weiss and

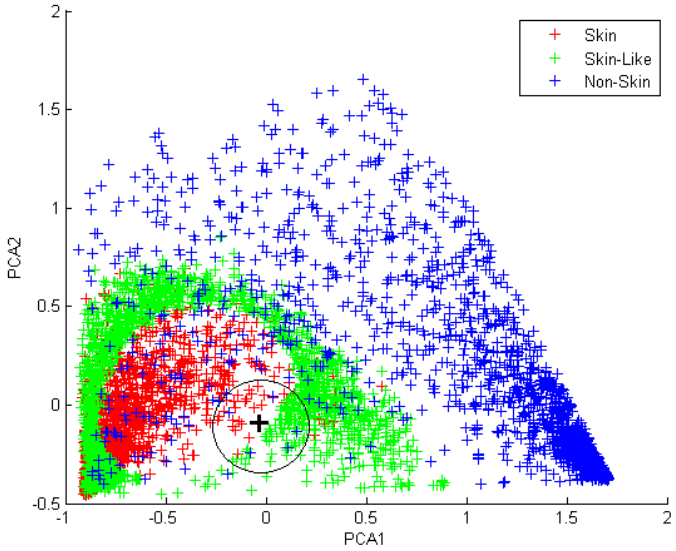


Figure 4.3: Class distribution of skin, skin-like and non-skin classes in modified training data

Provost [131] state that a balanced class distribution is shown to perform well compared to an unbalanced class distribution, when the area under the ROC curve is used as a measure to evaluate the classifier performance.

In this context, the binary skin classifier has to classify unknown data into skin and non-skin classes. Thus, if the distribution is obtained naturally, non-skin class becomes the majority class and skin class, which is the most important class to be learned, becomes the minority class. Kawulok et al. [118] suggest sampling every 15th pixel from every 15th row in each image of the training image data set to obtain the DSPF space. Under this sampling criterion, skin class contributes to less than 1% of the training data set. In order to demonstrate the unbalanced nature of the existing training data set, every 50th pixel on every 50th row was sampled, which increased the skin class to 1.15% of the total training data size. Then the number of dimensions

of the training data set is reduced to 2 by PCA to better visualize the distribution of data. The distribution of training data classes is shown in Figure 4.1. It can be clearly seen that the existing training data set shown in Figure 4.1 is unbalanced.

There are various techniques proposed in the past literature to solve the class imbalance problem. These can be categorized into cost-based approaches and sampling-based approaches [122]. Sampling-based approaches can be further categorized into undersampling methods, which eliminate samples from majority classes to decrease their effect on classifier while keeping the minor class intact, oversampling method, which increase samples from the minor class while keeping the majority class intact and hybrid methods, which combine under and over sampling.

The main objective of this research work is to distinguish between skin and similar color background regions in images. Thus in this chapter, I propose to restrict the samples taken from non-skin class to samples having color similar to skin color (skin-like class) to balance the class distribution and solve the class imbalance problem in the training data set. In the existing methods, skin segmentation ground truths (binary images representing skin pixels with 1 and non-skin pixels with 0) of the training images are used to sample pixels belonging to skin and non-skin classes. For example, if the ground truth image value of the 15th pixel from 15th row of a given training image is 1, that pixel is taken as a skin sample and if the ground truth value is 0, it is taken as a non-skin sample. In this method, I consider Bayesian skin probability maps in addition to the ground truths to construct the training data set. I sample six pixels from the pixels having value 1 in the ground truth image as skin samples and six pixels from the pixels having value 0 in the ground truth image and high skin probability (> 0.7) in the Bayesian skin probability map as skin-like samples.

Moreover, it is important to note that although the method I use to obtain a balanced data set undersamples the majority class, the undersampling is done in an informed manner. I.e. I select only the skin-like samples from the non-skin class. One of the problems with random undersampling is that we cannot control the information eliminated from the majority class during undersampling. Thus, during random undersampling, information significant in defining the decision boundary between skin and non-skin classes may be eliminated [121,132]. For example, Figure 4.3 depicts the distribution of equal sized skin, skin-like and non-skin classes. Random undersampling was used to construct the non-skin class to match the skin class in this experiment and skin-like class was constructed through informed undersampling as explained in the paragraph above. It is evident that an unknown data point (skin-like) marked by a black cross in Figure 4.3 will be falsely classified as skin if only the skin and random undersampled non-skin classes were present. Thus, it is evident that the proposed technique to obtain a balanced training data set is effective in skin classification.

4.3.2 Segmentation of Skin Candidate Region

As it was discussed in Section 4.1, in order to distinguish between overlapping skin and background in skin candidate regions, it is necessary to segment the skin candidate regions based on their color features prior to skin classification. Thus, I propose to use the Voronoï-based image segmentation technique introduced in Chapter 3 to segment skin candidate regions. The details of implementation of the Voronoï-based image segmentation algorithm in the context of skin segmentation are given in Section 4.3.4.

4.3.3 Multi-Manifold-Based Skin Classifier for Skin Classification

Once the skin candidate regions are segmented using the Voronoï-based image segmentation technique proposed in Section 4.3.2, the next step is to classify each segment as skin or skin-like. The balanced training data set built in Section 4.3.1 will be used to train the skin classifier. Each data point in the training data set will be represented by a 13-dimensional feature vector, which will be discussed in detail later in Section 4.4. Although this training data set is high dimensional, it has only a few degrees of freedom. For example, in this context, the pixels in the training data set vary mainly based on color, illumination and texture.

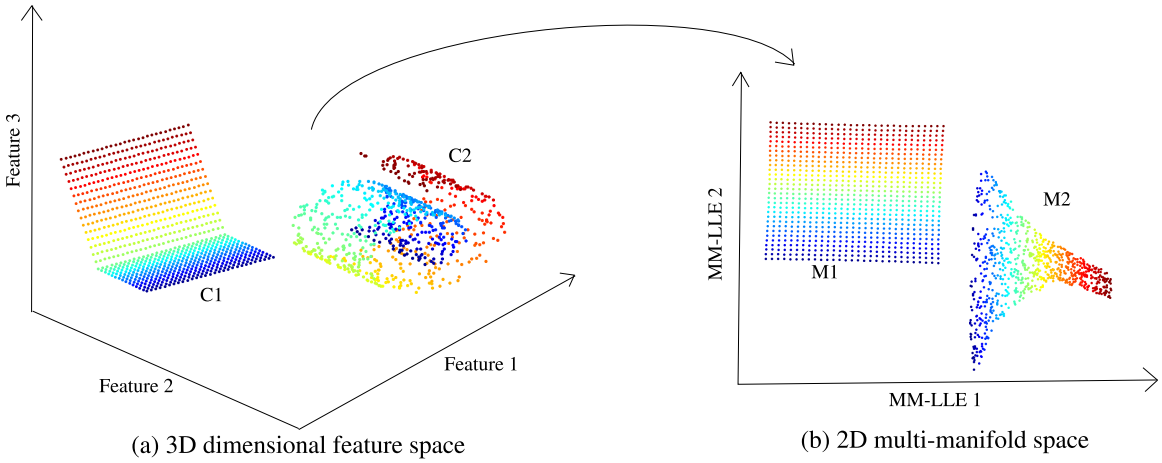


Figure 4.4: Learning multiple manifolds present in the high dimensional feature space

Therefore, the data points in the training data set lie on a low dimensional manifold in the high dimensional feature space. Also, it is important to note that there is no guarantee that the training data set is linear as real world data sets are often non-linear. For example, linear techniques like PCA cannot accurately reduce non-linear structures such as C_2 in Figure 4.4. Hence, non-linear dimensionality reduction techniques such as manifold learning are essential to learn a meaningful representation of training data in such cases. Furthermore, since I have two classes (skin and skin-like)

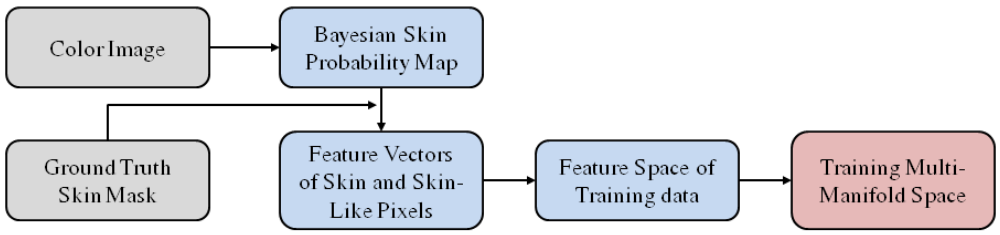
in the training data set, a manifold learning technique which is capable of learning data with a multiple class structure while preserving the intrinsic structure of each data class is important (see Figure 4.4).

Thus, I propose to use multi-manifold learning to learn the manifolds corresponding to skin and skin-like classes in order to obtain a meaningful representation of training data for classification. In this section, I propose the multi-manifold-based skin classifier for skin classification. The multi-manifold learning algorithm (MM-LLE) proposed in Chapter 2 will be used in this chapter to learn manifolds corresponding to skin and skin-like classes for skin classification as it has proven to provide the best classification accuracy among many other single and multiple manifold-based classification techniques as well as linear techniques such as PCA for various multi-class data sets reported in Chapter 2.

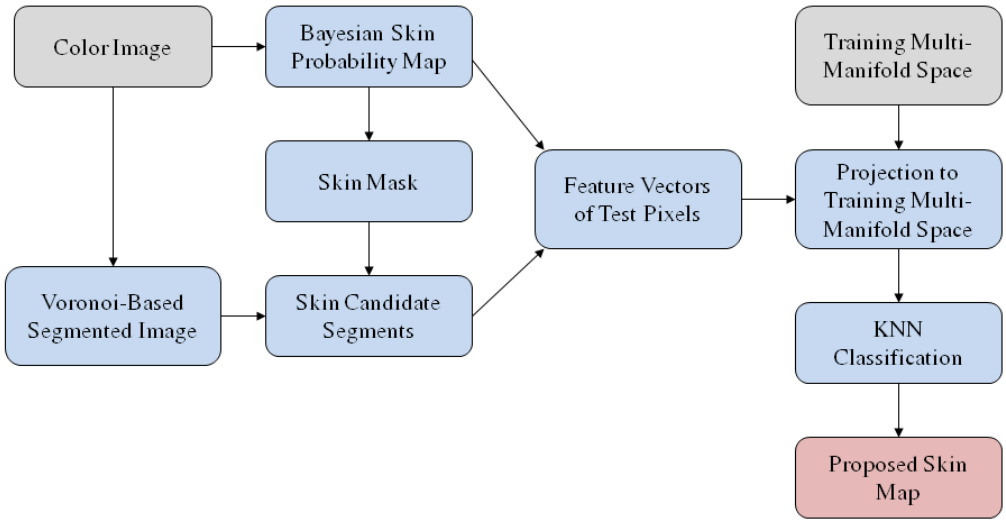
In order to construct the multi-manifold-based skin classifier, I use the result of Theorem 2.3 proven in Chapter 2. Theorem 2.3 proves that if the set of feature vectors of skin (C_s) and skin-like (C_{sl}) classes are far from each other in the feature space, then their corresponding manifolds M_s and M_{sl} are far from each other in the multi-manifold space as well. Thus, skin classification can be performed on the multi-manifold space.

Theorem 2.3 is graphically presented in Figure 4.4. In Figure 4.4, two data classes C_1 and C_2 such that $C_1 \not\delta C_2$ (i.e. $clC_1 \cap clC_2 = \emptyset$) will be reduced to their corresponding manifolds M_1 and M_2 while preserving their intrinsic structure and the remoteness between M_1 and M_2 through multi-manifold learning. From Theorem 2.3, given that the skin and skin-like classes are far from each other with respect to their textural features on the Bayesian skin probability map, their corresponding manifolds on multi-manifold space will be far from each other as well. Thus, in addition to

dimensionality reduction of the feature space, MM-LLE provides a meaningful low dimensional space by preserving the intrinsic structure of skin and skin-like data classes, which is ideal for skin classification.



(a) Training process



(b) Classification process

Figure 4.5: Flowcharts of training and classification processes of the proposed method

During the training phase, the training multi-manifold space will be learned by applying MM-LLE on the textural feature vectors obtained from Bayesian skin probability maps for both the skin and skin-like classes. The same 13-dimensional textural feature vector proposed in [118] will be used to represent data points in this work as well. During the classification phase, an unknown data point will be mapped onto

the multi-manifold space and will be classified as skin or skin-like based on the K-Nearest Neighbor Classification (KNN) technique. Figure 4.5 illustrates flow charts of the training and classification processes of the proposed method. The inputs to the process, intermediate states and the final results are colored in gray, blue and pink respectively.

4.3.4 Implementation of the Proposed Skin Segmentation Algorithm

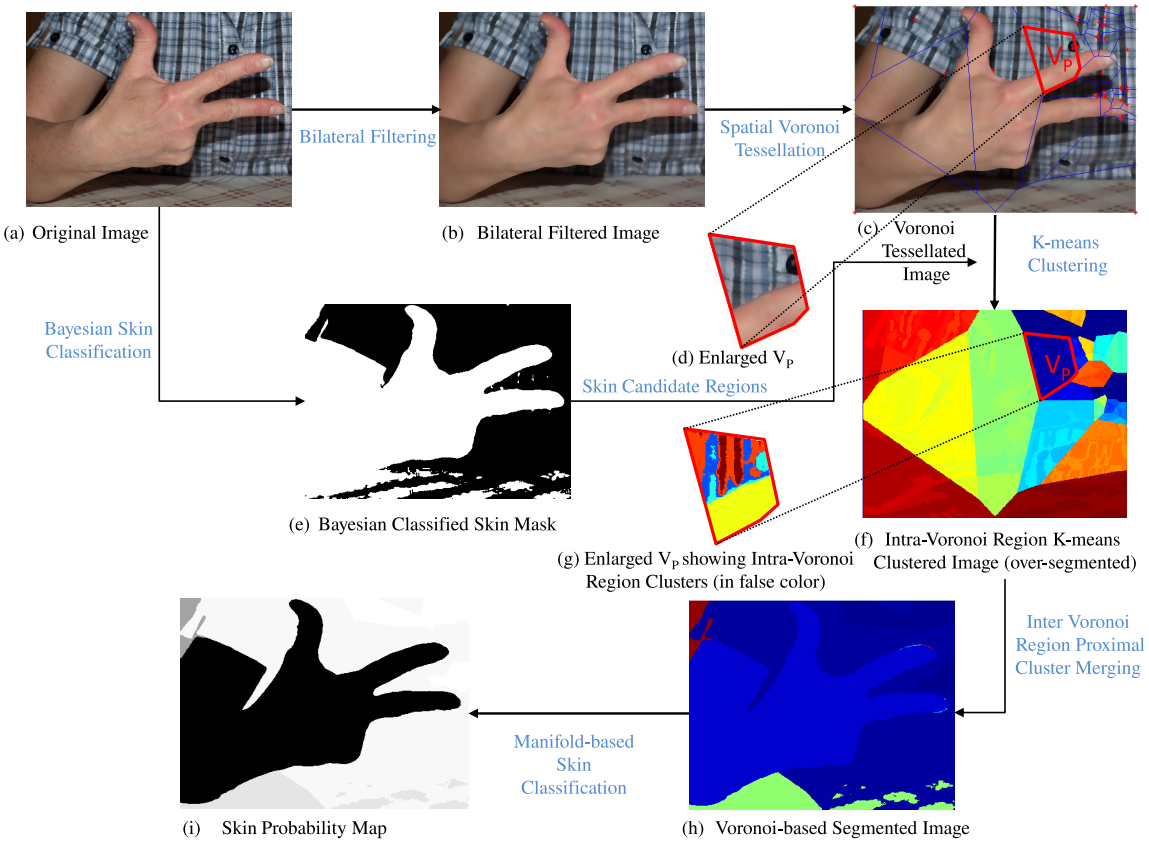


Figure 4.6: Stages of the proposed skin segmentation algorithm

There are six major steps in the implementation of the proposed skin segmentation algorithm. Implementation of Voronoï-based segmentation technique in skin segmentation is given in steps 3 to 5 and the implementation of multi-manifold-based

skin classifier is given in step 6. The major steps of the proposed skin segmentation algorithm with end result of each step are depicted in Figure 4.6.

1. **Pre-processing**

   In the pre-processing step, *Bilateral filtering* [133] was applied to smoothen the image. Bilateral filtering is a simple, non-iterative scheme for edge-preserving smoothing. During Bilateral filtering, *only perceptually similar colors are averaged together, and only perceptually visible edges are preserved* [133]. Thus, it removes unnecessary details in the image, which is beneficial in image segmentation. Filter parameters: window size=5, standard deviation = [3 0.1].

2. **Obtaining Skin Mask by Bayesian Skin Classification**

   In the next step, an initial skin mask (M_{skin}) was obtained by thresholding the result of the Bayesian skin classifier's skin probability map [114]. The Bayesian skin color probability look-up table learned from training data will be used to find this Bayesian skin probability map, which will be discussed in detail in Section 4.4. The skin acceptance threshold in [114] was set to a lower value (0.3), which provides a skin mask with a higher false positive rate (percentage of background pixels misclassified as skin pixels) (see Figure 4.6-(e)). However, this result is desirable as our aim is to reduce the false alarm rates by using the multi-manifold-based skin classification.

3. **Spatial Dirichlet Tessellation**

   In order to generate the spatial Dirichlet tessellation, *corner* points in the image were used as the seed points. The corner detector proposed by Shi and Tomasi [109] available in Matlab was used to extract image corners and the most prominent corners of the image were selected by specifying the minimum

accepted quality of corners in Matlab as 0.1. Thus, the number of corners N varies based on the image. A Dirichlet tessellation was obtained by providing the X, Y coordinates of the corner points in the Bilateral filtered image. Assuming N number of corner points were used as seed points, let the spatial Voronoï regions be $V = \{v_1, v_2, \ldots, v_N\}$.

4. **Feature Space Dirichlet Tessellation of Skin Candidate Regions**

   Next, the feature space of each Voronoï region v_i was considered. The feature space of each Voronoï region v_i is spanned by the color channel values (e.g. RGB or Cb-Cr) of all the pixels belonging to v_i. Only the spatial Voronoï regions having at least one skin candidate pixel were subjected to feature space Dirichlet tessellation. I.e. v_i for which $v_i \cap M_{skin} \neq \emptyset$ were subjected to feature space Dirichlet tessellation. The intermediate result, which is an over-segmented image is shown in false color in Figure 4.6-(f). In Figure 4.6-(f), each pixel of the original image is shown with a false color (e.g. red, green, blue, etc.) representing the pixel cluster that it belongs to. For example, each pixel of the spatial Voronoï region V_p (shown in Figure 4.6-(d)) extracted from the Voronoï tessellated true color image was assigned to five pixel clusters (shown in five colors in Figure 4.6-(g)) during the feature space Dirichlet tessellation. Thus, Figure 4.6-(g) shows the intra-Voronoï region pixel clusters (segmentation result) of the spatial Voronoï region V_p (e.g. pixels belonging to skin region in Figure 4.6-(d) were assigned to a single cluster shown in yellow in Figure 4.6-(g)). Likewise once all the spatial Voronoï regions were subjected to feature space Dirichlet tessellation, the intra-Voronoï region pixel clusters of all spatial Voronoï regions are shown in Figure 4.6-(f) in false color. The boundaries between pixel clusters cannot be seen properly in Figure 4.6-(f) as the number

of clusters is too high at this over-segmented stage.

5. **Inter-Voronoï Region Proximal Cluster Merging**

   Since the previous step resulted in an over-segmented image, in the next step of the algorithm, the clusters within different Voronoï regions v_i and v_j were compared by using the Centroid method and the inter-Voronoï region proximal clusters were merged as explained in Section 3.3 in Chapter 3. The resultant image is the Voronoï-based segmented image (I_{vor}) shown in false colors (see Figure 4.6-(h)). In Figure 4.6-(h), each pixel in the original image is shown with a false color representing the segment that it belongs to.

6. **Multi-Manifold-based Skin Classification**

   In I_{vor}, the skin candidate regions comprise of different segments, which were found by Voronoï-based segmentation in steps 3 to 5. The objective of this step is to classify each of these segments as skin or skin-like. Thus, next a number of sample pixels from each of these segments were obtained. Skin mask M_{skin} was used in this case to sample skin candidate pixels from each segment. I.e. pixels were sampled from the region $s'_i \cap M_{skin}$, where s'_i is a segment in I_{vor}. The number of pixels sampled was varied based on the size of the segment. Approximately 10% of the skin candidate pixels in a segment were sampled during this process. The textural feature vectors of these samples were projected onto the training multi-manifold space (the details are given in Section 4.4) and were classified as skin or skin-like based on KNN classification. The percentage of test pixels classified as skin is assigned as the skin probability of that particular segment. Likewise, the skinness probability of each segment in I_{vor} will be found to generate the skin probability map. The final skin probability map (I_{skin}) is shown in Figure 4.6-(i). In the skin probability map,

Algorithm 3 Multi-Manifold-based skin classification algorithm

Input: Skin Probability Map Map_{skin}, Voronoï Segmented Image I_{vor}
Output: Skin Segmented Image I_{skin}

function multi_manifold_classification (Map_{skin}, I_{vor})
for each $s'_i \subset I_{vor}$ **do**
  $segSize$ = number of elements in s'_i
  Set $nSample = segSize * 0.1$
  Find $s''_i = s'_i \cap M_{skin}$
  Pick random $nSample$ number of pixels from s''_i
  $positives = 0$
  for each pixel r_j **do**
    Find textural feature vector x_{ts} of r_j from Map_{skin}
    %Project x_{ts} onto multi-manifold space Y
    $x_{ts} \longmapsto y_{ts}, y_{ts} \in Y$
    Classify y_{ts} using KNN classification
    if y_{ts} is skin **then**
      Set $positives = positives + 1$
    end if
  end for
  Skin probability $skinProb = positives/nSample$
  Set s'_i in I_{skin} to $skinProb$
end for
return I_{skin}
end function

the higher the skinness probability, the darker the pixel color.

The pseudo code of the multi-manifold-based skin classification algorithm is given in Algorithm 3.

4.4 Experimental Results and Analysis

For the experiments, the proposed algorithm was implemented by using Matlab R2013a. The Face and Skin Detection (FSD) Database [7] was used as training data to obtain the skin and non-skin manifolds. FSD database contains 4000 color images that are diverse in terms of lighting conditions and skin types. The lighting

conditions include indoor lighting and outdoor lighting and the skin types include whitish, brownish, yellowish, and darkish skins. Hence, it is an ideal candidate as the training data to the proposed algorithm.

The FSD database was divided into two equal sized parts and 2000 images were used as the training data and the remaining 2000 images were used as test data. These training images and their skin segmentation ground truths were used to learn the Bayesian skin color probability look-up table, which maps every color value in the color space domain into the skin probability by following the method given in [114]. By using this skin color probability look-up table, the Bayesian skin probability map of each training image was found by mapping the color value of each pixel in the image to the corresponding skin probability. Then, these Bayesian skin probability maps of training images were used to learn the training multi-manifold space.

The training multi-manifold space was obtained as follows: six skin pixels and six skin-like pixels (if available) were obtained from each image in the training set as explained in Section 4.3.1. For each of these pixels, four features namely, median, minimal value, standard deviation and difference between maximum and minimum were computed for three different neighborhood sizes ($5 \times 5, 9 \times 9, 13 \times 13$) on the Bayesian skin probability map of the image as proposed in [118]. The Bayesian skin probability was appended at the end of each feature vector. Thus, each pixel in the training data set can be represented as a 13-dimensional feature vector. This 13-dimensional feature space was reduced to 3-dimensions via MM-LLE as explained in Section 4.3.3 to find the 3-dimensional multi-manifold space containing the skin and non-skin manifolds.

The experiments on the proposed method were conducted on two test data sets. Namely, the benchmark database for hand gesture recognition (HGR) [6] and the

remaining 2000 images from Face and Skin Detection (FSD) Database. In the case of the HGR data set, the first series of HGR (HGR1) was selected for the experiments presented in this chapter as in HGR1 series, the images were captured under uncontrolled background and lighting conditions, which allows the proper evaluation of the proposed algorithm. HGR1 consists of 899 images belonging to 12 individuals and 25 different gestures. For the HGR data set, feature space Dirichlet tessellation using Cb-Cr color channels of YCbCr color space provided better results compared to RGB channels, because the color variation in a given image in the HGR data set is much less compared to the FSD data set. Thus, Cb-Cr color channels were used for the HGR data set and RGB color channels were used for the FSD data set to span the feature space during feature space Dirichlet tessellation.

The scales $5 \times 5, 9 \times 9, 13 \times 13$ were used as neighborhood sizes to extract textural features from the Bayesian skin probability maps in the proposed method, because these neighborhood sizes are said to provide the best results for two test data sets reported in this chapter as per [118]. Since I extract textural features from the Bayesian skin probability map rather than from the original color image and given that the pixels having color similar to skin color are detected with high skin probability, we can say that the skin regions appears as a smooth texture on the Bayesian skin probability map. Also, most of the images in both the HGR and FSD data sets contain fairly large skin regions. Thus, the selected neighborhood sizes work for the reported data sets. However, different combinations of neighborhood sizes may be optimal for other applications as textural features used in this context are scale variant.

The Voronoï-based segmented image and the final skin probability map of the proposed method for some sample images from HGR and FSD data sets are shown in Figure 4.7. By observing the results given in 4.7, it is evident that the proposed

method successfully distinguishes skin pixels from skin-like background pixels even when the skin and background skin candidate regions are overlapped in the skin probability map.

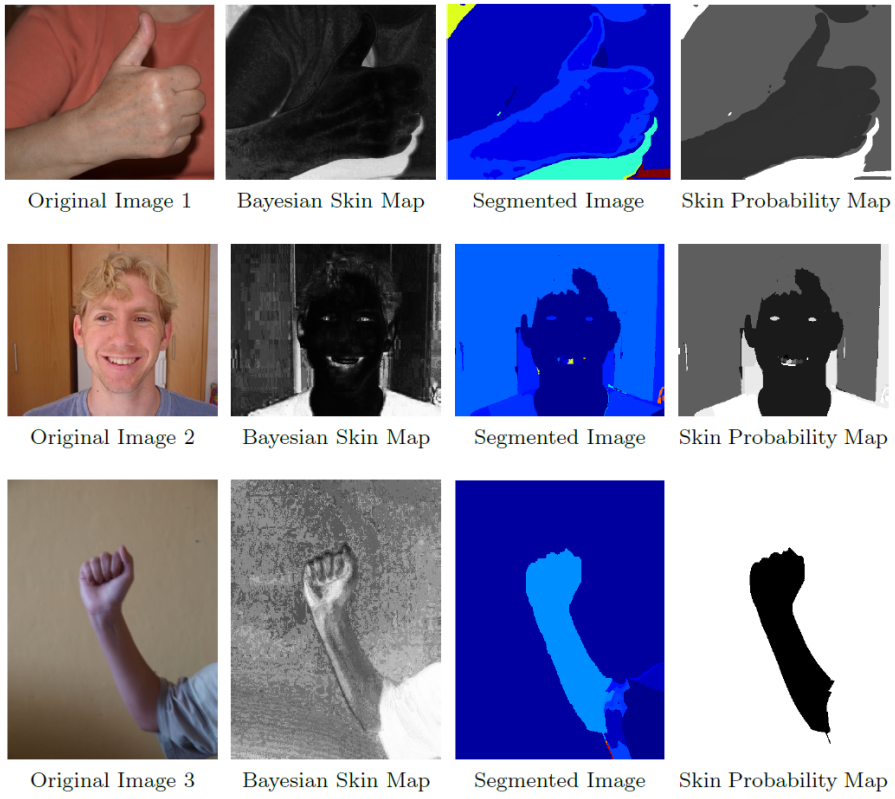


Figure 4.7: Results for some sample images from test data sets [6, 7]

The experimental results presented in this chapter provide comparisons of the proposed method with four other state-of-the-art methods. M.J. Jones and J.M. Rehg's method on Bayesian classifier in the RGB space [114] (Bayes), fast propagation-based skin region segmentation (FPSS) by Kawulok [128], Spatial based skin region segmentation (DSPF) by Kawulok et al. [118] and self-adaptive algorithm for skin regions segmentation (SASS) by Kawulok et al. [119].

Comparisons of the skin probability map obtained from the proposed method with skin probability maps of existing algorithms for some sample images from HGR1 se-

ries are shown in Figure 4.8. These images were selected, because all of them contain backgrounds with skin-like appearance. By observing Figure 4.8, it is evident that the proposed method provides better accuracy in segmenting skin pixels in the given sample images compared to the results of four existing techniques. The existing techniques classify most of the background pixels as skin (false positives), in particularly in images 2, 4 and 5. The results of [128], [118] and [119] are dependent on the seed selection. If seeds are erroneously detected in the background, [128], [118] and [119] propagate those seeds, which results in high false positive rate.

Likewise, experimental results of the proposed algorithm on some sample images from the FSD data set were compared with the four existing algorithms. The results are depicted in Figure 4.9. In the case of the FSD data set also the proposed method provides better accuracy in segmenting skin pixels from the background compared to the existing methods. The existing methods detect most of the similar color background pixels as skin, in particularly in images 3, 4 and 5. In the case of image 4, it is evident that [128], [118] and [119] have detected skin seeds in the dark region on the upper right hand corner of the Bayesian skin map and propagated them as a skin region through spatial analysis resulting in high false positives.

Furthermore, Figure 4.10 shows the skin probability maps generated by the proposed method for sample images with different skin types such as whitish, brownish, yellowish, and darkish skins under different illumination conditions taken from the FSD database. By observing the results given in Figure 4.10, it is evident that the proposed method is robust under different conditions such as skin type and illumination.

The skin segmentation performance was assessed by plotting the ROC (Receiver Operating Characteristic) curves with false negative rate against the false positive

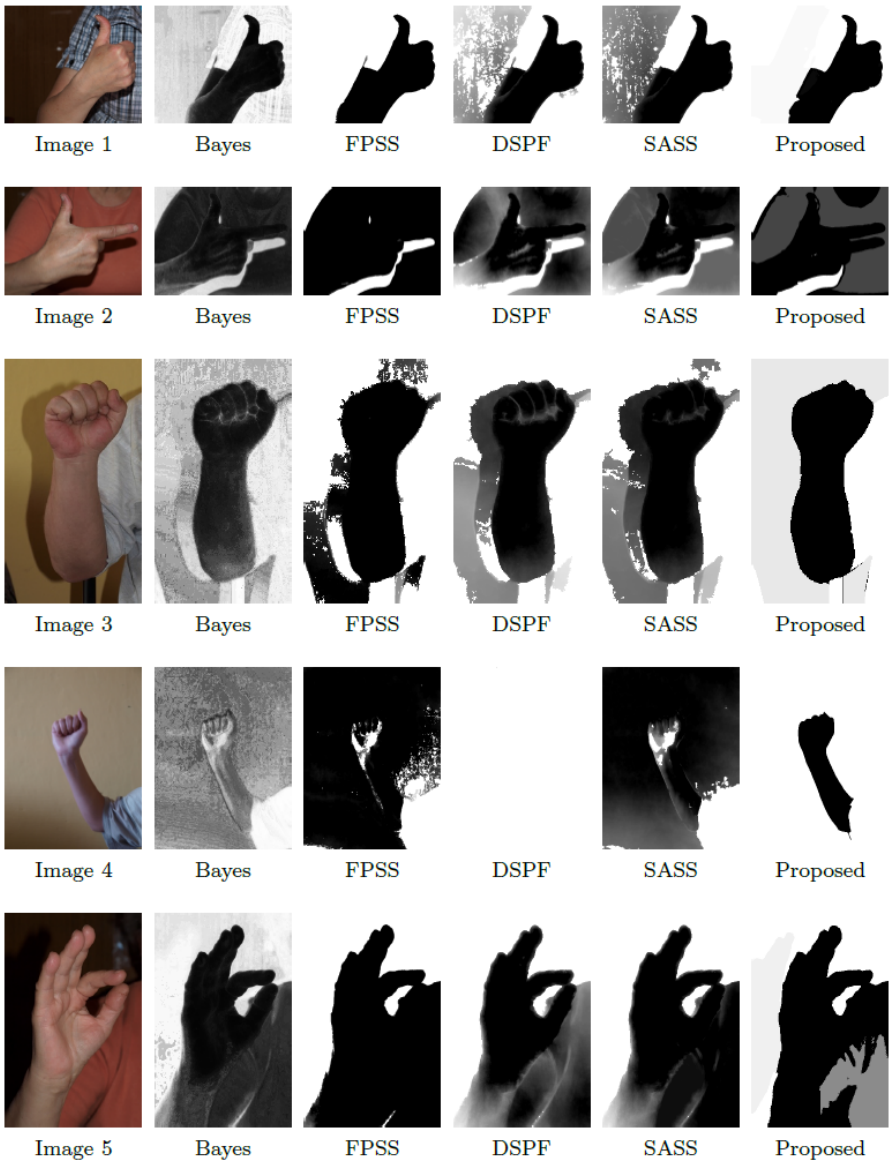


Figure 4.8: Comparison of skin probability maps of some sample images from the HGR database [6]

rate of the skin classifier. False negative rate in this context can be defined as the percentage of skin pixels misclassified as background pixels and the false positive rate can be defined as the percentage of background pixels misclassified as skin pixels. The standard skin segmentation ground truths provided in each benchmark data set were

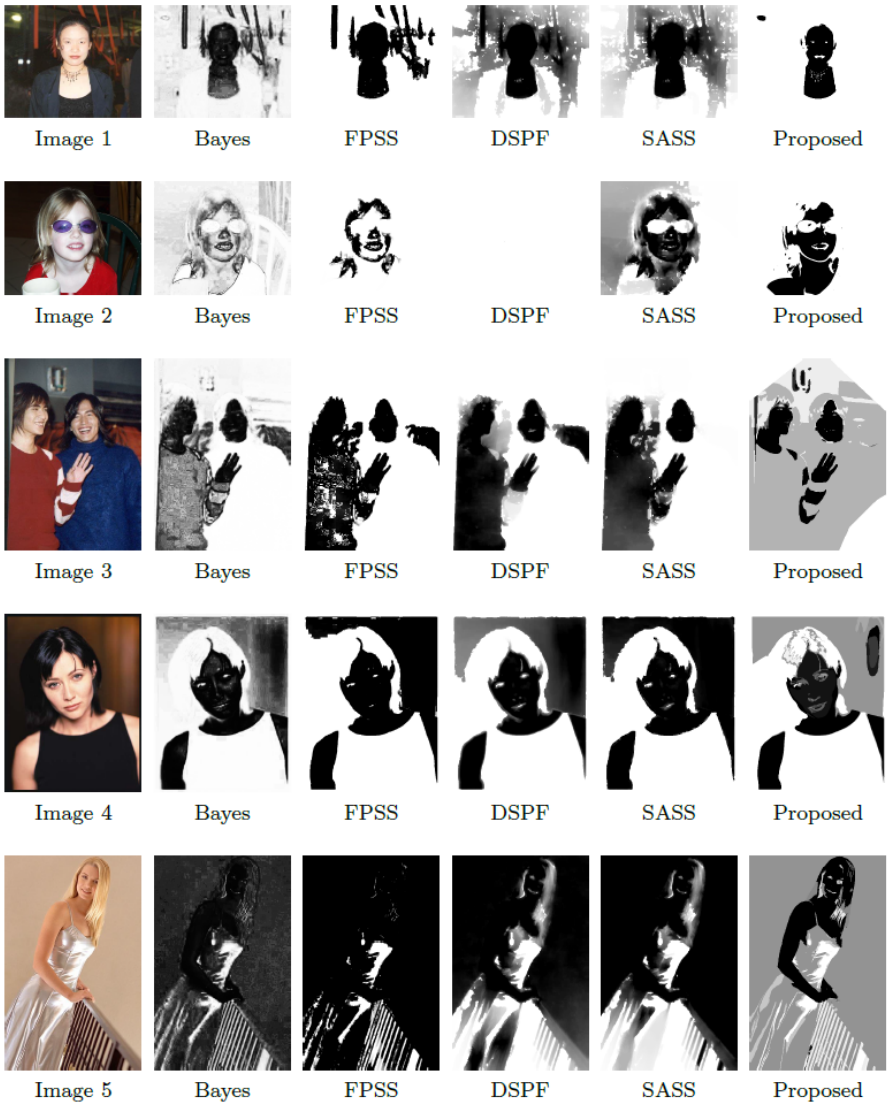


Figure 4.9: Comparison of skin probability maps of some sample images from the FSD database [7]

used to determine the false alarm rates. A satisfactory skin segmentation solution should provide low false negative and false positive rates as well as high F-measure indicating a good balance between precision (percentage of correctly classified pixels out of all the pixels classified as skin) and recall (percentage of skin pixels correctly classified as skin).

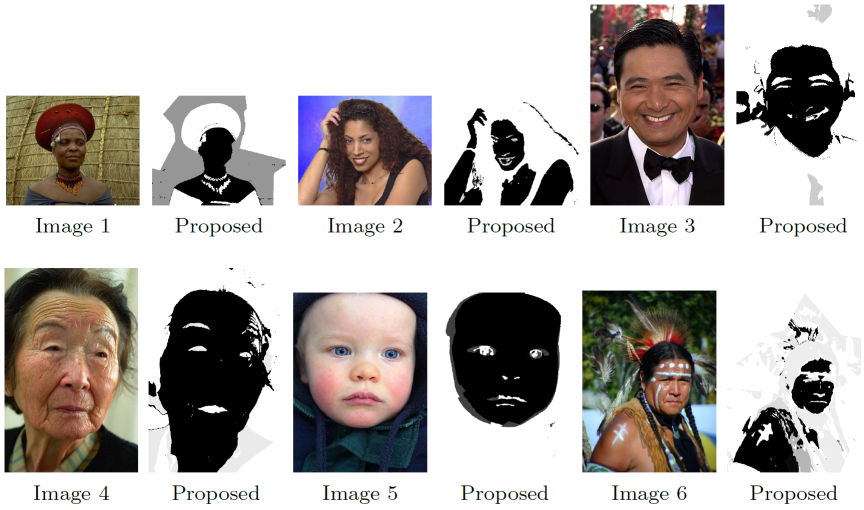
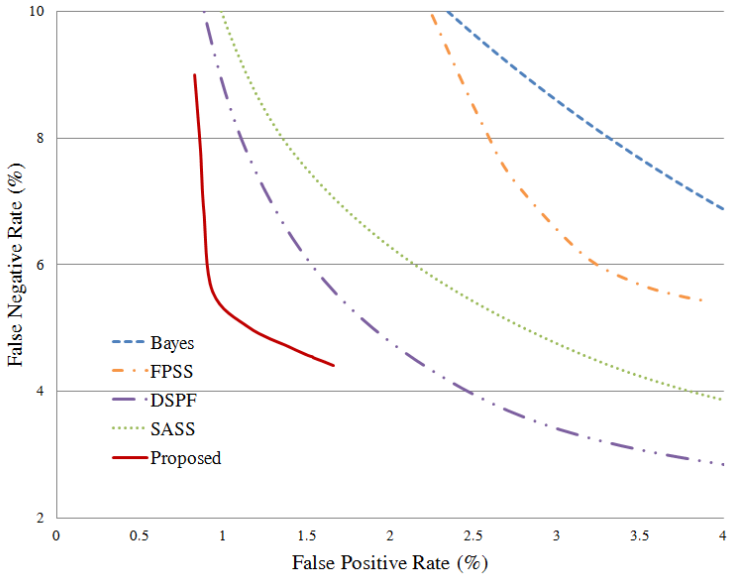


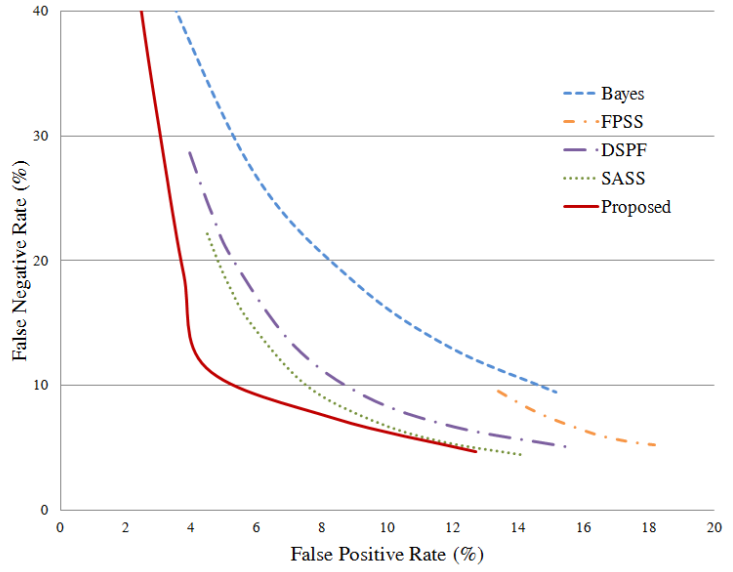
Figure 4.10: Skin probability maps of the proposed method for sample images with different skin types from the FSD database [7]

Figure 4.11 depicts the ROC curves obtained for different skin segmentation methods. By observing Figure 4.11, it is evident that the proposed method outperforms all the other methods for both data sets in terms of the false alarm rates. In the case of the HGR data set, DSPF outperforms SASS while for the FSD data set, SASS outperforms DSPF. Furthermore, Table 4.1 contains a comparison of false alarm rates, false positive rate (δ_{fp}) and false negative rate (δ_{fn}) at the minimal detection error ($\delta_{min} = \min(\delta_{fp} + \delta_{fn})$) and F-measures ($F$) for all the algorithms reported in this section.

In Table 4.1, the proposed method results in the lowest δ_{fp} for both HGR and FSD data sets. DSPF results in the lowest δ_{fn} for HGR data set and SASS results in the lowest δ_{fn} for FSD data set. It is also important to note that the proposed method results in the lowest δ_{min} for both HGR and FSD data sets, which means the lowest minimal detection error is given by the proposed method for both test data sets reported in this chapter. The proposed method reduces the δ_{fp} of HGR data set



(a) ROC curve for HGR data set



(b) ROC curve for FSD data set

Figure 4.11: Comparison of proposed method with existing techniques

Method	**HGR**				**FSD**			
	δ_{fp} (%)	δ_{fn} (%)	δ_{min} (%)	F	δ_{fp} (%)	δ_{fn} (%)	δ_{min} (%)	F
Bayes [114]	5.47	5.15	10.62	0.9470	15.14	9.48	24.62	0.8803
FPSS [128]	3.46	5.72	9.18	0.9536	15.05	7.31	22.35	0.8924
DSPF [118]	2.24	**4.34**	6.58	0.9668	9.31	9.10	18.41	0.9080
SASS [119]	3.16	4.57	7.72	0.9611	9.53	**7.16**	16.68	0.9175
Proposed	**1.66**	4.41	**6.07**	**0.9692**	**4.45**	11.43	**15.74**	**0.9177**

Table 4.1: Comparison of False Alarm Rates at Minimal Detection Error and F-Measures

to 1.66% and it reduces the δ_{fp} of FSD data set to 4.45%, which is a considerable reduction compared to DSPF, which results in the next lowest δ_{fp}. Furthermore, the proposed method gives the highest values for F-measure for both data sets, which means that it shows a good balance between precision and recall compared to other methods reported in this chapter.

Finally, Figure 4.12 shows some sample images from FSD database for which the proposed method fails to provide satisfactory results. It is important to note that the state-of-the-art methods reported in this chapter also fail to provide satisfactory results for most of these images. Kawulok et al. [118] claim that background regions (non-skin) with skin like appearance show a texture pattern on the skin probability map, which cannot be observed for the skin regions. Rather skin regions appear as a smooth texture on the Bayesian skin probability map. Thus, FPSS, DSPF, SASS and the proposed method use the textural features of Bayesian skin probability maps to distinguish between skin and non-skin regions. However, there can be some instances where non-skin regions do not appear as a texture pattern on the Bayesian skin probability map. For example, in image 2, almost all the pixels belonging to the t-shirt of the lady are detected with high skin probability, which is shown as a smooth

texture on the skin probability map (Bayes). Hence, all four methods including the proposed method, segment the t-shirt as a skin region.

Moreover, there can be instances where skin regions do not appear as a smooth texture on the Bayesian skin probability map. For example, the Bayesian skin probability maps of images 1 and 3 do not show a smooth texture on the skin regions. Thus, FPSS and DSPF fail to segment skin in image 1 while SASS and the proposed method detect skin regions with some false positives in the background. For image 3, all four methods fail to provide a satisfactory result. Other than a few failure cases as such, the proposed method provides successful skin segmentation for both HGR and FSD data sets, which is evident by the results given in Figure 4.11 and Table 4.1.

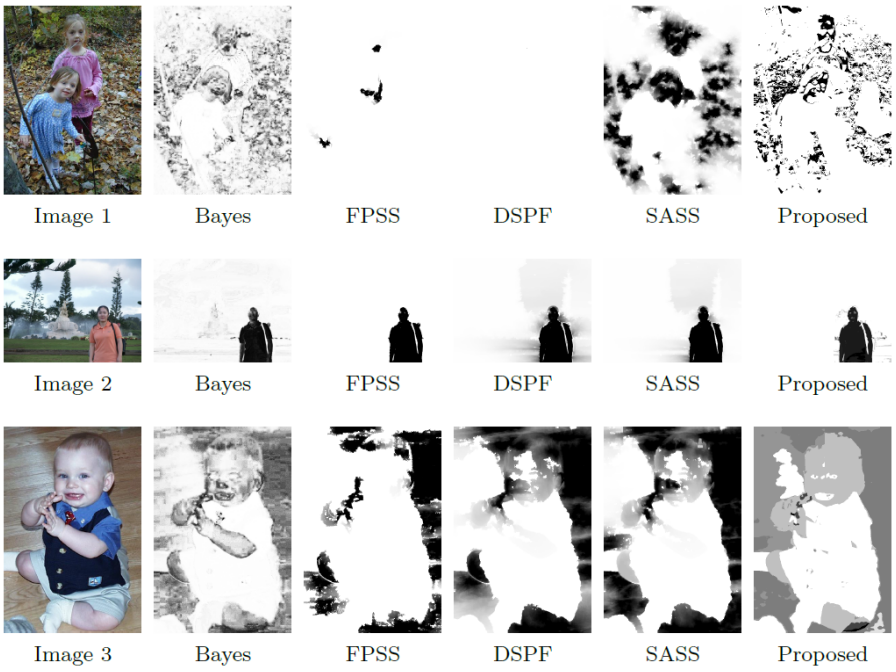


Figure 4.12: Some unsatisfactory results of the proposed method for test images from the FSD database [7]

4.5 Conclusion

In this chapter, a new skin segmentation method is introduced to segment skin in digital images. There are three major contributions of the proposed method. Firstly, in the proposed approach, the non-skin class of the training data is undersampled in an informed manner by sampling non-skin pixels with skin-like appearance as a solution to the class imbalance problem in the training data set. The proposed informed undersampling method may prevent information significant in defining the decision boundary between skin and non-skin classes being lost during undersampling. Also, the balanced training data set prevents the learning algorithms from being biased towards the majority class in the training data set, which may lead to improved skin classifier performance.

Secondly, the proposed method uses an image segmentation technique based on spatial and feature space Dirichlet tessellation to segment the skin candidate regions based on their color information. This color based image segmentation prior to skin classification allows the proposed method to distinguish overlapping skin and background skin candidate regions in the Bayesian skin probability map, which cannot be achieved by the state-of-the-art methods reported in this chapter. Thus, this contribution leads to reduced false positive rates of the skin classification process. Thirdly, the proposed method uses multi-manifold-based skin classification, which again leads to improved skin classification accuracy.

The experimental results reported in this chapter confirm that the proposed method outperforms the existing methods in terms of the false alarm rates on ROC curves. It is also important to note that the proposed method considerably reduces the false positive rate of the skin classifier. Since the existing techniques reported in this chapter are dependent on the seed selection, they lead to high false positive

rates if the seeds are erroneously detected in the background. Moreover, the proposed method results in the lowest minimal detection error and the highest F-measure for both data sets. Thus, we can conclude that the proposed method provides the best skin segmentation accuracy for both data sets compared to the other techniques reported in this chapter.

The processing time of the proposed method mainly depends on the computational complexity of the Voronoï-based segmentation and multi-manifold-based skin classification. Among these two modules, Voronoï-based segmentation is the most computationally expensive module. The Matlab implementation of the proposed method takes approximately 2 seconds to process an image in the HGR data set and approximately 5 seconds to process an image in the FSD data set on an Intel Core i7 2.8GHz processor. However, a practical implementation of the proposed method in a language like C++ can be expected to run much faster. Also parallel computing can be incorporated in both these modules to reduce the execution time of the proposed method significantly in practical applications.

The research work presented in this chapter considerably improves the skin segmentation accuracy by providing solutions to training data class imbalance problem, overlapping skin and background skin candidate regions in the Bayesian skin probability map and learning a meaningful representation of training data. In this work, image segmentation is used to verify the boundaries of skin regions in the original color image in order to distinguish them from similar color background regions on the Bayesian skin probability map. Based on the findings of the research work reported in this chapter, we can conclude that image-segmentation-based verification of the skin classification result improves the accuracy of skin segmentation results in general. Also with regard to skin classifier training data set, we can conclude that

use of a balanced training data set and a proper feature extraction method that can handle non-linear data with a multiple class structure such as multi-manifold learning to extract the most important features from the training data set improve the skin classifier accuracy in skin segmentation.

However, as discussed in Section 4.4, there are still some problems with regard to the textural features extracted from the Bayesian skin probability map that need to be addressed to solve the skin segmentation problem. Thus, it is worthwhile to explore on improvements to features vectors (e.g. scale invariant texture features) of skin and skin-like classes as future research work to overcome these issues.

Chapter 5

Application of Manifold Learning and Skin Segmentation in Hand Gesture Recognition

In this chapter, I propose a framework for hand gesture recognition by using the classical manifold learning and multi-manifold learning introduced in Chapter 2 for feature extraction and the skin segmentation technique introduced in Chapter 4 to accurately segment the hand region. Vision-based hand gesture analysis and recognition has been a widely discussed topic in intelligent human computer interaction systems in the recent past. It has various applications in virtual reality [58], sign language recognition [59–61], sterile human-machine interfaces [62, 63], tele-medicine [64], tele-rehabilitation systems [65, 66], hand gesture-controlled games [67] and many more. Vision-based hand gesture recognition can be subdivided into two main categories: model-based approaches and appearance-based approaches. Model-based approaches consider the kinematic parameters which map the 2D projection image to the 3D hand model, which can then be used to recognize new hand gestures. The appearance-based

approaches use the appearance of predefined 2D image templates of known hand gestures to recognize new hand gestures.

In a real world setting, hand gestures can be static or dynamic. Static hand gestures (hand postures) do not vary over time while the dynamic hand gestures may vary over time. Thus, I will be addressing the static hand gesture recognition and dynamic hand gesture recognition problems separately. The overview of the proposed hand gesture recognition framework is given in Figure 5.1.

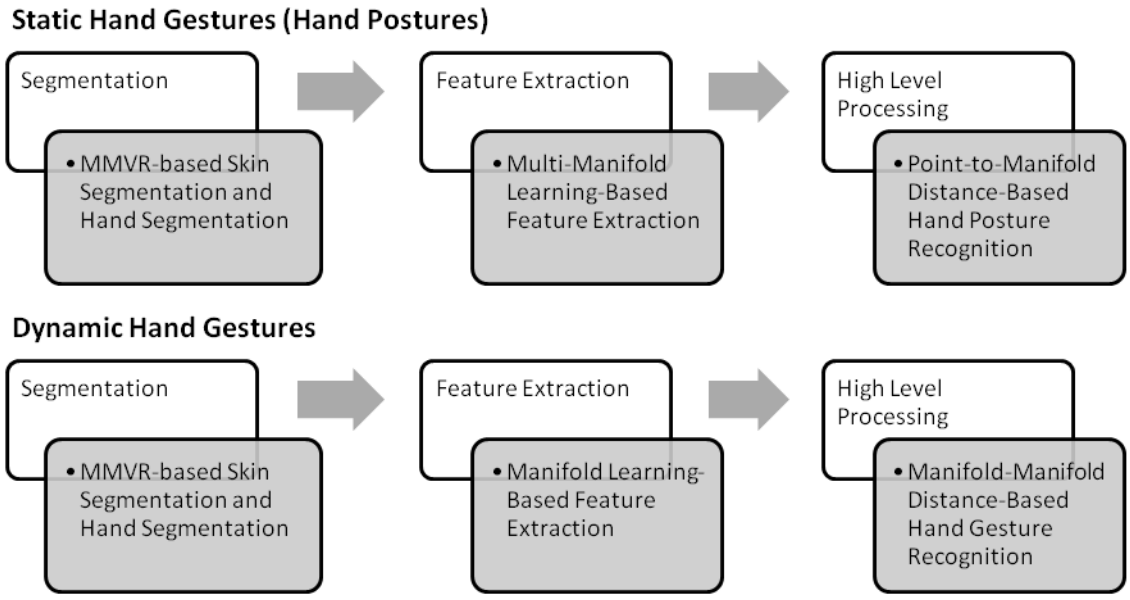


Figure 5.1: The proposed hand gesture recognition framework

5.1 Static Hand Gesture (Hand Posture) Recognition

5.1.1 Introduction

Static hand gestures are defined as orientation and position of hand in the space during an amount of time without any movement [68]. The definition of a hand posture in the context of this thesis is given in Definition 5.1. There are various methods proposed

in the past literature for hand posture recognition. For example, Kumar et al. in [134] use a genetic algorithm to extract features to discriminate hand postures and propose a classifier based on fuzzy rough sets for hand posture classification. Black and Jepson in [135] use an eigenspace approach for hand posture recognition. They use singular value decomposition to find the eigenspace (low dimensional embedding) of a given training data set. Some other approaches for hand posture recognition can be found in [136–139]. More recent approaches such as [140–142] use features extracted from Leap Motion and Microsoft Kinect devices such as position of the fingerprints, palm center and hand orientation in order to perform hand posture recognition.

Definition 5.1. ***Hand Posture***

In this context, a hand posture g can be defined as a 2D image showing the orientation and position of hand in the space at a given time t. g can be alternatively represented by its D-dimensional feature vector $\boldsymbol{x} = \{\phi_1(g), \phi_2(g), \ldots, \phi_D(g)\}$, where $\phi_1, \phi_2, \ldots, \phi_D$ are probe functions for different types of features.

5.1.2 Feature Extraction for Hand Posture Recognition

In appearance-based hand posture recognition, 2D images of hand postures are analyzed to extract features in order to recognize hand postures. Hand postures vary mainly in shape. Thus, shape analysis is important in hand posture recognition. As hand postures of different people may vary in size, orientation and position within a given 2D image of a hand posture, shape descriptors that are invariant to rotation, translation and scaling are essential for successful hand posture recognition.

Hu moment invariants (HM) [143] and Fourier descriptors (FD) [144] are two of the most widely used shape descriptors robust against rotation, translation and scaling. In these experiments, I will be using a combination of Hu's seven moment

invariants and Fourier descriptors to describe the shape of the hand postures. Hence, each hand posture will be represented by a 59-dimensional feature vector. Although the dimensionality of the feature space of hand posture data set is high, these hand postures may vary with only a few degrees of freedom. In other words, not all of the features in this 59-dimensional feature vector are useful in representing a given set of hand postures.

Kumar et al. [134] categorize the available features in a data set into four categories. Those are, (1) Predictive/relevant: the features that are good in discriminating between different classes; (2) Misleading: the features that affect the classification task negatively; (3) Irrelevant: the features that provide a neutral response to the classifier algorithm; and (4) Redundant: the features of a class that have other relevant features for the discrimination. They claim that the presence of misleading features will reduce the classification accuracy and the presence of irrelevant and redundant features will increase the computational burden. Hence, a feature selection mechanism to extract only the relevant features is important for a successful classification process. In [134], they use a genetic algorithm-based feature selection algorithm for feature extraction.

Manifold learning has also been explored in hand posture recognition for feature extraction in the past literature. Choi et al. in [145] use kernel ISOMAP for feature extraction in hand posture recognition and [146] proposes a variation of Locally Linear Embedding (LLE) called distributed locally linear embedding for feature extraction in hand posture recognition. However, it is important to note that only the classical manifold learning techniques, which are capable of learning only a single manifold for a given data set have been explored in the past literature for hand posture recognition.

5.1.3 Proposed Method

Manifold learning reduces the dimensionality of data while preserving the intrinsic structure of data. Thus, it is proven to provide a more meaningful low dimensional representation of a high dimensional data set. In hand posture recognition, usually a training data set of multiple known hand posture types is used to train the classifier in the training phase. In the testing phase, unknown hand postures will be classified based on the learning acquired from training data. As there will be multiple types of hand postures (i.e. multiple classes of data) in a given hand posture training data set, I will be using the multi-manifold learning technique proposed in Chapter 2 to extract features from the training data by learning the low dimensional multi-manifold space. It is experimentally proven in Chapter 2 that multi-manifold learning is more effective in learning data sets with a multiple class structure compared to single manifold learning techniques such as Isometric Feature Mapping (ISOMAP) [28] and Locally Linear Embedding (LLE) [30], which have been used in existing manifold learning-based hand posture recognition methods.

The key stages of the training and recognition phases of the proposed hand posture recognition system are given below. Also, Figure 5.2 depicts the flowchart of the proposed hand posture recognition method. The inputs to the process, intermediate states and the final results are colored in gray, blue and pink respectively.

Training Phase:

1. **Hand Segmentation:** First the hand posture of each image in the data set was extracted by using a skin segmentation algorithm. In the proposed method, I used the Multi-Manifold and Voronoï region (MMVR)-based skin segmentation algorithm proposed in Chapter 4 as it successfully segments skin even with

complex backgrounds. Finally, the hand region was extracted by using the wrist localization technique proposed in [147].

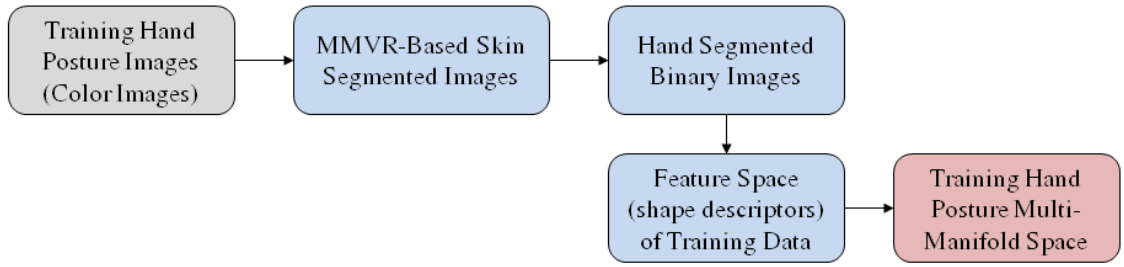
2. **Feature Extraction:** As explained in Section 5.1.2, I used a combination of Hu's seven moment invariants and Fourier descriptors to describe the shape of the hand postures. Hence, each hand posture will be represented by a 59-dimensional feature vector. Next, I applied the multi-manifold learning algorithm proposed in Chapter 2 to extract the salient features of the hand posture data set.

Recognition Phase:

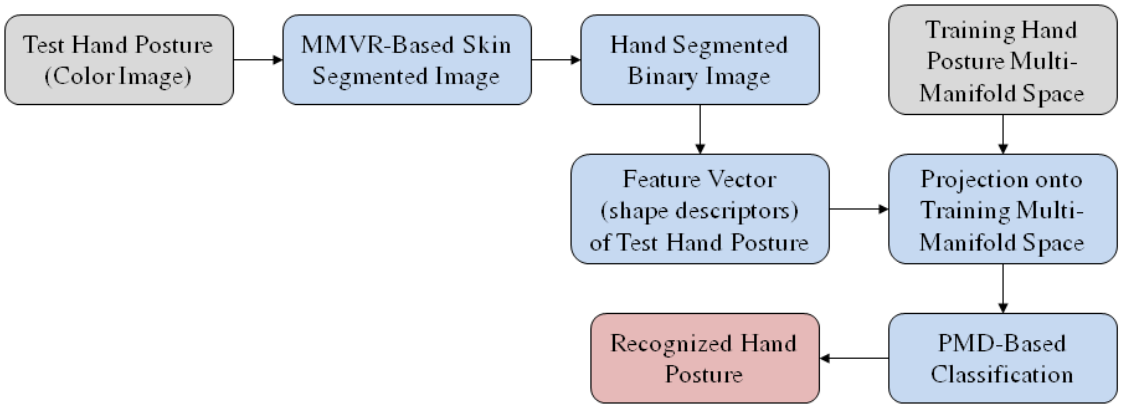
3. **New Posture Classification:** During the test phase, new hand postures were processed and their corresponding feature vectors were extracted as explained in steps 1 and 2. Next, these feature vectors were mapped onto the training multi-manifold space and classified by using the point-to-manifold distance-based classification method introduced in Chapter 2.

5.1.4 Experimental Results and Analysis

In order to confirm the effectiveness of MM-LLE for feature extraction in the context of hand posture recognition, I compared the recognition rates of MM-LLE with that of ISOMAP and LLE by varying the number of dimensions (d) in the low dimensional embedding. Figure 5.3 shows a comparison of recognition rates of MM-LLE, ISOMAP and LLE on 10 gestures selected from the Database for Hand Gesture Recognition (HGR) [6]. The hand postures representing numbers 1 to 5 and alphabetical letters A to E in American Sign Language were selected from the HGR data set for this



(a) Training Phase



(b) Recognition Phase

Figure 5.2: Flowcharts of training and recognition phases of the proposed static hand gesture (hand posture) recognition method

experiment. So the training data set consists of 10 different hand postures posed by 12 different individuals under uncontrolled background and lighting conditions.

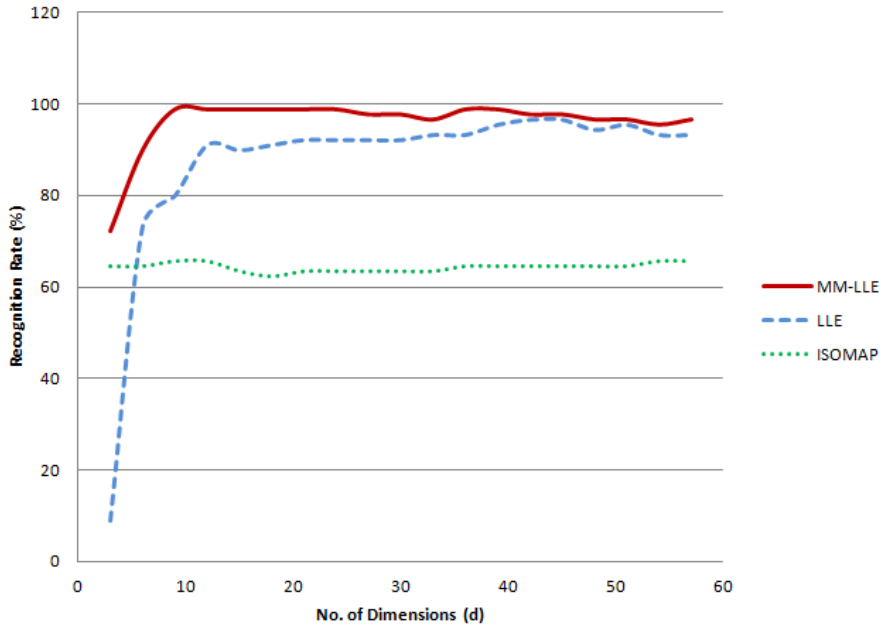


Figure 5.3: Comparison of recognition rates of manifold learning techniques on the HGR1 Data Set

According to Figure 5.3, MM-LLE reaches the highest recognition rate of 98.89% at $d = 9$ while LLE reaches its highest recognition rate of 96.67% at $d = 42$ and ISOMAP reaches its highest recognition rate of 65.55% at $d = 12$. Also, for the whole range of values for d, MM-LLE outperforms LLE and ISOMAP. Hence, it is evident that MM-LLE outperforms LLE and ISOMAP in terms of the hand posture recognition rate on the HGR data set [6].

Next, I compared the performance of MM-LLE for feature extraction in hand posture recognition with two other approaches proposed for hand posture recognition in the past literature. Kumar et al. propose a fuzzy rough sets approach for hand posture recognition in [134]. In their algorithm, they propose a classifier based on fuzzy rough sets (FRC) to classify new hand postures and they use a genetic algorithm

Method	**NUS**				**JTS**			
	25/75%	#F	50/50%	#F	25/75%	#F	50/50%	#F
FRC	91.66%	98	93.33%	92	95.83%	72	**98.75%**	63
SVM	91.8%	1000	92.5%	1000	94.44%	1000	97.91%	1000
ISOMAP	64.44%	27	66.67%	18	77.5%	48	65.42%	9
LLE	95%	36	96.67%	27	90.56%	12	96.25%	24
MM-LLE	**97.78%**	9	**98.33%**	12	**96.25%**	9	98.5%	15

Table 5.1: Comparison of recognition rates of different hand posture recognition methods

to extract the discriminative features of hand postures. Furthermore, they use C_2 standard model features (SMFs) proposed by Serre et al. in [148,149], which are scale and position-tolerant. In [134], the experimental results are reported on two test data sets for both the FRC classifier and Support Vector Machine (SVM) classifier with a polynomial kernel.

Table 5.1 provides a comparison of the hand posture recognition rates of MM-LLE, LLE and ISOMAP with the recognition rates of FRC and SVM reported in [134] on two different test data sets: NUS (National University Singapore) hand posture data set I [134, 150] and Jochen Triesch Static Hand Posture Database (JTS) [151, 152]. The NUS hand posture data set I consists of 10 classes of hand postures and 24 sample images per class, which were captured by varying the position and size of the hand within the image frame. The JTS data set consists of 10 hand postures (a, b, c, d, g, h, i, l, v, y) of 24 persons in 2 different backgrounds (light and dark).

The experimental results are reported for two different training/test ratios: 25/75% and 50/50% of each data set and the number of features (#F) used during classification are given in Table 5.1 as well.

By observing Table 5.1, it is evident that MM-LLE provides the highest hand posture recognition rates compared to other methods, except at 50/50% training/test

ratio for JTS data set. Even in this case, MM-LLE gives 98.5% recognition rate which is only 0.25% less than the highest recognition rate 98.75% given by FRC. Also, it is important to note that FRC provides 98.75% at 63 features while MM-LLE provides 98.5% at 15 features, which is a much lower number of features compared to 63. The number of features used for classification is beneficial when it comes to computational complexity of the classification process. It is also important to note that MM-LLE provides the highest recognition rate at comparatively much lower number of features (dimensions) compared to other methods for all experimental scenarios reported in Table 5.1.

5.2 Dynamic Hand Gesture Recognition

5.2.1 Introduction

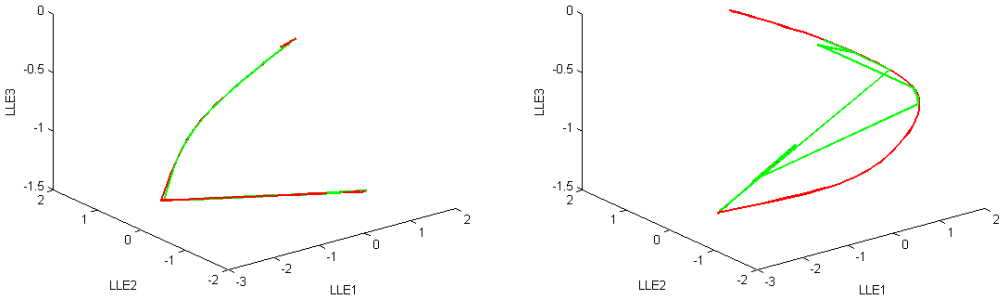
Dynamic hand gestures can be defined as the movement of hand during a given amount of time. Thus, in addition to spatial variations, dynamic hand gestures show temporal variations as well. There are various dynamic hand gesture recognition methods proposed in the past literature. Dynamic gestures can be viewed as actions composed of a sequence of static gestures that are connected by a continuous motion. Hence, most of the past literature performed dynamic hand gesture recognition by identifying individual hand postures that make up the dynamic hand gesture [146, 153, 154].

According to [155], in the image sequence of a hand gesture, there are local motions and global motions. The global motion is the translation of the hand and the local motion is non-rigid motion of the fingers or rotation of the hand. Thus, the global motion of the hand gesture cannot be captured solely by analyzing individual

hand postures in a dynamic hand gesture sequence. As a solution, Chen et al. [155] use Fourier descriptors and motion vectors to estimate the entire motion and they use a Hidden Markov Model (HMM) to represent the statistical behavior of an observable symbol sequence in a dynamic gesture. Similar dynamic gesture recognition method based on HMM were proposed in [156,157]. Some other recent dynamic hand gesture recognition approaches, which use motion divergence fields and depth sensors (Microsoft Kinect device) can be found in [158,159] respectively.

Each frame (image) representing the static hand gestures that make up the dynamic hand gesture can be viewed as points in the high dimensional image space. Since these images are sampled from a single hand gesture and since a given hand gesture has only fewer degrees of freedom, they lie on a low dimensional manifold in this high dimensional image space. Manifold learning is capable of learning low dimensional structures (manifolds) hidden in high dimensional data. Hence, rather than classifying the dynamic hand gesture through classification of static hand gestures that make up the dynamic gesture, I think that manifold learning can be used to analyze both the local and global motion (hand trajectory) of the dynamic gesture simultaneously.

Furthermore, although the within class variation of dynamic hand gestures is high, ideally, the same dynamic hand gesture performed by different people or the same person at different times should have the same underlying manifolds. On the other hand, different hand gestures should ideally have different underlying manifolds. Therefore, I think that it is possible to compare dynamic gestures by comparing their corresponding manifolds. For example, in Figure 5.4, 2 manifolds (shown in red and green) of the same gesture performed by two different people look identical while 2 manifolds of different gestures performed by the same person look different.



(a) 2 manifolds of the same gesture (b) 2 manifolds of different gestures

Figure 5.4: Comparison of manifolds for the same and different dynamic gestures

Thus, I propose a novel dynamic hand gesture recognition method using manifold-manifold distance-based manifold matching. Since, only a single manifold needs to be learned for a given gesture, I will be using LLE, which is a single manifold learning technique to learn the manifold corresponding to a given hand gesture. Unknown hand gestures will be mapped onto the reference hand gesture manifold space via nearest neighbor interpolation technique, which will be discussed later in Section 5.2.2 and will be classified based on the manifold-manifold distance introduced in Chapter 2. Also, I experiment on combinations of different types of features such as binary image input, shape descriptors and motion vectors to represent the original feature set used to extract features.

5.2.2 Related Work

Manifold Learning for Feature Extraction in Dynamic Hand Gesture Recognition

Manifold learning has been considered in the past literature for dynamic hand gesture recognition. However, most of these methods analyze the hand postures that make up the dynamic hand gesture individually in order to recognize the dynamic hand gesture and they do not consider the movement of the hand altogether with the variations in hand posture when performing dynamic hand gesture recognition. For example, [154] uses Locality Preserving Projection (LPP) to learn multiple viewpoint manifolds of various hand postures and then uses these multi-viewpoint manifolds to classify each static hand posture sampled from a dynamic hand movement. Thus in [154], each pose in the feature space is explicitly mapped to the low-dimensional embedding space. Similarly in [146], hand motion is analyzed by reconstructing the static hand postures sampled from the hand movement.

Nearest-Neighbor Interpolation Method

This section explains the nearest-neighbor interpolation method originally proposed in [29], which can be used to map new data points onto the training manifold space. Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be the set of feature vectors of the training data set, where each $\boldsymbol{x}_i$ is a D-dimensional feature vector. Let $M = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$ be the underlying manifold of X such that each $\boldsymbol{x}_i \mapsto \boldsymbol{y}_i$ through manifold learning. A new data point $\widehat{\boldsymbol{x}}$ can be mapped onto the low dimensional embedding space M through the following steps.

1. Identify k-nearest neighbors of $\widehat{\boldsymbol{x}}$, $N_k(\widehat{\boldsymbol{x}})$ among $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$.

2. Compute the linear weights w_i that reconstruct $\widehat{\boldsymbol{x}}$ from its neighbors, $\widehat{\boldsymbol{x}} = \sum_{i=1}^{k} w_i \boldsymbol{x}_i$, subject to the constraint $\sum_{i=1}^{k} w_i = 1$.

3. Find low dimensional embedding coordinates of $\widehat{\boldsymbol{x}}$, $\widehat{\boldsymbol{y}} = \sum_{i=1}^{k} w_i \boldsymbol{y}_i$, where y_i are the low dimensional embedding coordinates of $x_i \in N_k(\widehat{\boldsymbol{x}})$.

5.2.3 Proposed Method

In this section, I build the theoretical foundation of the proposed manifold matching-based dynamic hand gesture recognition method. Ideally, two instances of the same dynamic hand gesture class should have the same underlying manifold. In reality however, underlying manifolds of two instances of the same gesture may not be the same, but we can expect them to be similar. Thus, I start with some definitions of important terms used to explain the proposed method and then I deduce a lemma and a theorem based on those definitions to prove that two instances of the same dynamic gesture class have similar underlying manifold. Furthermore, I use metric proximity (denoted by δ_{D_x}, D_x is the distance metric) to define the similarity between two instances of the same gesture class and similarity between their corresponding manifolds.

First, I define dynamic hand gesture and dynamic hand gesture class in Definitions 5.2 and 5.4 respectively and I define the Hausdorff distance as a metric to define the similarity between sets of feature vectors of two instances of the same dynamic gesture class in Definition 5.3.

Definition 5.2. ***Dynamic Hand Gesture***

In the context of this thesis, a dynamic hand gesture G can be defined as a sequence of hand postures $G = \{g_1, g_2, \ldots, g_N\}$, where N is the total number of hand postures

sampled from the hand gesture. $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ *is the set of feature vectors corresponding to each* $g_i \in G$.

Definition 5.3. ***Hausdorff Distance***

Let X *and* Y *be two non-empty subsets of a metric space. Hausdorff distance* D_H *can be defined by,*

$$D_H(X,Y) = \max\left\{\sup_{x\in X}\inf_{y\in Y}\|x-y\|, \sup_{y\in Y}\inf_{x\in X}\|x-y\|\right\}.$$

Definition 5.4. ***Dynamic Hand Gesture Class***

In this context, a dynamic hand gesture class C_G *is a collection of dynamic hand gestures* $C_G = \{G_1, G_2, \ldots, G_M\}$*, where each* G_i *is an instance of the same dynamic hand gesture* G *and for each* $G_i, G_j \in C_G$*,* $D_H(X^i, X^j) < \varepsilon \Rightarrow X^i \delta_{D_H} X^j$*, where* X^i *and* X^j *are the sets of feature vectors of* G_i *and* G_j *respectively and* ε *is a predefined threshold.*

Lemma 5.1. *Let* G_i *and* G_j *be two dynamic hand gesture instances and let* X^i *and* X^j *be sets of feature vectors of* G_i *and* G_j *respectively. If* $G_i \in C_G$ *and* $G_j \in C_G$*, where* C_G *is a dynamic hand gesture class representing the dynamic hand gesture* G*, then* $\forall x_i \in X^i$ *there is at least one* $x_j \in X^j : \|x_i - x_j\| < \varepsilon$ *and* $\forall x_j \in X^j$ *there is at least one* $x_i \in X^i : \|x_i - x_j\| < \varepsilon$*, where* ε *is a predefined threshold.*

Proof. If $G_i \in C_G$ and $G_j \in C_G$, then from Definition 5.4, $D_H(X^i, X^j) < \varepsilon$. Immediately from Definition 5.3, in order to satisfy the condition $D_H(X^i, X^j) < \varepsilon$, $\forall x_i \in X^i$ there should be at least one $x_j \in X^j : \|x_i - x_j\| < \varepsilon$ and $\forall x_j \in X^j$ there should be at least one $x_i \in X^i : \|x_i - x_j\| < \varepsilon$. □

Then, I use the Manifold-Manifold Distance (MMD) defined by Definition 2.8 in Chapter 2 as a metric to compare the similarity between two manifolds.

Theorem 5.1. *Let G_i and G_j be two dynamic hand gesture instances and let X^i and X^j be sets of feature vectors of G_i and G_j respectively. Let $f : X^i \rightarrow M_i$ and $f : X^j \rightarrow M_j$, where f is a neighborhood preserving mapping, and M_i, M_j are underlying manifolds of G_i and G_j respectively. If $G_i \in C_G$ and $G_j \in C_G$, where C_G is a dynamic hand gesture class representing the dynamic hand gesture G, then $M_i \ \delta_{D_M} \ M_j \Rightarrow D_M(M_i, M_j) < \varepsilon'$. D_M is the manifold-manifold distance defined in Definition 2.8 in Chapter 2.*

Proof. If $G_i \in C_G$ and $G_j \in C_G$, then from Lemma 5.1, $\forall x_i \in X^i$ there is at least one $x_j \in X^j : \|x_i - x_j\| < \varepsilon$ and $\forall x_j \in X^j$ there is at least one $x_i \in X^i : \|x_i - x_j\| < \varepsilon$. Given that $\|x_i - x_j\| < \varepsilon$, from Definition 2.4, $x_j \in N_{\varepsilon,k}(x_i)$ and similarly, $x_i \in N_{\varepsilon,k}(x_j)$. From Definition 2.6, $N_{\varepsilon,k}(x_i) \mapsto N_{\varepsilon',k}(y_i), y_i = f(x_i) \in M_i$ and $N_{\varepsilon,k}(x_j) \mapsto N_{\varepsilon',k}(y_j), y_j = f(x_j) \in M_j$. Therefore, $\forall y_i \in M_i$ there is at least one $y_j \in M_j : \|y_i - y_j\| < \varepsilon'$ and $\forall y_j \in M_j$ there is at least one $y_i \in M_i : \|y_i - y_j\| < \varepsilon'$. Thus from Definition 2.8, $D_M(M_i, M_j) < \varepsilon' \Rightarrow M_i \ \delta_{D_M} \ M_j$. □

Hence, from Theorem 5.1, we can conclude that two instances of the same dynamic hand gesture have similar underlying manifolds, which can be used to classify an unknown gesture instance by comparing its manifold with that of a known gesture instance. Next, I give the step by step process of the proposed manifold matching-based dynamic hand gesture recognition method. Also, Figure 5.5 depicts the flowchart of the proposed hand gesture recognition method. The inputs to the process, intermediate states and the final results are colored in gray, blue and pink respectively.

Training Phase:

1. **Hand Segmentation:** First the hand region of each frame in the video of dynamic hand gesture was extracted by using a skin segmentation algorithm. In

the proposed method, I used the Multi-Manifold and Voronoï region (MMVR)-based skin segmentation algorithm proposed in Chapter 4 as it successfully segments skin even with complex backgrounds. Finally, the hand region was extracted by using the wrist localization technique proposed in [147].

2. **Feature Extraction:** Each frame in the video of dynamic hand gesture can be represented by different features. Different feature vectors can be used to describe the hand region of a given video frame, which will be discussed in detail in Section 5.2.4. A given gesture G can be represented by its set of feature vectors $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ is the feature vector representing the i^{th} image and N is the number of images of the image sequence that makeup the gesture G. Next, I apply LLE manifold learning on the set of feature vectors X of gesture G to learn the low dimensional manifold $M = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N]$, where $\boldsymbol{y}_i \in \mathbb{R}^d$ and $d <<< D$. During this phase, the reference manifold of each gesture will be learned using LLE.

Recognition Phase:

3. **Manifold Alignment:** In the recognition phase, in order to compare two manifolds, we need to bring them into the same space and remove any rotational, translational and scaling factors. Manifold alignment is a technique used to align two manifolds so that they can be compared. Since we use LLE to learn the low dimensional manifold space, we can use the technique based on nearest-neighbor interpolation proposed in [29] to map new data points into a given manifold space. Through this method, we can map all the points of a new gesture onto the manifold space of each known gesture and then compare those two manifolds.

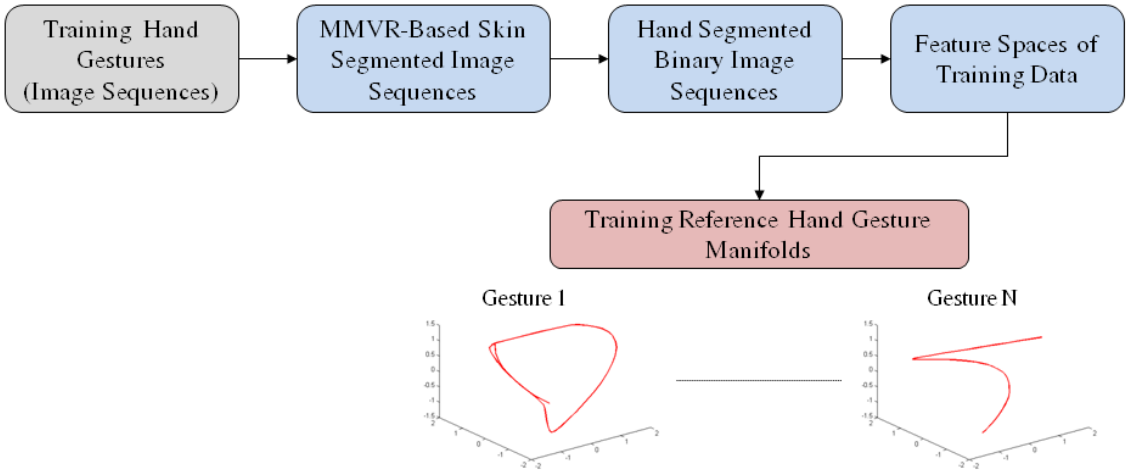
4. **New Gesture Classification:** In this step, a given unknown hand gesture will be mapped onto the reference manifold space of each known hand gesture as explained in step 3 and they will be compared by using the manifold-manifold distance defined by Definition 2.8. Finally, the new gesture will be classified based on class label of the reference manifold that is nearest (which gives the minimum MMD to the test manifold) to it.

5.2.4 Experimental Results and Analysis

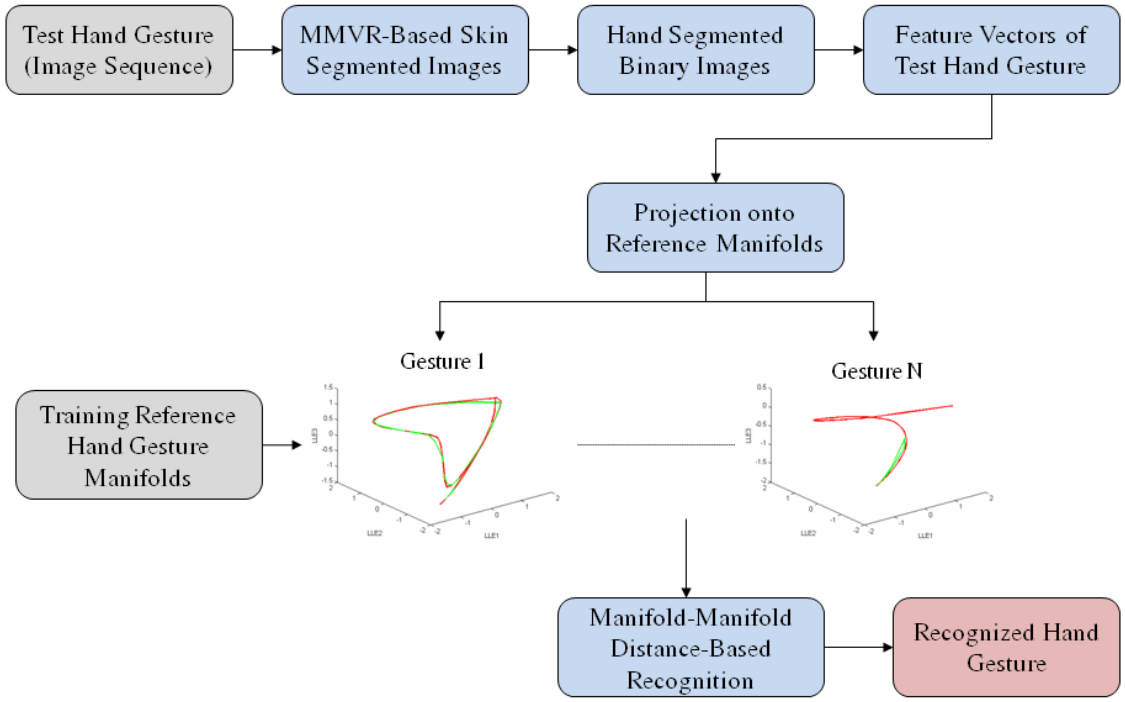
Four different well known dynamic hand gestures namely, "round", "slide", "click" and "zoom" were used during these experiments. Videos of six different people performing at least 3 variations of each of the four gestures were captured by using a Microsoft Lifecam Cinema webcam (30 frames per second). The image sequence (frames) of each video file was saved and processed separately to segment the hand region in each image. So altogether there are 20 image sequences (variations) per each gesture. Figure 5.6 shows sample frames extracted from each of the four gesture types.

The hand posture sequences of "round" and "slide" gestures mainly vary based on location (global motion/translation), the hand posture sequence of "click" gesture varies mainly based on the size (global motion perpendicular to camera) of the hand region and finally the hand posture sequence of "zoom" gesture varies mainly based on the shape (local motion of fingers) of the hand region. Therefore, three different combinations of features were used during these experiments as given below to track all these variations.

1. **Binary Images:** Raw binary images of the segmented hand region in each $g_i \in G$. Each binary image contains value 1 for pixels belonging to the hand



(a) Training Phase



(b) Recognition Phase

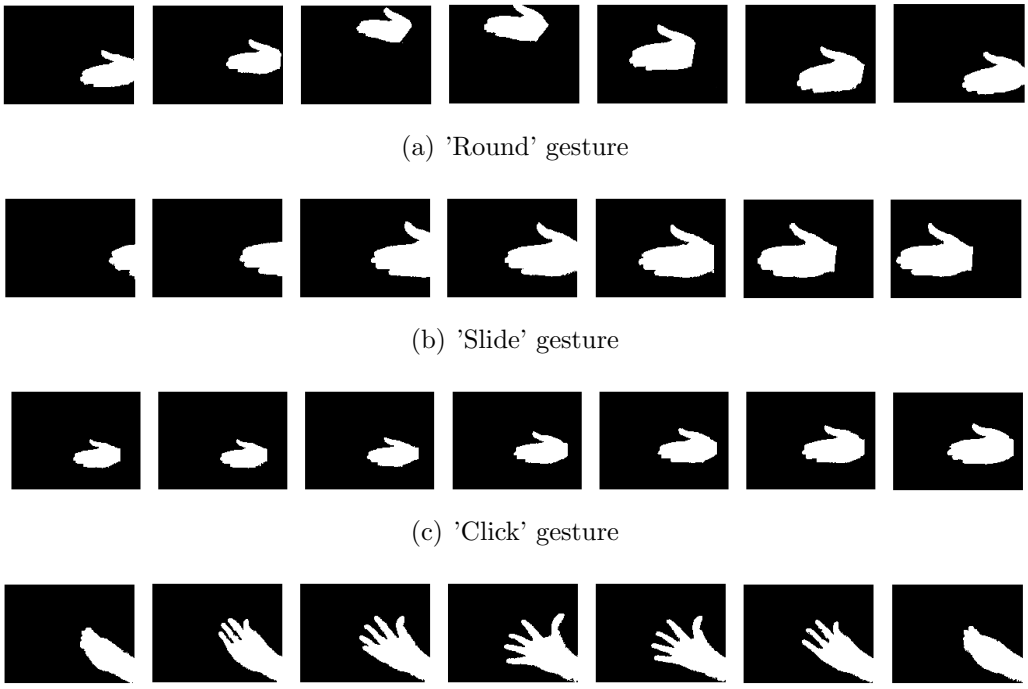
Figure 5.5: Flowcharts of training and recognition phases of the proposed dynamic hand gesture recognition method

region and 0 otherwise and each image is of size 320×240. Thus, each image was represented by a 76800-dimensional feature vector. Raw binary images contain all information regarding global and local motion of each gesture type.

2. **Shape and Motion Descriptors:** Shape descriptors (Fourier descriptors and Hu moment invariants mentioned in Section 5.1.2) to track the hand shape and motion vectors to estimate motion of the hand through space-temporal image intensity gradients proposed in [155, §3.2] to track the motion of the hand of each $g_i \in G$. 7 Hu moment invariants, 52 Fourier descriptors and 22 motion descriptors make up a 81-dimensional feature vector.

3. **Shape, Size and Location Descriptors:** Shape descriptors (Fourier descriptors and Hu moment invariants) and two other features defined to track the changes in size and location of the hand region were used for this combination. Those two parameters are: change of area of hand region ($area(g_i) - area(g_0)$) and change of X,Y coordinates of the centroid of the hand region ($centroid(g_i) - centroid(g_0)$), where g_0 is the initial hand posture of the sequence. 7 Hu moment invariants, 52 Fourier descriptors, change of area and change of X,Y coordinates of centroid make up a a 62-dimensional feature vector.

Feature vectors of one of the image sequence out of 20 sequences were selected to learn the reference manifold and the feature vectors of rest of the 19 sequences were mapped onto the reference manifold space by using the nearest-neighbor interpolation method. A manifold, which captured the entire gesture accurately was selected as the reference manifold in each case. Figure 5.7 shows the final result of test manifolds (in green) being mapped onto the reference manifold (in red) space for the four gestures used in these experiments. By observing Figure 5.7, it is evident that the underlying



(a) 'Round' gesture

(b) 'Slide' gesture

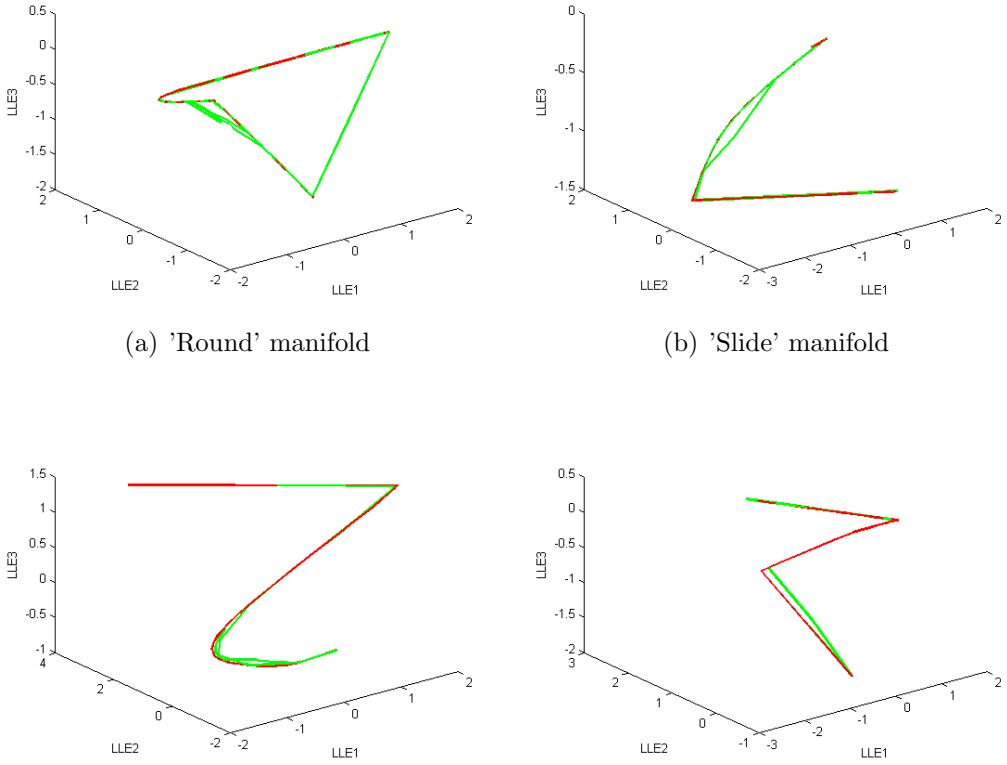
(c) 'Click' gesture

(d) 'Zoom' gesture

Figure 5.6: Sample frames from four different types of dynamic gesture videos

manifolds of the same gesture performed under several variables (different people, different times, different backgrounds, etc.) are similar.

Next, I performed dynamic hand gesture recognition by comparing test manifolds with the reference manifold using manifold-manifold distance. Each test gesture was mapped onto each of the four reference manifold spaces and was assigned the class label of the reference manifold that was closest to it in terms of the manifold-manifold distance. Then, I compared the results of the proposed manifold matching-based dynamic hand gesture recognition method with HMM-based dynamic hand gesture recognition method [155–157] as HMM is one of the most widely used dynamic hand gesture recognition approach. I used 50% of my data set as training data to learn the HMM model and tested the HMM model with both training and test data.



(a) 'Round' manifold

(b) 'Slide' manifold

(c) 'Click' manifold

(d) 'Zoom' manifold

Figure 5.7: Test manifolds mapped onto reference manifold space for the four types of gestures

Figure 5.8 shows the comparison of recognition rates and Table 5.2 shows the confusion tables of the proposed manifold matching-based method and HMM-based method for each dynamic hand gesture class and for each type of feature vectors. By observing Figure 5.8 and Table 5.2, it is evident that the proposed method provides the best recognition rates for all gesture types and for all type of feature vectors compared to the recognition rates of the HMM-based method. HMM-based method fails to provide satisfactory results for most cases except for 95% recognition rate for Round gesture with shape, size & location feature vectors. But under the same

feature vectors, HMM fails to recognize Click and Zoom gestures at all.

The proposed method provides good recognition rates for Round and Slide gestures for all three feature vector types. Under shape & motion and shape, size & location feature vectors, the proposed method reaches 100% recognition rate for the Round gesture. The recognition rate of the proposed method for the Click gesture is poor under shape & motion feature vectors and shape, size & location feature vectors compared to its recognition rate for binary image feature vectors. For the Zoom gesture, the proposed method provides the best result under shape, size & location feature vectors.

Binary Images

	Manifold Matching				**HMM**			
	Round	Slide	Click	Zoom	Round	Slide	Click	Zoom
Round	**0.9**	0.1	0	0	**0.15**	0.2	0.6	0.05
Slide	0.05	**0.9**	0	0.05	0	**0.5**	0.3	0.2
Click	0.05	0.1	**0.8**	0.05	0	0.3	**0.25**	0.45
Zoom	0.1	0.15	0	**0.75**	0	0.6	0.25	**0.15**

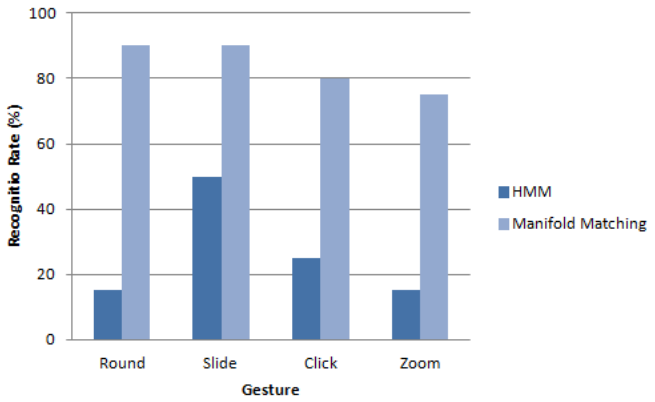
Shape & Motion Descriptors

	Manifold Matching				**HMM**			
	Round	Slide	Click	Zoom	Round	Slide	Click	Zoom
Round	**1**	0	0	0	**0.7**	0.05	0.05	0.2
Slide	0.05	**0.9**	0.05	0	0	**0.65**	0	0.35
Click	0.15	0.25	**0.55**	0.05	0.15	0.25	**0.3**	0.3
Zoom	0.1	0.25	0.05	**0.6**	0	0.4	0.15	**0.45**

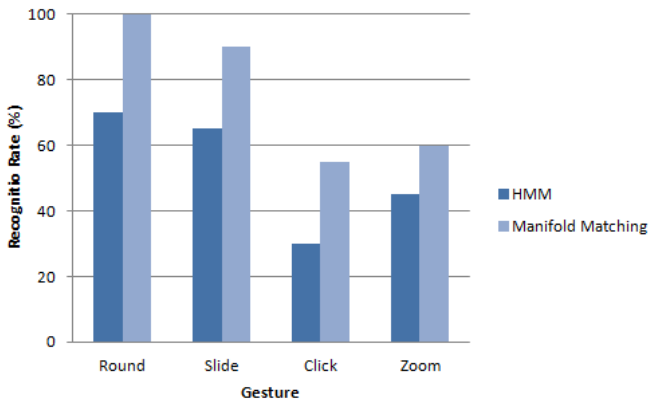
Shape, Size & Location Descriptors

	Manifold Matching				**HMM**			
	Round	Slide	Click	Zoom	Round	Slide	Click	Zoom
Round	**1**	0	0	0	**0.95**	0.05	0	0
Slide	0	**0.8**	0.2	0	0.45	**0.55**	0	0
Click	0.15	0.2	**0.65**	0	0.25	0.75	**0**	0
Zoom	0	0	0.15	**0.85**	0.8	0.2	0	**0**

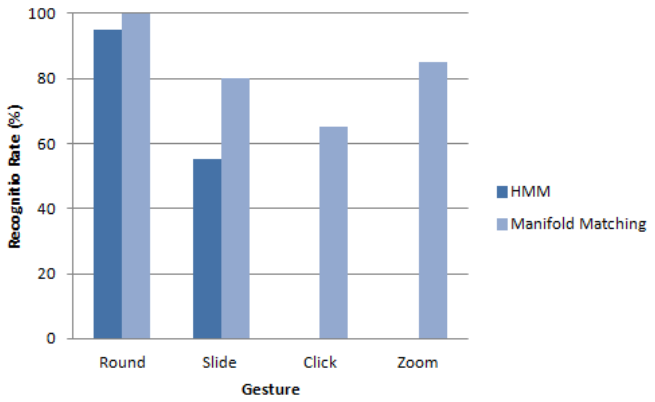
Table 5.2: Confusion tables of dynamic hand gesture recognition results of HMM and the proposed method



(a) Recognition rates for raw binary images



(b) Recognition rates for shape & motion descriptors



(c) Recognition rates for shape, size & location descriptors

Figure 5.8: Comparison of recognition rates of HMM and the proposed method for different gestures

Finally, Figure 5.9 gives a comparison of the overall recognition rates of HMM-based and manifold-matching based dynamic hand gesture recognition methods for each type of feature vector. By observing Figure 5.9, it is evident that the proposed method provides the best overall recognition rates for all three feature vector types compared to HMM. Also, it is important to note that the proposed method provides the best overall recognition rate when raw binary images are used as feature vectors. This may be due to the fact that raw binary images contain all the information while the other feature vectors contain only a predefined set of features such as shape and motion. The second best result for the proposed method is given for the shape, size & location feature vectors.

For HMM, the best overall recognition rate is given for the shape & motion feature vectors and it provides the worst recognition rates when the raw binary images are used as feature vectors. This may be due to the very high dimensionality (76800-dimensional) of the raw binary image feature vectors as HMM uses vector quantization to convert the multi-dimensional feature vector sequences to one-dimensional symbol sequences. I think that lack of training data may have been another reason for HMM to provide poor recognition rates in these experiments, but HMM failed to provide satisfactory recognition rates even when I tested the model with the training data that were used to train the HMM model.

5.3 Conclusion

By observing the experimental results given in Sections 5.1.4 and 5.2.4, it is evident that manifold learning-based feature extraction provides the best hand gesture recognition rates in both static and dynamic hand gesture recognition. In hand posture recognition, MM-LLE outperformed genetic algorithm-based feature extraction, SVM

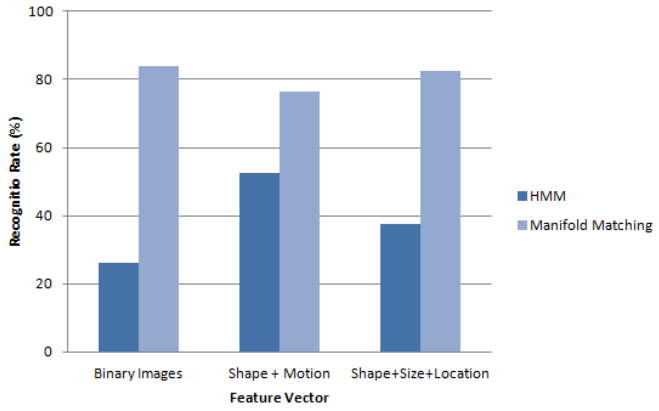


Figure 5.9: Comparison of overall recognition rates of HMM and the proposed method

and single manifold learning techniques reported in Section 5.1.4. Also, it is important to note that MM-LLE provides the highest recognition rate at comparatively much lower number of features (dimensions) compared to all other methods, which is advantageous when it comes to computational complexity of the recognition stage.

With regard to dynamic hand gesture recognition, it was shown in Section 5.2.3 that two instances of the same gesture class have similar underlying manifolds, which allows recognition of new hand gestures by comparing their corresponding manifolds with reference manifolds of known gestures. Experimental results reported in Section 5.2.4 show that the proposed manifold matching-based dynamic hand gesture recognition method outperforms HMM-based hand gesture recognition for the given test data set. Also, it is important to note that the best overall recognition rate of the proposed dynamic hand gesture technique was given when raw binary images were used as feature vectors. By observing the results given in Section 5.2.4, it is evident that manifold learning can be used to simultaneously analyze local and global motion as it supports automatic feature extraction, which captures both global and local motion in a given hand gesture data set. Overall, we can conclude that manifold

learning provides a tool for successful feature extraction in both static and dynamic hand gesture recognition.

Furthermore, the skin segmentation technique proposed in Chapter 4 provides accurate boundaries of the hand region during the hand segmentation process, which is important when selecting features representing hand gestures such as shape descriptors. This also leads to improved recognition rates in both static and dynamic hand gesture recognition.

Chapter 6

Conclusion and Future Directions

6.1 Conclusion

Development of the computer vision field is crucial for the next generation of computing, robotic and artificial intelligence systems. The rapid development of the computer vision field demands advancements in computer vision technologies. Feature extraction and segmentation are two key stages of a typical computer vision system and in this thesis, I proposed novel solutions for some of the inherent problems in feature extraction and segmentation stages of a computer vision system by using multi-manifold learning-based feature extraction, Voronoï region-based image segmentation and multi-manifold learning and Voronoï region-based skin segmentation. Furthermore, I proposed a framework for hand gesture recognition by applying the proposed solutions in feature extraction and segmentation to solve the hand gesture recognition problem.

In Chapter 2, I addressed the problem of simultaneously learning multiple manifolds present in a data set with a multiple class structure. The multi-manifold learning algorithm proposed in Chapter 2 is capable of simultaneously learning multiple

manifolds present in a data set and it provides automatic feature extraction, which improves the classification accuracy in higher level processing stages (e.g. recognition, classification). Also, I have shown in Chapter 2 that point-to-manifold distance can be used as a metric to classify an unknown data sample mapped onto the training multi-manifold space leading to higher classification accuracy compared to KNN classification on the multi-manifold space. The computational complexity of the proposed multi-manifold learning algorithm is also much lower compared to the existing multi-manifold learning algorithms.

I addressed the problem of adaptive unsupervised image segmentation in Chapter 3. I proposed a cluster-based image segmentation algorithm based on spatial and feature space Dirichlet tessellation, which is capable of automatically finding the number of clusters and cluster centroids of a given image. I have also shown in Chapter 3 that the Voronoï region wise clustering reduces the computational complexity of the image segmentation problem significantly while improving the accuracy of the segmentation process. Next, I provided solutions to some of the inherent problems in skin segmentation, namely, the skin classifier training data class imbalance problem, overlapping skin and background skin candidate regions in the Bayesian skin probability map and learning a meaningful low dimensional representation of training data in Chapter 4. I proposed an informed undersampling method to solve the class imbalance problem and I used the multi-manifold learning algorithm proposed in Chapter 2 for feature extraction to learn a more meaningful low dimensional representation of training data, which has led to improved skin classification accuracy. Furthermore, I have shown in Chapter 4 that image segmentation-based verification of the skin classification result improves the accuracy of the skin segmentation process.

Finally, in Chapter 5, I proposed a framework for hand gesture recognition by

using manifold learning for feature extraction and the proposed skin segmentation technique for segmenting the hand region in gesture images. It was shown in Chapter 5 that manifold learning provides a tool for successful feature extraction in both static and dynamic hand gesture recognition and the proposed skin segmentation technique provides accurate boundaries of the hand region, which is important when selecting features such as shape descriptors. Furthermore, in Chapter 5, I have shown that different instances of the same dynamic hand gesture have similar underlying manifolds, which provides a novel way of hand gesture recognition through manifold matching. Overall, I can conclude that the proposed improvements in feature extraction and segmentation as well as the manifold-based classification and recognition have led to improved recognition rates in both the static and dynamic hand gesture recognition systems. Figure 6.1 summarizes the overall contribution of the research work presented in this thesis.

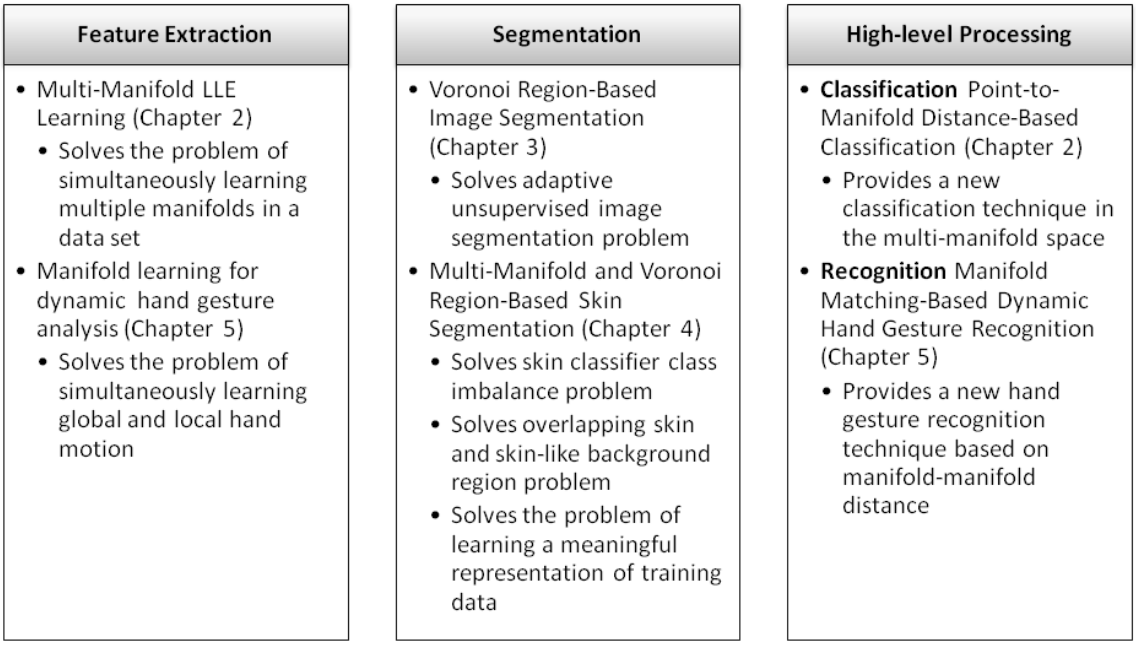


Figure 6.1: Summary of the overall contribution of the thesis

6.2 Future Directions

In this section, I discuss some of the potential research directions, which may help further advance the research work presented in this thesis.

1. **Estimating the Optimum Dimensionality of Multi-Manifold Embedding Space**

   In Chapter 2, I used minimum manifold-manifold distance as a metric to find the optimum low dimensional embedding, which provided a high recognition rate. There are automatic intrinsic manifold approximation techniques proposed in the past literature to estimate the dimensionality of the underlying manifold in the case of classical single manifold learning techniques. However, in the case of multiple manifold learning, determining the optimum dimensionality of the multi-manifold space becomes more challenging as there are multiple manifolds and intrinsic dimensionality of each manifold can be different from one another. Future research work on estimating the optimum dimensionality of multi-manifold embedding space may improve the performance of the higher level processes such as recognition.

2. **Use of Scale and Rotation Invariant Textural Features for Skin Segmentation**

   The textural features used during the experiments in Chapter 4 were scale variant textural features. The neighborhood sizes used to extract these textural features were decided assuming that the skin regions of the test data sets were fairly large. However, it is worthwhile to further research on rotation and scale invariant textural features [160] to further improve the skin classifier accuracy.

3. **Learning a Generic Reference Hand Gesture Manifold**

In Chapter 5, I simply selected a manifold of a dynamic hand gesture, which captured the entire dynamic hand gesture accurately as the reference manifold for manifold matching-based recognition. However, if the reference manifold is found in such a way that it represents the general structure of all instances of a given hand gesture class, it may provide better gesture recognition rates. Manifolds of several instances of the same dynamic hand gesture can be used as training data to learn a generic manifold, which can then be used as the reference hand gesture manifold during recognition phase.

4. **Use of More Sophisticated Hand Gesture Features**

   In Chapter 5, I used only the features extracted from raw binary images of segmented hand regions such as shape, motion, etc. However, it is worthwhile to experiment further on other sophisticated features, which can be extracted from motion and depth sensors such as Microsoft Kinect device and Leap motion controller. Addition of these features may provide better gesture recognition rates.

Bibliography

[1] S. A. Nene, S. K. Nayar, H. Murase *et al.*, "Columbia object image library (coil-20)," Technical Report CUCS-005-96, Tech. Rep., 1996.

[2] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1075–1088, 2003.

[3] J. Z. Wang, J. Li, and G. Wiederhold, "Simplicity: Semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, 2001.

[4] "The Berkeley Segmentation Dataset and Benchmark," https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/, 2007, [Online; accessed 19-Feb-2016].

[5] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.

[6] "Database for Hand Gesture Recognition (HGR)," http://sun.aei.polsl.pl/~mkawulok/gestures/, 2013, [Online; accessed 14-Jul-2015].

[7] S. L. Phung, A. Bouzerdoum, and D. Chai, "Skin segmentation using color pixel classification: analysis and comparison," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 1, pp. 148–154, 2005.

[8] D. Storcheus, A. Rostamizadeh, and S. Kumar, "A survey of modern questions and challenges in feature extraction," in *Proceedings of The 1st International Workshop on Feature Extraction: Modern Questions and Challenges, NIPS*, 2015, pp. 1–18.

[9] J. Milnor, "Topological manifolds and smooth manifolds," in *Proc. Internat. Congr. Mathematicians (Stockholm, 1962)*, 1963, pp. 132–138.

[10] S. Willard, *General Topology*. Courier Corporation, 1970.

[11] J. F. Peters, "Computational Proximity," in *Excursions in the Topology of Digital Images*. Springer, 2016, pp. 1–62.

[12] T. Rowland, "Manifold," http://mathworld.wolfram.com/Manifold.html, 2016, [Online; accessed 30-Aug-2016].

[13] I. Jolliffe, "Principal component analysis," *Springer Series in Statistics, Berlin: Springer, 1986*, vol. 1, 1986.

[14] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural NetworksICANN'97*. Springer, 1997, pp. 583–588.

[15] W. Liu, H. Zhang, D. Tao, Y. Wang, and K. Lu, "Large-scale paralleled sparse principal component analysis," *Multimedia Tools and Applications*, pp. 1–13, 2013.

[16] S. G. Krantz, *A Guide to Topology*. Mathematical Association of America, 2009, no. 40.

[17] E. W. Weisstein, "Continuous Function," http://mathworld.wolfram.com/ContinuousFunction.html, 2016, [Online; accessed 30-Aug-2016].

[18] Y. Han, Z. Xu, Z. Ma, and Z. Huang, "Image classification with manifold learning for out-of-sample data," *Signal Processing*, vol. 93, no. 8, pp. 2169–2177, 2013.

[19] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based-nearest-neighbor for hyperspectral image classification," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 48, no. 11, pp. 4099–4109, 2010.

[20] D. de Ridder and R. P. Duin, "Locally linear embedding for classification," *Pattern Recognition Group, Dept. of Imaging Science & Technology, Delft University of Technology, Delft, The Netherlands, Tech. Rep. PH-2002-01*, pp. 1–12, 2002.

[21] S. You and H. Ma, "Manifold topological multi-resolution analysis method," *Pattern Recognition*, vol. 44, no. 8, pp. 1629–1648, 2011.

[22] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using laplacianfaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 3, pp. 328–340, 2005.

[23] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 10, pp. 1713–1727, 2008.

[24] E. W. Weisstein, "Homeomorphic," http://mathworld.wolfram.com/Homeomorphic.html, 2016, [Online; accessed 30-Aug-2016].

[25] P. Comon, "Independent component analysis," *Higher-order statistics*, pp. 29–38, 1992.

[26] Z. Liang and Y. Lee, "Eigen-analysis of nonlinear pca with polynomial kernels," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 6, no. 6, pp. 529–544, 2013.

[27] M. A. Cox and T. F. Cox, "Multidimensional scaling," *Handbook of Data Visualization*, pp. 315–347, 2008.

[28] J. Tenenbaum, V. D. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[29] K. Saul and S. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *The Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[30] S. Roweis and K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[31] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[32] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[33] W. Chojnacki and M. J. Brooks, "A note on the locally linear embedding algorithm," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 08, pp. 1739–1752, 2009.

[34] M. Insall, T. Rowland, and E. W. Weisstein, "Embedding," http://mathworld.wolfram.com/Embedding.html, 2016, [Online; accessed 30-Aug-2016].

[35] D. Gluss and E. W. Weisstein, "Lagrange Multiplier," http://mathworld.wolfram.com/LagrangeMultiplier.html, 2016, [Online; accessed 30-Aug-2016].

[36] H.-D. Cheng, X. Jiang, Y. Sun, and J. Wang, "Color image segmentation: advances and prospects," *Pattern recognition*, vol. 34, no. 12, pp. 2259–2281, 2001.

[37] W. ladys law Skarbek and A. Koschan, "Colour image segmentation a survey," *IEEE Transactions on circuits and systems for Video Technology*, vol. 14, no. 7, 1994.

[38] G. Dirichlet, "Über die reduktion der positiven quadratischen formen mit drei unbestimmten ganzen zahlen," *J. für die reine und angewandte Math.*, vol. 40, pp. 209–227, 1850.

[39] G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques," *J. für die reine und angewandte Math.*, vol. 133, pp. 97–178, 1907.

[40] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[41] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

[42] Q. Du, M. Gunzburger, L. Ju, and X. Wang, "Centroidal voronoi tessellation algorithms for image compression, segmentation, and multichannel restoration," *Journal of Mathematical Imaging and Vision*, vol. 24, no. 2, pp. 177–194, 2006.

[43] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "On points geometry for fast digital image segmentation," in *The 8th International Conference on Information Technology and Telecommunication. Ireland: IEEE*, 2008, pp. 54–61.

[44] M. Suhail, M. Obaidat, S. Ipson, and B. Sadoun, "Content-based image segmentation," in *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, vol. 5. IEEE, 2002, pp. 6–pp.

[45] S. Bilal, R. Akmeliawati, M. J. E. Salami, and A. A. Shafie, "Dynamic approach for real-time skin detection," *Journal of Real-Time Image Processing*, vol. 10, no. 2, pp. 371–385, 2015.

[46] K. Radlak and B. Smolka, "A novel approach to the eye movement analysis using a high speed camera," in *Advances in Computational Tools for Engineering Applications (ACTEA), 2012 2nd International Conference on.* IEEE, 2012, pp. 145–150.

[47] J.-S. Lee, Y.-M. Kuo, P.-C. Chung, and E.-L. Chen, "Naked image detection based on adaptive and extensible skin color model," *Pattern recognition*, vol. 40, no. 8, pp. 2261–2270, 2007.

[48] J. Stöttinger, A. Hanbury, C. Liensberger, and R. Khan, "Skin paths for contextual flagging adult videos," in *International symposium on visual computing.* Springer, 2009, pp. 303–314.

[49] L. Daxiang, Z. Xiaoqiang, and L. Ying, "Pornographic images filtering using pmk-based mil algorithm," *Int. J. Comput. Sci*, vol. 10, no. 3, 2013.

[50] L. Ballerini, X. Li, R. B. Fisher, and J. Rees, "A query-by-example content-based image retrieval system of non-melanoma skin lesions," in *MICCAI International Workshop on Medical Content-Based Retrieval for Clinical Decision Support.* Springer, 2009, pp. 31–38.

[51] A. Baldi, R. Murace, E. Dragonetti, M. Manganaro, and S. Bizzi, "Automated content-based image retrieval: Application on dermoscopic images of pigmented skin lesions," in *Skin Cancer.* Springer, 2014, pp. 523–528.

[52] M. Silveira, J. C. Nascimento, J. S. Marques, A. R. Marçal, T. Mendonça, S. Yamauchi, J. Maeda, and J. Rozeira, "Comparison of segmentation methods for melanoma diagnosis in dermoscopy images," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, no. 1, pp. 35–45, 2009.

[53] L. Ballerini, R. B. Fisher, B. Aldridge, and J. Rees, "A color and texture based hierarchical k-nn approach to the classification of non-melanoma skin lesions," in *Color Medical Image Analysis.* Springer, 2013, pp. 63–86.

[54] M.-J. Chen, M.-C. Chi, C.-T. Hsu, and J.-W. Chen, "Roi video coding based on h. 263+ with robust skin-color detection technique," *Consumer Electronics, IEEE Transactions on*, vol. 49, no. 3, pp. 724–730, 2003.

[55] S.-F. Huang, M.-J. Chen, and M.-S. Li, "Region-of-interest segmentation based on bayesian theorem for h. 264 video transcoding," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*. IEEE, 2011, pp. 1–4.

[56] F. Saxen and A. Al-Hamadi, "Color-based skin segmentation: An evaluation of the state of the art," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4467–4471.

[57] M. Kawulok, J. Nalepa, and J. Kawulok, "Skin detection and segmentation in color images," in *Advances in Low-Level Color Image Processing*. Springer, 2014, pp. 329–366.

[58] Y. Shen, S.-K. Ong, and A. Y. Nee, "Vision-based hand interaction in augmented reality environment," *Intl. Journal of Human–Computer Interaction*, vol. 27, no. 6, pp. 523–544, 2011.

[59] X. Chai, G. Li, Y. Lin, Z. Xu, Y. Tang, X. Chen, and M. Zhou, "Sign language recognition and translation with kinect," in *IEEE Conf. on AFGR*, 2013.

[60] S. Lang, M. Block, and R. Rojas, "Sign language recognition using kinect," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2012, pp. 394–402.

[61] I. Hussain, A. K. Talukdar, and K. K. Sarma, "Hand gesture recognition system with real-time palm tracking," in *2014 Annual IEEE India Conference (INDICON)*. IEEE, 2014, pp. 1–6.

[62] E. ul Haq, S. J. H. Pirzada, M. W. Baig, and H. Shin, "New hand gesture recognition method for mouse operations," in *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2011, pp. 1–4.

[63] M. Czupryna and M. Kawulok, "Real-time vision pointer interface," in *ELMAR, 2012 Proceedings*. IEEE, 2012, pp. 49–52.

[64] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, and J. Handler, "A real-time hand gesture interface for medical visualization applications," in *Applications of Soft Computing*. Springer, 2006, pp. 153–162.

[65] M. Gutiérrez, P. Lemoine, D. Thalmann, and F. Vexo, "Telerehabilitation: controlling haptic virtual environments through handheld interfaces," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 2004, pp. 195–200.

[66] R. Boian, A. Sharma, C. Han, A. Merians, G. Burdea, S. Adamovich, M. Recce, M. Tremaine, H. Poizner *et al.*, "Virtual reality-based post-stroke hand rehabilitation," *Studies in health technology and informatics*, pp. 64–70, 2002.

[67] T. Starner, B. Leibe, B. Singletary, and J. Pair, "Mind-warping: towards creating a compelling collaborative augmented reality game," in *Proceedings of the 5th international conference on Intelligent user interfaces*. ACM, 2000, pp. 256–259.

[68] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[69] M. Soltanolkotabi, E. Elhamifar, E. J. Candes *et al.*, "Robust subspace clustering," *The Annals of Statistics*, vol. 42, no. 2, pp. 669–699, 2014.

[70] M. Fan, H. Qiao, B. Zhang, and X. Zhang, "Isometric multi-manifold learning for feature extraction," in *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*. IEEE Computer Society, 2012, pp. 241–250.

[71] W. Yang, C. Sun, and L. Zhang, "A multi-manifold discriminant analysis method for image feature extraction," *Pattern Recognition*, vol. 44, no. 8, pp. 1649–1657, 2011.

[72] X. Niyogi, "Locality preserving projections," in *Neural information processing systems*, vol. 16, 2004, p. 153.

[73] A. B. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak, "Multi-manifold semi-supervised learning," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 169–176.

[74] J. Lu, Y.-P. Tan, and G. Wang, "Discriminative multimanifold analysis for face recognition from a single training sample per person," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 39–51, 2013.

[75] Y. H. Pang, A. Teoh, E. K. Wong, and F. S. Abas, "Supervised locally linear embedding in face recognition," in *Biometrics and Security Technologies, 2008. ISBAST 2008. International Symposium on.* IEEE, 2008, pp. 1–6.

[76] D. Liang, J. Yang, Z. Zheng, and Y. Chang, "A facial expression recognition system based on supervised locally linear embedding," *Pattern Recognition Letters*, vol. 26, no. 15, pp. 2374–2389, 2005.

[77] B. Geng, D. Tao, C. Xu, Y. Yang, and X.-S. Hua, "Ensemble manifold regularization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 6, pp. 1227–1233, 2012.

[78] E. Čech, *Topological Spaces, revised Ed. by Z. Frolik and M. Katětov.* London: John Wiley & Sons, 1966, 893pp.

[79] A. D. Concilio, "Proximity: A powerful tool in extension theory, function spaces, hyperspaces, boolean algebras and point-free geometry," *Contemporary Math.*, vol. 486, pp. 89–114, 2009.

[80] J. F. Peters, "Topology of digital images: Basic ingredients," in *Topology of Digital Images. Visual Pattern Discovery in Proximity Spaces, Intelligent Systems Reference Library 63.* Springer, 2014, pp. 1–75, xv + 411pp, Zentralblatt MATH Zbl 1295 68010.

[81] R. Wang, S. Shan, X. Chen, Q. Dai, and W. Gao, "Manifold–manifold distance and its application to face recognition with image sets," *Image Processing, IEEE Transactions on*, vol. 21, no. 10, pp. 4466–4479, 2012.

[82] M. Zhang and A. A. Sawchuk, "Manifold learning and recognition of human activity using body-area sensors," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2. IEEE, 2011, pp. 7–13.

[83] O. Kayo, "Locally linear embedding algorithm: extensions and applications." Ph.D. dissertation, Faculty of Technology, University of Oulu, 2006.

[84] "AT&T Laboratories Cambridge, Cambridge University Computer Laboratory, The ORL Database of Faces," http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, 2002, [Online; accessed 23-Dec-2014].

[85] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on.* IEEE, 1994, pp. 138–142.

[86] "Image Engineering Laboratory, Sheffield Face Database," https://www.sheffield.ac.uk/eee/research/iel/research/face, [Online; accessed 23-Dec-2014].

[87] W. Liu, D. Tao, J. Cheng, and Y. Tang, "Multiview hessian discriminative sparse coding for image annotation," *Computer Vision and Image Understanding*, vol. 118, pp. 50–60, 2014.

[88] W. Liu and D. Tao, "Multiview hessian regularization for image annotation," *Image Processing, IEEE Transactions on*, vol. 22, no. 7, pp. 2676–2687, 2013.

[89] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[90] R. L. Cannon, J. V. Dave, and J. C. Bezdek, "Efficient implementation of the fuzzy c-means clustering algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 248–255, 1986.

[91] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *Information Theory, IEEE Transactions on*, vol. 21, no. 1, pp. 32–40, 1975.

[92] Z. Yu, O. C. Au, R. Zou, W. Yu, and J. Tian, "An adaptive unsupervised approach toward pixel clustering and color image segmentation," *Pattern Recognition*, vol. 43, no. 5, pp. 1889–1906, 2010.

[93] K. S. Tan, N. A. M. Isa, and W. H. Lim, "Color image segmentation using adaptive unsupervised clustering approach," *Applied Soft Computing*, vol. 13, no. 4, pp. 2017–2036, 2013.

[94] A. Colorni, M. Dorigo, V. Maniezzo *et al.*, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142. Paris, France, 1991, pp. 134–142.

[95] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *ICML*, vol. 1, 2000.

[96] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[97] S. R. Vantaram and E. Saber, "An adaptive bayesian clustering and multivariate region merging based technique for efficient segmentation of color images," in *Acous-*

tics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011, pp. 1077–1080.

[98] M. Sujaritha and S. Annadurai, "Color image segmentation using adaptive spatial gaussian mixture model," *International journal of signal processing*, vol. 6, no. 1, pp. 28–32, 2010.

[99] C. Rosenberger and K. Chehdi, "Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 1. IEEE, 2000, pp. 656–659.

[100] D. E. Ilea and P. F. Whelan, "Ctexan adaptive unsupervised segmentation algorithm based on color-texture coherence," *Image Processing, IEEE Transactions on*, vol. 17, no. 10, pp. 1926–1939, 2008.

[101] L. G. Ugarriza, E. Saber, S. R. Vantaram, V. Amuso, M. Shaw, and R. Bhaskar, "Automatic image segmentation by dynamic region growth and multiresolution merging," *Image Processing, IEEE Transactions on*, vol. 18, no. 10, pp. 2275–2288, 2009.

[102] K. Bhoyar and O. Kakde, "Color image segmentation based on jnd color histogram," *International Journal of Image Processing (IJIP)*, vol. 3, no. 6, p. 283, 2010.

[103] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[104] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer vision, graphics, and image processing*, vol. 29, no. 1, pp. 100–132, 1985.

[105] H. Zhang, J. E. Fritts, and S. A. Goldman, "Image segmentation evaluation: A survey of unsupervised methods," *computer vision and image understanding*, vol. 110, no. 2, pp. 260–280, 2008.

[106] J. F. Peters, "Computational proximity. excursions in the topology of digital images." *Intelligent Systems Reference Library*, 2016, in press.

[107] C. Romesburg, *Cluster analysis for researchers*. Lulu Press, 2004.

[108] M. J. Norušis, *IBM SPSS statistics 19 statistical procedures companion*. Prentice Hall, 2012.

[109] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[110] S. Ghosh and S. K. Dubey, "Comparative analysis of k-means and fuzzy c-means algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, pp. 34–39, 2013.

[111] J. Liu and Y.-H. Yang, "Multiresolution color image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 7, pp. 689–700, 1994.

[112] M. Borsotti, P. Campadelli, and R. Schettini, "Quantitative evaluation of color image segmentation results," *Pattern recognition letters*, vol. 19, no. 8, pp. 741–747, 1998.

[113] C. Rosenberger and K. Chehdi, "Genetic fusion: application to multi-components image segmentation," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 6. IEEE, 2000, pp. 2223–2226.

[114] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, 2002.

[115] X. Wang, X. Zhang, and J. Yao, "Skin color detection under complex background," in *Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on*. IEEE, 2011, pp. 1985–1988.

[116] A. Y. Taqa and H. A. Jalab, "Increasing the reliability of skin detectors," *Scientific Research and Essays*, vol. 5, no. 17, pp. 2480–2490, 2010.

[117] P. Ng and C.-M. Pun, "Skin color segmentation by texture feature extraction and k-mean clustering," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on*. IEEE, 2011, pp. 213–218.

[118] M. Kawulok, J. Kawulok, and J. Nalepa, "Spatial-based skin detection using discriminative skin-presence features," *Pattern Recognition Letters*, vol. 41, pp. 3–13, 2014.

[119] M. Kawulok, J. Kawulok, J. Nalepa, and B. Smolka, "Self-adaptive algorithm for segmenting skin regions," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, pp. 1–22, 2014.

[120] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.

[121] H. He and E. A. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[122] V. Garcia, J. S. Sanchez, R. A. Mollineda, R. Alejo, and J. M. Sotoca, "The class imbalance problem in pattern classification and learning," in *II Congreso Español de Informática (CEDI 2007). ISBN*. Citeseer, 2007, pp. 978–84.

[123] J. Kovač, P. Peer, and F. Solina, *Human skin color clustering for face detection*. IEEE, 2003, vol. 2.

[124] K. Sobottka and I. Pitas, "Face localization and facial feature extraction based on shape and color information," in *Image Processing, 1996. Proceedings., International Conference on*, vol. 3. IEEE, 1996, pp. 483–486.

[125] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 696–706, 2002.

[126] P. Yogarajah, J. Condell, K. Curran, P. McKevitt, and A. Cheddad, "A dynamic threshold approach for skin tone detection in colour images," *International Journal of Biometrics*, vol. 4, no. 1, pp. 38–55, 2011.

[127] K. Bhoyar and O. Kakde, "Skin color detection model using neural networks and its performance evaluation 1," 2010.

[128] M. Kawulok, "Fast propagation-based skin regions segmentation in color images," in *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE, 2013, pp. 1–7.

[129] G. A. Seber, *Multivariate observations*. John Wiley & Sons, 2009, vol. 252.

[130] P. Radtke, E. Granger, R. Sabourin, and D. Gorodnichy, "Adaptive ensemble selection for face re-identification under class imbalance," in *Multiple Classifier Systems*. Springer, 2013, pp. 95–108.

[131] G. M. Weiss and F. Provost, "Learning when training data are costly: the effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, pp. 315–354, 2003.

[132] A. Y.-c. Liu, "The effect of oversampling and undersampling on classifying imbalanced text datasets," Ph.D. dissertation, Citeseer, 2004.

[133] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 839–846.

[134] P. P. Kumar, P. Vadakkepat, and A. P. Loh, "Hand posture and face recognition using a fuzzy-rough approach," *International Journal of Humanoid Robotics*, vol. 7, no. 03, pp. 331–356, 2010.

[135] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[136] S. Marcel and O. Bernier, "Hand posture recognition in a body-face centered space," in *International Gesture Workshop*. Springer, 1999, pp. 97–100.

[137] C.-C. Wang and K.-C. Wang, "Hand posture recognition using adaboost with sift for human robot interaction," in *Recent progress in robotics: viable robotic service to human*. Springer, 2007, pp. 317–329.

[138] X. Yin and M. Xie, "Finger identification and hand posture recognition for human–robot interaction," *Image and Vision Computing*, vol. 25, no. 8, pp. 1291–1300, 2007.

[139] Y. Fang, K. Wang, J. Cheng, and H. Lu, "A real-time hand gesture recognition method," in *2007 IEEE International Conference on Multimedia and Expo*. IEEE, 2007, pp. 995–998.

[140] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 1565–1569.

[141] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 759–760.

[142] J. Suarez and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 411–417.

[143] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[144] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Transactions on computers*, vol. 100, no. 3, pp. 269–281, 1972.

[145] H. Choi, B. Paulson, and T. Hammond, "Gesture recognition based on manifold learning," in *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2008, pp. 247–256.

[146] S. S. Ge, Y. Yang, and T. H. Lee, "Hand gesture recognition and tracking based on distributed locally linear embedding," *Image and Vision Computing*, vol. 26, no. 12, pp. 1607–1620, 2008.

[147] J. Nalepa, T. Grzejszczak, and M. Kawulok, "Wrist localization in color images for hand gesture recognition," in *Man-Machine Interactions 3*. Springer, 2014, pp. 79–86.

[148] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, pp. 411–426, 2007.

[149] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 994–1000.

[150] "The NUS Hand Posture Dataset II," http://www.vadakkepat.com/NUS-HandSet/, 2013, [Online; accessed 17-Aug-2015].

[151] "Jochen Triesch Static Hand Posture Database," http://www.idiap.ch/resource/gestures/, 1996, [Online; accessed 8-Jul-2016].

[152] J. Trisch and C. Malsburg, "Robust classification of hand postures against complex background," in *Proc. International Conference On Automatic Face and Gesture Recognition*, 1996.

[153] P. Peixoto, J. Carreira *et al.*, "A natural hand gesture human computer interface using contour signatures," in *Proceedings of the IASTED HCI Conference*, vol. 476, 2005.

[154] Y. Wang, Z. Luo, J. Liu, X. Fan, H. Li, and Y. Wu, "Real-time estimation of hand gestures based on manifold learning from monocular videos," *Multimedia Tools and Applications*, vol. 71, no. 2, pp. 555–574, 2014.

[155] F.-S. Chen, C.-M. Fu, and C.-L. Huang, "Hand gesture recognition using a real-time tracking method and hidden markov models," *Image and vision computing*, vol. 21, no. 8, pp. 745–758, 2003.

[156] A. Ramamoorthy, N. Vaswani, S. Chaudhury, and S. Banerjee, "Recognition of dynamic hand gestures," *Pattern Recognition*, vol. 36, no. 9, pp. 2069–2081, 2003.

[157] Z. Yang, Y. Li, W. Chen, and Y. Zheng, "Dynamic hand gesture recognition using hidden markov models," in *Computer Science & Education (ICCSE), 2012 7th International Conference on.* IEEE, 2012, pp. 360–365.

[158] X. Shen, G. Hua, L. Williams, and Y. Wu, "Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields," *Image and Vision Computing*, vol. 30, no. 3, pp. 227–235, 2012.

[159] A. Kurakin, Z. Zhang, and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European.* IEEE, 2012, pp. 1975–1979.

[160] F. Riaz, A. Hassan, S. Rehman, and U. Qamar, "Texture classification using rotation- and scale-invariant gabor texture features," *IEEE Signal Processing Letters*, vol. 20, no. 6, pp. 607–610, 2013.