Low Dimension Hierarchical Subspace Modelling of High Dimensional Data

Oksana Samko

Cardiff University
School of Computer Science

June 2009

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Table of Contents

List of Tables

List of Figures

Abstract

Building models of high-dimensional data in a low dimensional space has become extremely popular in recent years. Motion tracking, facial animation, stock market tracking, digital libraries and many other different models have been built and tuned to specific application domains. However, when the underlying structure of the original data is unknown, the modelling of such data is still an open question. The problem is of interest as capturing and storing large amounts of high dimensional data has become trivial, yet the capability to process, interpret, and use this data is limited.

In this thesis, we introduce novel algorithms for modelling high dimensional data with an unknown structure, which allows us to represent the data with good accuracy and in a compact manner. This work presents a novel fully automated dynamic hierarchical algorithm, together with a novel automatic data partitioning method to work alongside existing specific models (talking head, human motion). Our algorithm is applicable to hierarchical data visualisation and classification, meaningful pattern extraction and recognition, and new data sequence generation. Also during our work we investigated problems related to low dimensional data representation: automatic optimal input parameter estimation, and robustness against noise and outliers. We show the potential of our modelling with many data domains: talking head, motion, audio, *etc.* and we believe that it has good potential in adapting to other domains.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Dr. Dave Marshall and Dr. Paul Rosin, for their guidance, suggestions and patience in helping me to complete this work. It would have been an impossible task without their advice and support.

I would like to express my sincere thanks to Dr Darren Cosker and Dr Yulia Hicks for helping me to understand the numerous techniques I had to become familiar with when I first started, and for providing me with some of their data. I would like to thank Benjamin Havell for patiently reading through my thesis, and providing feedback that has served to improve this thesis. I would like to extend my gratitude to the Computer Vision Group staff for their support all these years.

Although the PhD is a journey of research, no journey can be travelled alone. I would like to thank my friends and family for the encouragement their have given me, who believed in me even when I had doubts.

Finally I thank all others who have directly and indirectly helped me in this thesis work in the department and outside the department.

Chapter 1

Introduction

1.1 Motivation

Digital multimedia data has grown greatly in recent years along with the rapid development of attendant technologies. The growing complexity and insufficiency of existing tools for managing this data expansion have highlighted the need for better tools and techniques, identifying and recognising meaningful content among the principal goals of multimedia analysis. This is however extremely difficult, given that it depends on many open issues in computer vision.

This thesis is dedicated to the study and development of methods involved in hierarchical modelling of high dimensional digital data in a low dimensional subspace. These include low dimensional subspace modelling, hierarchical data representation, temporal dynamic framework and sub-parts data decomposition.

In order to construct a dynamic model, we need sufficient training data to learn computational representations for the data domain. The initial digital data is typically represented by either raw data vectors in a multidimensional space or, alternatively, by some features that describe aspects of the data. Either way, the initial data required for processing typically is high dimensional. Many techniques have been

developed to attempt to represent the data in a lower dimensional subspace. Nevertheless, problems in the usually high non-linearity of the original data mean that computing effective low dimensional representations is still a major research area.

A hierarchical approach to data decomposition allows greater accuracy when representing non-linear data from the real world. The hierarchy helps to locate the required data point among all the points, as at each level of the tree one can discard large sets of clusters and only explore the more promising clusters. Many papers have been published on hierarchical data decomposition [11, 23, 71, 87]. However, the current models are not built automatically and require some parameters to be set manually.

A recent paradigm of dynamic data modelling includes Hierarchical Hidden Markov Models (HHMMs). HHMMs are well suited to handle uncertainty and sequential decision making with reasonable computational efficiency [94]. As a result, the application of HHMMs to dynamic data representation has gained great popularity which is rapidly growing, especially in semantic content identification. Given the high suitability of HHMM data modelling, it is extremely desirable to develop machine learning algorithms which can learn such models from training data automatically and effectively.

General models can process high dimensional data with good accuracy, but modelling data with no a priori knowledge with a general algorithm can be insufficient in some applications. In such cases one can model the data using existing model-based algorithms. To make this process automated, one needs an effective algorithm for automatic data partitioning.

In this thesis, we address the above problems and present novel techniques for

accurate modelling of high dimensional data with no a priori knowledge.

1.2 Applications

Because our methods are not using any a priori data information, they have a broad application area.

In this thesis we demonstrate our methods' ability to work with different formats of motion capture (mocap) data: angular and 3D coordinates, video frames. Mocap systems are extremely popular and widely applicable in film and game making, military, entertainment, sports, and medicine. In particular, applications include activity recognition, visual surveillance, anomaly detection and alarms. Using our model one can evaluate human motion without special body suits or tracking devices.

In addition to the data domains tested, the new methods presented here are applicable to a much wider scope, including medical data analysis, such as DNA or protein structure analysis; stock trading in financial analysis; biometric authentication.

Moreover, our model is flexible and it is possible to use parts of our system as a part of other models.

1.3 Research Overview

The problem which we solve in this thesis is simply stated as:

Given input data, devise efficient and robust algorithms that can automatically decompose the initially high dimensional space into a reduced dimension, hierarchical, non-linear subspace.

This problem is extremely hard to solve if we know nothing about the input data. We do not use any underlying data information for the model construction, in contrast

to the models based on data with a priori knowledge such as talking head or human motion models. We have to assume that the training data fully represents the data's intrinsic topology and convex, and our model is based on these assumptions.

In this thesis we introduce novel methods which were created to solve the problem stated above. Figure 1.1 shows an overview of the major processes for our methods. There are two branches: the main one based on investigating the data's intrinsic topology using Isomap [118], and the additional one for finding a partial decomposition of the data using Nonnegative matrix factorisation (NMF) [80]. In the main part of our model we create the hierarchical algorithm which is useful for hierarchical data visualisation and classification. Also in the case of several data sets one can estimate the corresponding data points having new unseen points with this model. Using the hierarchical algorithm, we develop a dynamic framework to create the ability for data pattern extraction and recognition, and new data sequence generation. In the additional part of our algorithm we implement a novel automatic data partitioning method to work alongside existing specific models (talking head, human motion). Experimental results demonstrate that the models produced using the above methods represent the original data correctly and accurately.

1.4 Major Contributions

In summary, the major contributions of this work are:

- A novel algorithm for automatic parameter setting for the Isomap algorithm;
- A novel method for construction of a hierarchical model using clustering and non-linear mapping;

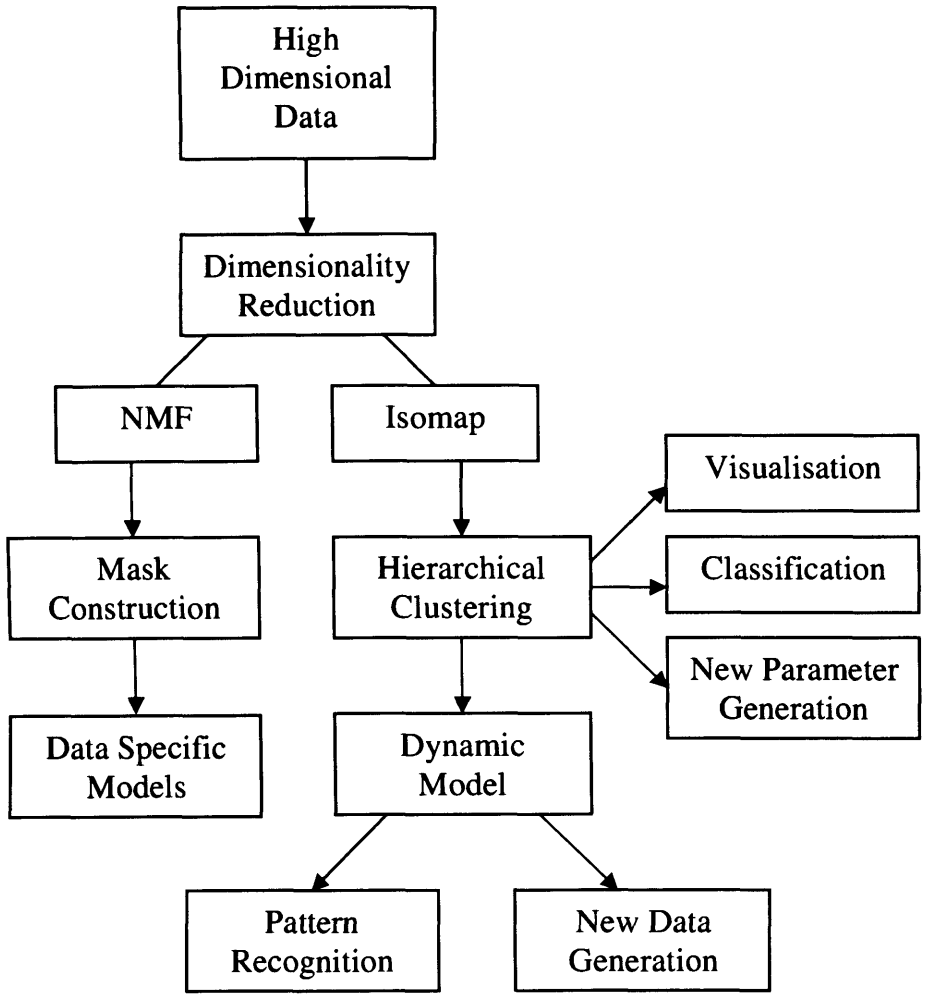

Figure 1.1: An overview of the major processes described in this thesis

- A novel method for automatic construction of a Hierarchical Hidden Markov Model structure for discovering semantic data patterns;
- A novel automatic method for learning a meaningful sub-part representation from real world data with an unknown a priori structure.

1.5 Thesis Organisation

The structure of this thesis is as follows:

- In Chapter 2 we give a review of relevant literature relating to dynamic data modelling. In particular the review covers work on dimensionality reduction problems, data intrinsic dimensionality estimation, hierarchical clustering and dynamic data representation.
- Chapter 3 describes the dimensionality reduction routine: the Isomap algorithm together with related problems and ways to solve them, such as a new data sampling into embedded space and Isomap back projection. We present a new algorithm for automatically setting the Isomap parameter and show numerous experiments on synthetic and real data sets to evaluate the performance of our method.
- In Chapter 4 we address how a parameterised data set is used to construct a hierarchical model in the low-dimensional space, with descriptions of each of its major processes, and show examples of the model's application to different data obtained from various sources (video, audio, images, motion capture). We consider the case of having several joint data sets at the same time and having new unseen point estimation of the corresponding data point with our model.

- Chapter 5 outlines developing a dynamic framework with the structure obtained using a static hierarchical model. Specifically it describes how to apply the hierarchy constructed in the previous chapter, to Hierarchical Hidden Markov Model (HHMM) structure definition. We present training and testing processes for the constructed hierarchial dynamic model which include representation of the HHMM as a Dynamic Bayesian Network (DBN) [94].

- In Chapter 6 we present the applications of our automatic model from the previous chapter to real world data. We perform data pattern classification, new data synthesis and reconstruction of the high-dimensional data from the model.

- Chapter 7 relates to the automatic partial data representation method, which one can use further to work with algorithms specifically created for certain applications. We demonstrate the effectiveness of our algorithm with several examples.

- In Chapter 8 we give a conclusion and present our future work, particularly a modified Isomap algorithm to deal with noise and outliers.

1.6 Publications

The research described in this thesis is based on the following publications:

- O.Samko, P.L.Rosin, D.Marshall "Robust automatic data decomposition using a modified sparse NMF". Proc. of Mirage 2007: 225-234.

- O.Samko, D.Marshall, P.L.Rosin "Selection of the optimal parameter value for the Isomap algorithm". Pattern Recognition Letters, 27(9):968-976, 2006.

- O.Samko, D.Marshall, P.L.Rosin "Automatic construction of Hierarchical Hidden Markov Model structure for discovering semantic patterns in motion data". Submitted to VISAPP 2010.

- O.Samko, D.Marshall, P.L.Rosin "Automatic construction of a hierarchical model using clustering and non-linear mapping". Submitted to Pattern Recognition Letters.

- O.Samko, P.L.Rosin, D.Marshall "Robust Isomap algorithm for temporal trajectory extraction from a noisy high dimensional data". (In preparation).

Chapter 2

Literature Review

The problem of hierarchical data modelling closely relates to research in several areas. In Section 2.1 we investigate dimensionality reduction methods, compare them and give application examples. Section 2.2 is related to Section 2.1 and describes the state of the art in techniques for the estimation of data's intrinsic dimensionality. Section 2.3 includes an overview of the clustering techniques, which we applied in the hierarchy construction. In Section 2.4 we discuss research efforts in dynamic frameworks and Hierarchical Hidden Markov Models, which we use to represent the dynamic variations in our model.

2.1 Reducing Dimensionality of the Data Space

2.1.1 Review of Dimensionality Reduction Techniques

There are a number of dimensionality reduction techniques, that allow users to analyse and visualise complex data sets better in comparison to the original high dimensional space. These techniques may be separated into two classes, linear and nonlinear. Examples of linear methods are Principal Component Analysis (PCA, [70]) or the

original metric multidimensional scaling (MDS, [121]). Isomap [118], Locally Linear Embedding (LLE) [109], Hessian LLE [39], Laplacian eigenmaps [4], Geodesic Nullspace Analysis (GNA) [15], Semidefinite Embedding (SDE) [127] and nonlinear PCA [72] are nonlinear examples. Nonlinear algorithms are used to reveal low dimensional manifolds that are not detected by classical and well known linear methods, thus they represent the original data more accurately. In our work we investigated methods from both these classes to choose the optimal one. Here we present a short review of dimensionality reduction techniques, their comparison and related problems.

The classical techniques for dimensionality reduction, PCA and MDS, are simple to implement and efficiently computable. Geometrically, PCA rotates the data so that its primary axes lie along the axes of the coordinate space and move it so that its centre of mass lies on the origin. Classical MDS aims to represent the data points in a lower dimensional space while preserving as much of the pairwise similarities between the data points as possible.

Recently, several new promising distance-preserving methods such as Isomap and LLE, have been proposed. The Isomap algorithm of Tenenbaum et al. [118, 113] extends MDS by a sophisticated distance measurement to achieve nonlinear embedding. They build a graph on the data that is only locally connected, and then measure pairwise distances by the length of the shortest path on that graph. This length is an approximation to the distance between its end points, as measured within the underlying manifold. Finally, MDS is used to find a set of low-dimensional points with similar pairwise distances. Isomap is a *global* method, i.e. it constructs an embedding derived from the geodesic distance between all pairs of points.

The Isomap algorithm is well understood and produces reasonable mapping results, so it has become very popular. There are theoretical studies supporting the use of Isomap, such as its convergence proof [8]. Also, in [38] Donoho and Grimes study which kinds of phenomena Isomap can truly recover the underlying structure of. Apart from the original Isomap, which is characterised by the free parameter K or ϵ (number of neighbours or neighbour region size), there exist several Isomap modifications for different data cases [78, 114].

The Locally Linear Embedding (LLE) algorithm of Roweis and Saul [109] ([110] in more detail) computes a different local quantity. They calculate the best coefficients to approximate each point by a weighted linear combination of its neighbours, and then try to find a set of low-dimensional points, which can be linearly approximated by its neighbours with the same coefficients that were determined from the high-dimensional points. LLE is a *local* method, because its cost function only considers the placement of each point with respect to its neighbors. As a local method, it tends to characterize the local geometry of manifolds accurately, but breaks down at the global level, and therefore is not suitable for the global data analysis.

Belkin and Niyogi [4] proposed the Laplacian Eigenmaps technique. Again the neighbourhood graph is determined, then the graph Laplacian is computed (as an approximation to the manifold Laplacian). The embedded coordinates are then found as the eigenvectors of the null space of a quadratic form made up of the average of the Laplacian operator over the manifold. Similar to LLE, it is a *local* method.

Donoho and Grimes [39] proposed Hessian Eigenmaps (hLLE). Here the Laplacian is replaced by the Hessian. For this method the assumptions are relaxed from isometry of the mapping between the high-dimensional and embedded spaces and

convexity of the parameter space, to local isometry of the mapping and connectedness of the parameter space. A drawback of Hessian LLE versus the other methods is the Hessian approach requires estimation of second derivatives, and this is known to be numerically noisy or difficult for very high-dimensional data samples.

Weinberger and Saul [127] proposed Semidefinite Embedding (SDE) - a new method based on semidefinite programming. After constructing the graph, they compute the Gram matrix of the maximum variance embedding that is centered on the origin and preserves the distances of all edges in the neighborhood graph. At the last step they extract a low dimensional embedding from the dominant eigenvectors of the Gram matrix learned by semidefinite programming. One major drawback of SDE is that solving a semidefinite program requires huge computational resources.

Comparing Isomap, LLE, Hessian LLE and Laplacian eigenmaps algorithms (see Figure 2.1), we find that each algorithm attempts to estimate and preserve a different geometric signature of the manifold sampled by the inputs. Isomap estimates geodesic distances between inputs; LLE estimates the coefficients of local linear reconstructions; hLLE and Laplacian eigenmaps estimate the Hessian and Laplacian on the manifold, respectively; SDE estimates local angles and distances. Of these algorithms, only Isomap, hLLE, and SDE attempt to learn isometric embeddings; they are therefore the easiest to compare.

Overall, the different algorithms for manifold learning in Figure 2.1 should be viewed as complementary; each has its own advantages and disadvantages. LLE, hLLE, and Laplacian eigenmaps construct sparse matrices, and as a result, they are easier to scale to large data sets. On the other hand, their eigenvalue spectra do not reliably reveal the underlying dimensionality of sampled manifolds, as Isomap and

Method	Matrix	Mapping	Signature	Manifold
Isomap	dense	isometric	geodesic distances	open, geodesically convex
SDE	dense	isometric	local distances	open, connected
LLE	sparse	conformal	local angles	manifold that can be conformally mapped
hLLE	sparse	isometric	Hessian	open, connected
Laplacian eigenmaps	sparse	proximity preserving	discrete Laplacian	open, convex

Figure 2.1: Comparison of manifold learning algorithms in terms of the matrices they compute, the mappings they learn, the geometric signatures they exploit, and types of manifolds for which the method works

SDE do. For the latest survey on manifold leaning techniques see [101].

In this thesis, we use Isomap as a dimensionality reduction technique, because its algorithm is based on the assumption that the input high-dimensional data lies on an intrinsically smooth manifold in a low-dimensional space [8]. This assumption fits with the interframe correlation in the original data sequence, that help to maintain the dynamic variations of the data. Moreover, Isomap also advantageous in the sense that it can be computed faster than other manifold learning methods.

2.1.2 Applications of Nonlinear Dimensionality Reduction Techniques

Nonlinear dimensionality techniques, especially Isomap and LLE, are widely used for various applications. Lim et al. [84] applied Isomap for visualisation of large data sets. Kouropteva et al. [76] used modified LLE combined with a support vector machine to classify handwritten digits. Tsai et al. [122] applied Isomap and LLE to pose estimation. After trying these techniques for pose estimation, the authors

conclude that Isomap has a higher accuracy rate than LLE.

In [100], image frames from videos of natural periodic motions were analysed using Isomap embedding. They find that the relationships between the Isomap embedded parameters accurately describe the relationships between the video frames, and that they can be used for intuitive video segmentation. In [69], a similar approach was used.

Isomap has been recently applied to gesture and motion recognition. Li et al. employed Isomap to the CMU gait image database [82] to classify types of walk (slow, fast, etc.). Blackburn and Rebeiro employed Isomap and Dynamic Time Warping (DTW) for motion recognition [10]. They used Isomap to reduce the dimensionality of individual frames and then DTW was employed to match embedded manifolds to test sequences.

2.1.3 Isomap Algorithm Problems and Existing Solutions

There are a few problems with the original Isomap algorithm. Isomap is quite sensitive to outliers, closed manifold, manifolds with holes, varying density manifolds and large manifolds. A review of the Isomap algorithm modifications related to these problems is presented here.

Possible Topological Instability

Balasubramanian and Schwartz [3] accused Isomap of being topologically unstable by showing an example where ϵ-Isomap succeeded without noise, but failed with added noise for constant ϵ. Naturally, Tenenbaum et al. [119] showed that with added noise different branches of the manifold might get connected and hence a smaller neighbourhood size resolves the spurious problem. Thus they pointed out the

importance of choosing the right Isomap parameter; in this thesis we propose a novel method for optimal Isomap parameter selection, see Section 3.5.

Large Data Sets Problem and Out-of-Sample Projection

For a large data set of N points the main bottleneck is the classical MDS stage, which reduces the task to an $O(N^3)$ eigenvalue problem on a dense $N \times N$ matrix. Storage of the full distance matrix is also an issue. Landmark Isomap [115] solves both problems, since it only requires an $n \times N, n < N$ submatrix of the full distance matrix, and the eigenvalue problem is $O(n^3)$. Landmark Isomap employs landmark MDS (LMDS) instead of classical MDS. In LMDS the idea is to choose q points, called landmarks, where $q > r$ (where r is the rank of the distance matrix), but $q \ll n$, and to perform MDS on the landmarks, mapping them to R^d. The remaining points are then mapped to R^d using only their distances to the landmark points (so in LMDS, the only distances considered are those to the set of landmark points). As first pointed out in [7] and explained in more detail in [99], LMDS combines MDS with the Nystrom algorithm.

The same principle is used to project new points into the existing embedded space: the basic idea is to express the embedding coordinates of a point as a weighted sum of the coordinates of its nearest neighbors; see Section 3.3 fore more details.

Short-Circuiting Problem

In the presence of noise or when the data is sparsely sampled, short-circuit edges pose a threat to the Isomap algorithm [3]. Short-circuit edges occur when the folds in the manifolds come close, such that the distance between the folds of the manifolds is less than the distance from the neighbours.

Locally Linear Isomaps (LL-Isomaps) [112], a hybrid of Isomap and LLE, based on the local linear properties of the manifolds, was proposed to increase Isomap's robustness to short-circuiting. This algorithm has an additional parameter which one needs to set manually, and was illustrated with artificial data sets only.

Donoho and Grimes [37] pointed out the possibility of recovering non-convex manifolds by applying Isomap to a suitable decomposition of the manifold into overlapping geodesically convex pieces. However, a fully automatic procedure based on a general principle would be preferable in solving this problem.

2.1.4 Partial Data Representation: NMF

Among all dimensionality reduction methods, Non-negative Matrix Factorization (NMF) [80] is a promising tool for learning parts of the objects and images. While methods like Isomap seek to represent relationships between the different data points, NMF tries to represent relationships within the data features and therefore is useful in partial data representation tasks.

NMF factorises the data set into two matrix factors whose entries are all non-negative and produces a parts-based representation of the data because it allows only additive, not subtractive, combinations of basis components. For this reason, the non-negativity constraints are compatible with the intuitive notion of combining parts to form a whole. Because a parts-based representation can naturally deal with partial occlusion and some illumination problems, it has received much attention recently. NMF or its variations have been used in image classification [16, 50, 51, 53], face expression recognition [17], face detection [21], face and object recognition [85, 86, 105]. Donoho and Stodden describe in [36] the conditions under which NMF finds relevant data parts.

The original NMF algorithm produces global, not spatially localised, parts from the training set. To improve the NMF algorithm, Local NMF (LNMF) [83] was proposed for learning spatially localised, parts-based representations of visual patterns. It incorporates the following three constraints into the original NMF formulation.

- LNMF attempts to minimise the number of basis components required to represent the initial data. This implies that a basis component should not be further decomposed into more components.
- To minimise redundancy between different bases, LNMF attempts to make different bases as orthogonal as possible.
- Only bases containing the most important information should be retained. LNMF attempts to maximise the total "activity" on each component, i.e. the total squared projection coefficients summed over all training images.

LNMF has some drawbacks: it does not produce oriented filters from natural image data [62]. Further, there is no way to explicitly control the sparseness of the representation, should this be needed.

Hoyer in [61] extended the NMF framework to include an adjustable sparseness parameter. Later in [62] he presented an extension of those ideas, sparse Non-negative Matrix Factorisation (sNMF). The main improvement was that sparseness is adjusted explicitly, rather than implicitly. This allows it to discover parts-based representations that are qualitatively better than those given by basic NMF. Theis et al. in [120] proved that the employed projection step proposed by Hoyer in [62] has a unique solution, and that it indeed finds this solution.

Another NMF extension was presented by Wang and Jiar in [126]. The method,

Fisher Non-negative Matrix Factorization (FNMF), adds both the non-negative constraint and the Fisher constraint to matrix factorisation. The authors showed that FNMF achieves better performance than NMF and Local NMF. However one cannot in any way control the degree to which the representation is sparse.

Guillamet and Vitria in [52] introduced the use of the Earth Movers Distance (EMD) as a relevant metric that takes into account the positive definition of the NMF bases. They showed that NMF with EMD is able to deal with occlusions and it gives the best recognition results.

Recently it has been suggested that NMF can play a useful role in speech and audio processing [111, 116]. An auditory "scene", composed of overlapping acoustic sources, can be viewed as a complex object whose constituent parts are the individual sources.

2.2 Data Intrinsic Dimensionality

There is a consensus in the high-dimensional data analysis community that the only reason any methods work with very high-dimensional data is that, in fact, the data are not truly high-dimensional. Rather, they are embedded in a high-dimensional space, but can be efficiently summarised in a space of a much lower dimension, such as a nonlinear manifold. Then one can reduce the dimensionality without losing much information for many types of real-life high-dimensional data, such as images, and avoid the curse of dimensionality [5]. Learning these data manifolds can improve performance in classification and other applications, but if the data structure is complex and nonlinear, dimensionality reduction can be a hard problem.

The dimensionality of the embedding is a key parameter for manifold projection

methods: if the dimensionality is too small, important data features are collapsed onto the same dimension, and if the dimensionality is too large, the projections become noisy and, in some cases, unpredictable. There is no consensus, however, on how this dimensionality should be determined. LLE and its variants assume the manifold dimensionality is provided by the user. Polito and Perona proposed in their paper that this dimensionality should be known in advance [103]. They believe that the intrinsic dimensionality can be determined by using other dimensionality reduction methods. For example, Wang et al. [125] used Isomap to get the value of the intrinsic dimensionality for LLE. Isomap provides error curves that can be "eyeballed" to estimate the dimensionality. The charting algorithm, a recent LLE variant [14], uses a heuristic estimate of dimensionality which is essentially equivalent to the regression estimator of [98].

Also, other intrinsic dimensionality estimation methods have been proposed: estimating packing numbers of the manifold [73], or the k-NN method [29] where the dimensionality is estimated from the length of the minimal spanning tree on the geodesic NN (nearest neighbour) distances computed by Isomap.

2.3 Subspace Clustering for High-Dimensional Data

Clustering is one of the most useful methods in the data mining process for discovering groups and identifying interesting distributions and patterns in the underlying data. Thus, the main concern in the clustering process is to reveal the organisation of data patterns into sensible groups, which allow us to discover similarities and differences, as well as to derive useful inferences about them. Clustering looks at the properties of whole clusters instead of individual objects - a simplification that might be useful

when handling large amounts of data.

A lot of work has been done in the area of clustering. Jain et al. published a survey of clustering techniques [66]. More recent data mining texts include a chapter on clustering [47]. One of the more recent and comprehensive studies that dealt with the subject of subspace clustering was presented by Parsons et al. in [96]. Methods for evaluating and assessing the results of clustering algorithms were presented in [54, 55].

There are many different clustering algorithms, they can be grouped into two classes: hierarchical clustering and flat or non-hierarchical clustering. As a result of non-hierarchical clustering data points are divided into different groups, and the relations between groups are undetermined. These algorithms often have an iterative nature, they start with some initial state and at each step of iteration they improve the clusters until convergence [40]. Hierarchical clustering, unlike non-hierarchical, provides a data structure that represents the relationships between all points in the data set. Below we present a short review on both clustering classes. In our work we combine hierarchical and flat methods to keep their best properties.

2.3.1 Flat Clustering Algorithms

In general, flat clustering methods attempt to minimise a cost function or an optimality criterion which associates a cost to each instance-cluster assignment. The goal of this kind of algorithm is to solve an optimisation problem to satisfy the optimality criterion imposed by the model, which often means minimising the cost function. Flat clustering algorithms include:

1. K-means

   K-means clustering is an elementary but very popular approximate method that can be used to simplify and accelerate convergence. Its goal is to find K mean vectors $(v_1, ..., v_K)$ which will be the K cluster centroids. It is traditional to set number of clusters K manually, and to let K samples randomly chosen from the data set serve as initial cluster centers.

   In general, K-means does not achieve a global minimum over the assignments. In fact, since the algorithm uses discrete assignment rather than a set of continuous parameters, the minimum it reaches cannot even be properly called a local minimum. In addition, results can greatly depend on initialisation. Non-globular clusters which have a chain-like shape are not detected with this method. Despite these limitations, the algorithm is used fairly frequently as a result of its ease of implementation (see [40]).

2. Fuzzy K-means

   Fuzzy K-means partitions a set of data into K clusters so that the distances within the cluster are minimised [49].

   In every iteration of the classical K-means procedure, each data point is assumed to be in exactly one cluster. This condition can be relaxed by assuming each sample x_i has some graded or fuzzy membership in a cluster ω_k. These memberships correspond to the probabilities $p(\omega_k|x_i)$ that a given sample x_i belongs to class ω_k.

   The incorporation of probabilities as graded memberships sometimes improves the convergence of fuzzy K-means over its classical counterpart [40].

3. Gaussian Mixture Models and EM algorithm

   A Gaussian mixture model (GMM) is a kind of mixture density model, which assumes that each component of the probabilistic model is a Gaussian density [9]. The number of desired components has to be specified prior to the clustering procedure.

   Selection of an appropriate value for the number of GMM clusters k is important, since it affects both the level of noise in the constructed model (i.e. its smoothness), and the model accuracy. Generally, the larger the value of k, the more accurate the GMM result. However, a large number of GMM clusters also contributes towards noise in the model. We present our method for an automatic selection of this number in Section 4.3.

   Learning a Gaussian mixture model is in essence an unsupervised clustering task. The Expectation-Maximisation (EM) algorithm is used [34] to determine the maximum likelihood parameters of a mixture of k Gaussians in the feature space.

2.3.2 Hierarchical Clustering Algorithms

The advantage of hierarchical clustering is that it provides a structure for the whole data set. Hierarchical clustering builds a hierarchy of clusters which can be represented as a tree. The individual elements of the clustered data set form the leaves, while the root of the tree represents the whole data set.

Despite the fact that hierarchical clustering algorithms are computationally expensive, they are usually preferred over the flat algorithms. The bottleneck of the efficiency problem lays in the calculation of distances between all the pairs in the

data set. The advantage of the hierarchical algorithms is that they provide more information about the data set as a whole.

Hierarchical clustering algorithms can be classified into agglomerative and divisive approaches [40]. Both methods are based in the measures of the dissimilarities among the current cluster set in each iteration. Agglomerative algorithms merge some of the clusters, depending on how similar they are, and divisive algorithms split them. We focus on the agglomerative algorithms, but all these ideas apply to the divisive algorithms as well. The tree (called dendogram if the tree is binary) shows which clusters were agglomerated in each step. It can be easily broken at selected links to obtain clusters or groups of desired cardinality or radius. This number of clusters or groups can also be determined as a function of some merging threshold. The idea is that with a threshold of zero, the number of clusters is equal of the number of data points, and with a high threshold the data is partitioned in just one single cluster. Another tree advantage is that its structural representation is easy to generate and to store.

2.3.3 Clustering Problems

In the literature (see [6, 42, 43, 90, 123]), a wide variety of algorithms has been proposed for different applications and sizes of data sets, but still there are many unsolved problems in the general clustering theory. Most clustering benchmarks deal with low dimensional real world or synthetic data sets, not high dimensional data.

Often in high dimensional data, many dimensions are irrelevant and can mask existing clusters in noisy data. Feature selection removes irrelevant and redundant dimensions by analysing the entire data set. Subspace clustering algorithms localise the search for relevant dimensions allowing them to find clusters that exist in multiple,

possibly overlapping subspaces [96].

A frequent problem many clustering algorithms encounter is the choice of the number of clusters [88]. Quite different kinds of clusters may emerge when its value is changed. Most clustering methods require the user to specify a value, though some provide means to estimate the number of clusters inherent within the data.

Handling strongly overlapping clusters or noise in the data is often difficult. For instance, when clusters are not well separated and compact, the distance between clusters as the average distance between pairs of samples does not work well. Large data sets or vectors with many components can be a serious computational burden.

2.3.4 Clustering Evaluation

Clustering is a difficult task, because normally there is no a-priori information about the structure of the data or about the number of clusters. Hence, the accuracy of clustering entirely depends upon the data and the way the training algorithm is able to capture the structure in the data.

A fundamental way to evaluate clustering is to measure how "natural" the resulting clusters are. Here naturalness implies a fitting metric between the clusters and the data structure. If the resulting clusters make sense we expect that the assignment of the data samples to each cluster will not violate the structure inherent in the data. This idea can be captured mathematically using concepts from information theory. Entropy, first introduced by Shannon [30], measures how unexpected the information contained in a message is. The associated concept of relative entropy between probability density functions was defined by Kullback and Leibler [30]. We can think that clustering should divide the data set into disjoint subsets C_1 and C_2 where some interclass distance is maximised. This is named the splitting method for clustering

[57], and has been traditionally formulated as the Euclidean distance

$$d = \Sigma_j N_j \|m_j - m\|^2 \tag{2.3.1}$$

where m_j are the cluster centres (normally j=1,2), N_j is the number of samples in each cluster and m is the data mean.

In Gokcay and Principe's work [48], a new evaluation function based on a recently developed information theoretic measure defined from Renyi's entropy was proposed. The function was tested on five synthetic data sets with two classes of data and cluster distributions of varying geometric properties. Despite the simplicity of the test data, the minimum of the cluster evaluation function provided the correct clustering in all cases, which shows promising potential in this methodology. However, the proposed function is already computationally taxing for large data sets, and with the addition of dynamic adjustments of the kernel, it would not be a viable alternative for the evaluation of clusters from real images. Hence, for data sets of non-trivial sizes, a more robust and yet efficient method has to be found.

2.4 Dynamic Framework: HHMMs

This thesis focuses on use of Hierarchical Hidden Markov models (HHMMs) as a dynamic framework for hierarchical modelling of a high dimensional data. However there are several other interesting approaches that were considered during the development of this framework which are discussed in this section. By applying knowledge from the previous research, the HHMMs attempt to address the drawbacks of other models and to exploit the data structure in order to model this structure in a way that is more robust and offers deeper understanding of the data for different applications.

A dynamic framework for a data analysis may include various models such as Markov random fields (MRFs) [74], Bayesian networks (BNs) [97], Hidden Markov Models (HMMs) [104], Kalman Filter Models (KFMs) [67], Stochastic Context Free Grammars (SCFGs) [65] and other neural network architectures [60]. Generally, dynamic models represent variables as vertices in a graph structure and probabilistic interdependencies between variables as edges.

The use of HMMs in data modelling has become more and more popular, because HMMs have proven to be a powerful probabilistic framework for modelling stochastic processes. Starner et al. [117] applied HMMs to hand movement and sign language recognition. In their work, hands are segmented and tracked by using skin color feature. A high accuracy is achieved for the application of American Sign Language recognition with a lexicon of forty words. Wilson and Bobick also reported similar results on gesture recognition [129]. Applications of HMMs include biosequence analysis [41], speech recognition [28, 68], motion modelling [56, 71]. The main drawback of HMMs is that they do not take into account the structure of the underlying problem. The elements in the data sequence may be related by specific relationships and these relationships form the data semantic pattern. Thus the flat model like HMMs can not provide information about the data intrinsic relationships.

Dynamic Bayesian Networks (DBNs) generalise HMMs by allowing the state space to be represented in factored form, instead of as a single discrete random variable [94]. The network graph of a DBN represents data flow over time, where each variable within the model is assigned a time index as an extra parameter. This index parameter allows the probability distribution to be modified depending on the current time. Murphy in [94] describes different models that can be represented as DBNs, and how

to approximate inference and learn DBN models from data sequences. For example, Murphy considers HHMMs as a special kind of dynamic Bayesian network and derives a simpler inference algorithm. The complexity of such a DBN is linear in time, but exponential in relation to the depth of the HHMM.

Although much progress in learning the dynamic models has been made in many applications, the dynamic modelling field is still in an early stage. There are few dynamic models that are able to work with many different applications. They have drawbacks such as complexity, free parameters, accuracy, time complexity, etc. In this work, we construct a fully automated general dynamic framework using Hierarchical Hidden Markov Models (HHMMs) [44].

HHMMs generalise the standard HMMs by making each of the hidden states an "autonomous" probabilistic model of its own, that is, each state is an HHMM as well. An HHMM generates sequences by a recursive activation of one of the sub states of a state. This sub state might also be composed of sub states and would thus activate one of its sub states, etc. [44]. HHMM hidden state inference and parameter estimation can be efficiently learned using the expectation-maximisation (EM) algorithm.

Prior applications of HHMM falls into three categories:

- Supervised learning where manually segmented training data is available, hence each sub HMM is learned separately on the segmented sub-sequences, and cross-level transitions are learned on the transition statistics across the subsequences. For example, extron/intron recognition in DNA sequences [63], action recognition [65], and more examples summarised in [93] fall into this category.

- Unsupervised learning, where segmented data is not available for training, and the parameters of different levels are jointly learned; existing instances of these

HHMMs are [130, 131].

- A mixture of the above, where alignment at a higher level is given, yet parameters still need to be estimated across several levels. Examples abound: the application of building a speech recognition system with word level annotation [135], text parsing and handwriting recognition [44].

Our work represents a unique approach for dynamic modelling of a high dimensional data without a priori knowledge using HHMM with no supervision.

2.5 Summary

In the last decade, there was a large amount of research in high dimensional data modelling. Significant progress has been made in exploring of the underlying low dimensional data subspace by applying the dimensionality reduction techniques. Non linear dimensionality reduction techniques became very popular for revealing a low dimensional data manifold, as they represent real world data more accurately. As we noted in Section 2.1, each of them has its own advantages and disadvantages. In our work we use Isomap to reduce the data dimensionality because of its property of preserving the data's geodesic distances. We use this property for further data analysis. The quality of the Isomap low dimensional data projection depends on its parameter specifying neighbourhood size. We implemented a fully automated method to set this parameter, which we present in the next chapter.

Another related area in high dimensional data modelling is hierarchical clustering: the real world data description usually is not "flat". Many clustering algorithms for different applications have been proposed; however, the problem of effective, robust

and automatic hierarchical clustering of a high dimensional data with no a priori knowledge remains, in general, unsolved.

Recently, there has been great interest growing in the dynamic data modelling field. Although many techniques have been proposed for many applications, the field is still in an early stage. In this thesis we introduce a novel approach for dynamic modelling of high dimensional data without a priori knowledge using HHMM with no supervision. For the model construction, we incorporate our novel fully automated hierarchical clustering algorithm described in Chapter 4.

Chapter 3

Nonlinear Dimensionality Reduction Using Isomap

3.1 Introduction

Usually, data from the real world is of a high dimensional nature, and so is very difficult to understand and analyse. Dimensionality reduction is an important and necessary preprocessing of such data. Non-linear dimensionality reduction techniques represent low dimensional nonlinear manifolds better than the linear ones, as we surveyed in the previous Chapter. All nonlinear algorithms share the same basic approach, consisting of three steps:

1. Computing neighbourhoods in the input space;
2. Constructing a square matrix with as many rows as elements in the input data set;
3. Calculating spectral embedding using the eigenvectors of this matrix.

One usually has to manually specify the number of neighbours used in the first step. There are a number of problems associated with the choice of the neighbourhood

parameter. A large number of nearest neighbours causes smoothing or elimination of small-scale structures in the manifold. In contrast, too small a neighbourhood can falsely divide the continuous manifold into disjoint sub-manifolds.

Kouropteva et al. [75] presented an automatic method used to detect the optimal value of the neighbourhood parameter for LLE. We have extended this idea to choose the optimal number of neighbours K and the neighbour region ϵ for Isomap.

Among all nonlinear dimensionality reduction techniques, we choose Isomap [118] to represent the data in a low dimensional subspace, as discussed in Section 2.1. As a geodesic method, Isomap reveals the underlying data structure and preserves the data geometry.

We introduce the Isomap algorithm in Section 3.2, also we present a function to embed a new point into the Isomap space in Section 3.3, and in Section 3.4 we give an automated Isomap inverse projection algorithm.

In Section 3.5 we introduce the new method for choosing the neighbourhood size for Isomap and demonstrate how it works, providing examples in Section 3.6. We also compare the mapping produced by Isomap with the optimal parameter chosen by our algorithm with mappings produced by PCA, LLE and Isomap with different parameter values for the same data set. We use the following approaches to compare the results:

- Using various measures, such as residual variance, Spearman ρ, Pearson's correlation coefficient, see [45, 95]. For example, Navarro and Lee [95] applied Isomap and MDS to the graphical depiction of a document set. They employed variance and spatial visualisation to compare these techniques. Friedrich [45] compared Isomap, LLE and PCA methods on different data sets, synthetic and

real, using correlation coefficient and visualisation.

- Using classification (i.e. comparing how well these methods recognise images from the test data set). Since we cannot add new samples into LLE and Isomap spaces in a straightforward manner, we use their extensions to classify images. Yang [132] has extended Isomap to improve classification with a Fisher Linear Discriminant (FLD) phase. Ridder et al. [106, 107] presented Supervised LLE (SLLE). Kouropteva et al. [76] applied SLLE combined with support vector machines (SVM) for classifying handwritten digits.

The purpose of these comparisons is to demonstrate that Isomap with the automatically selected optimal parameter (K or ϵ) performs better than PCA and LLE with various data sets, see Section 3.6.

3.2 Isomap Algorithm

The Isomap technique is based on the idea of creating a high dimension to low dimension transformation as a graph problem. The Isomap algorithm extends the classical techniques of principal component analysis (PCA) and multidimensional scaling (MDS) to a class of nonlinear manifolds.

On input, the Isomap algorithm requires the distances $d_X(i,j)$ between all pairs i,j from N data points in the high-dimensional input space X, measured using either the standard Euclidean metric, or some application-specific metric.

The algorithm outputs coordinate vectors Y_i in a (lower) d-dimensional Euclidean space Y that best represents the intrinsic geometry of the data.

The complete isometric feature mapping, or Isomap, algorithm has three steps:

1. The estimation of the neighbourhood graph.

   The first step determines which points are neighbours on the manifold M, based on the distances $d_X(i,j)$ between pairs of input points i,j in the input space X. Given the input points, we determine the set of neighbours for each point either by K nearest neighbours (K-Isomap) or all those within some fixed radius ϵ (ϵ-Isomap). These neighbourhood relations are represented as a weighted graph G over the data points, with edges of weight $d_X(i,j)$ between neighbouring points. Note, that in the case of K-Isomap, vertices in the graph may have degree greater than K since the K nearest neighbourhood relationship need not to be symmetric.

2. Computing the shortest path graph given the neighbourhood graph.

   In its second step, Isomap estimates the geodesic distances $d_M(i,j)$ between all pairs of points on the manifold by computing the shortest path lengths $d_G(i,j)$ in the graph G. First we set $d_G(i,j) = d_X(i,j)$ if i,j are linked by an edge, and $d_G(i,j) = \infty$ otherwise. Then for each value of $k = 1,2,...,N$, we replace all entries $d_G(i,j)$ by $\min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$. The matrix of final values $D_G = \{d_G(i,j)\}$ will contain lengths of the shortest paths between all pairs of points in G.

3. Construction of lower dimensional embedding.

   The final step applies classical MDS to the matrix of graph distances $D_G = \{d_G(i,j)\}$, constructing an embedding of the data in a d-dimensional Euclidean space that best preserves the manifold's estimated intrinsic geometry. The coordinate vectors Y_i for points in Y are chosen to minimise the cost function

$$E = \|\tau(D_G) - \tau(D_Y)\|_{L^2} \tag{3.2.1}$$

where D_Y denotes the matrix of Euclidean distances $\{d_y(i,j) = ||y_i - y_j||\}$ and $||A||_{L^2}$ is the matrix norm $\sqrt{\sum_{i,j} A^2{}_{ij}}$. The τ operator converts distances to inner products, which uniquely characterise the geometry of the data in a form that supports efficient optimisation [118].

The only free parameter, K or ϵ, which is the neighbourhood factor, appears in the first step. Tenenbaum et al. [119] pointed out that the success of Isomap depends on being able to choose an appropriate neighbourhood size, that is neither so large that it introduces short-circuit edges into the neighbourhood graph, nor so small that the graph becomes too sparse to approximate geodesic paths accurately. However, in the original Isomap paper [118] the parameter value was chosen manually.

3.3 Kernel Trick: New Data Sampling into Isomap Space

We project the new data into Isomap space by using the "kernel trick" in the same manner as explained in [22]. Having the new data $X^{new} \in \mathbb{R}^D$ with the elements $\{X_l^{new}\}_{l=1}^L$, we can compute the corresponding Isomap coordinates $Y^{new} \in \mathbb{R}^d$ from the equation

$$[Y_l^{new}]_i = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{N} [\nu_i]_j k(X_l^{new}, X_j) \tag{3.3.1}$$

where $[\cdot]_i$ represents the ith element of a vector, $i = 1..d$, λ_i is ith Isomap eigenvalue, ν_i is ith Isomap eigevector, X is the original data with N points, and k is the kernel which is described by

$$k(X_l^{new}, X_j) = -\frac{1}{\sqrt{2}}(\tilde{D}_{lj}^2 - \frac{1}{N}\sum_{i=1}^{N}\tilde{D}_{li}^2) \tag{3.3.2}$$

$\tilde{D}_{lj}$ denotes geodesic distance between points X_l^{new} and X_j. We show the example of new data projection for the randomly generated swissroll data in Figure 3.1. Here the Isomap parameter value is $\epsilon = 3.4$ (selected by our method).

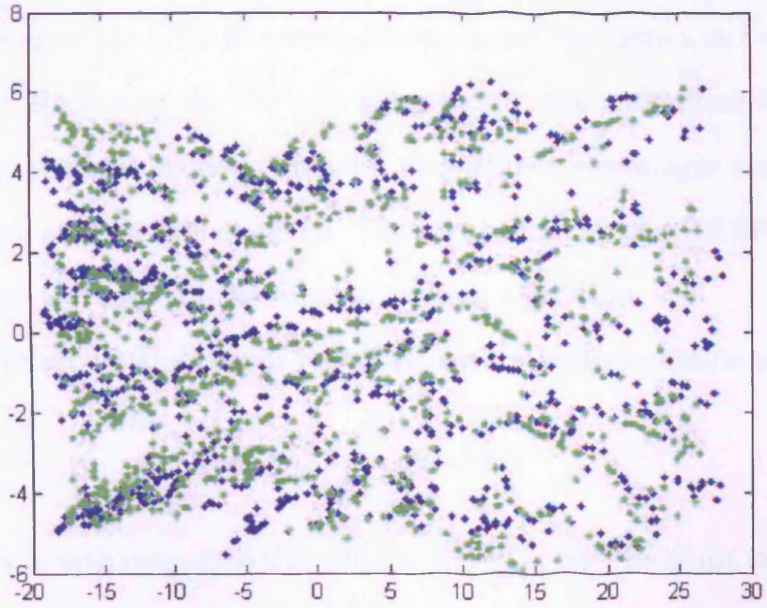

Figure 3.1: A new data projection into Isomap space (1000 points, green) for the swissroll data, 1000 points (blue), 4% noise.

The same idea is shared in Landmark Isomap [115]. This method was created to handle large data sets and involves landmark points and an out-of-sample extension of Isomap. The algorithm is similar except for the kernel from Equation (3.3.2), see [115] for details. In our work we use Landmark Isomap for large data and set the Landmark Isomap parameter using our method from Section 3.5.

3.4 Isomap Inverse Projection

In Section 3.2 we built a mapping $f : X \rightarrow Y$ from the original high dimensional data $X \subseteq \mathbb{R}^D$ to its low dimensional representation $Y \subseteq \mathbb{R}^d$, $d \ll D$. Here we construct its back projection, $f^{-1} : Y \rightarrow X$. As Isomap is a non-linear method and f is implicit function, we can not construct a straightforward inverse mapping, therefore we use a Generalised Radial Basis Functions (GRBFs) interpolation network [?].

At the first step of the GRBFs construction, we set the distances between Isomap manifold points. Reflecting the intrinsic geometry of the underlying data manifold, GRBFs networks with the geodesic distance metric have advantages over GRBFs networks with the usual Euclidean distance. Thus at the first step of GRBFs construction we apply the first two steps of the original Isomap algorithm.

The GRBF interpolation models functions with a linear equation of the form

$$f(Y) = \sum_{k=1}^{D} \omega_k h_k(Y) \tag{3.4.1}$$

The linearity is with respect to the weights $\{\omega_k\}_{k=1}^{D}$, not the input variable Y. The basis functions $\{h_k\}_{k=1}^{D}$ are the transfer functions of the hidden units in the network.

A radial basis function network is a representation of Equation (3.4.1) as a network with the input, the hidden layer and the output nodes. Each hidden node is represented by a single GRBF, with an associated center position and weight. Each output node is defined by a weighted summation of the hidden nodes, using the ω_k as weights. We use Gaussian functions [124] for the network construction, but with the geodesic distance metric.

Let $c_i \in \mathbb{R}^d$, $i = 1..m$ be a set of m cluster centers in the embedding space obtained by using a K-Means clustering algorithm. These centers represent embedded

manifold structure. We experimentally choose m such that $m = \frac{2}{3}N$ to get an accurate data representation. With this number our clusters are just large enough to hold all significant data variations, and small enough to reach efficient computation times. Selecting less cluster centers resulted in a significant loss of data variations. Let $r_i \in \mathbb{R}^d$ be a set corresponding to the c_i radii. We set r_i to be equal to the maximal geodesic distance within the cluster.

Given a network consisting of m GRBFs with centres c_i and radii r_i and a training set with N patterns $\{(x_j, y_j)\}_{j=1}^{N}$ the optimal network weights can be found by minimising the sum of squared errors

$$E = \sum_{j=1}^{N} (f(y_j) - x_j)^2 \tag{3.4.2}$$

This leads to a set of N linear equations in N_c unknown weights and can be solved with the so-called normal equation

$$W = (H^\top H)^{-1} H^\top X \tag{3.4.3}$$

where H is the design matrix, its elements are $H_{ji} = h_i(y_j)$ and $X = [x_1 ... x_N]$ is the N dimensional vector of training set output values.

The resulting projection can be written as

$$A = H * W \tag{3.4.4}$$

The outline of our algorithm is shown in Table 3.1.

After training the GRBFs network, we can use the learned projection to calculate the mapping of the new points from the Isomap space into the original high dimensional space.

1. Calculate geodesic distances and shortest paths on Y using the first two steps of the Isomap algorithm.
2. Set GRBFs parameters m, c, r.
3. Calculate GRBFs H.
4. Using Equation 3.4.3, calculate W.
5. Calculate the resulting GRBF projection from Equation 3.4.4.

Table 3.1: GRBFs construction algorithm

3.5 Optimal Neighbourhood Parameter Value for the Isomap Algorithm

How does one correctly choose the optimal parameter value for the Isomap algorithm? In fact, neighbourhood selection can be quite critical. If the neighbourhood is too large, the local neighbourhoods will include the data points from other branches of the manifold, shortcutting them, and leading to substantial errors in the final embedding. If it is too small, it will lead to discontinuities, causing the manifold to fragment into a large number of disconnected clusters.

Here, we present a method for determining the optimal neighbourhood size, inspired by the approach of Kouropteva et al. [75] for LLE parameter selection. By "optimal" here we mean that with this parameter Isomap will solve the dimensionality reduction problem most accurately (finding meaningful low-dimensional structures of data hidden in their high-dimensional observations).

The scale-invariant K parameter is typically easier to set than the neighbourhood radius ϵ, but Tenenbaum et. al. [118] noted that when the local dimensionality varies across the data set, the approach which uses K nearest neighbours, may yield

misleading results.

We note that most of the algorithms for the automatic selection of optimal values of K and ϵ are similar, excluding the first step. We describe our algorithm relative to K, making notes about ϵ if necessary. Automatic selection of the optimal parameter value algorithm consists of four steps, which are detailed in Table 3.2.

1. Choose the interval of possible values of K, $K_{opt} \in [K_{min}, K_{max}]$.
   - K_{min} is the minimal value of K, with which the neighbourhood graph (from the second step of Isomap) is connected.
   - Maximal values:
     - choose K_{max} according to the equation
     $$\frac{2 * P}{N} \leq K + 2 \,, \tag{3.5.1}$$
     where P is the number of edges and N is the number of nodes in the neighbourhood graph.
     - choose ϵ_{max} as $max(d_X(i,j))$, where $d_X(i,j)$ is the Isomap input matrix.
2. Calculate the cost function $E(K)$ (see Equation (4.3.1)) for each $K \in [K_{min}, K_{max}]$.
3. Compute all minima of $E(K)$ and corresponding K which compose the set S_K of initial candidates for the optimal value.
4. For each $K \in S_K$ we run Isomap and determine K_{opt} using the formulae
   $$K_{opt} = arg \min_K (1 - \rho^2_{D_x D_y}) \,, \tag{3.5.2}$$
   where D_x and D_y are the matrices of Euclidean distances between pairs of points in input an output spaces, respectively, and ρ is the standard linear correlation coefficient, taken over all entries of D_x and D_y.

Table 3.2: Algorithm for automatic selection of the optimal parameter value

During the first step, we choose the interval containing the possible optimal values

of K or ϵ. The minimal values of K and ϵ are chosen using the same approach. We take the minimum possible K or ϵ such that the corresponding graph (produced by the second step of Isomap) is connected (with small parameter values this graph is disconnected).

The procedures we use to choose the upper bound are different for K and ϵ. We take the upper bound for ϵ to be the diameter of our input data set (we define this diameter as the maximum distance between any two points in the input space). In the case of K, after completing numerous experiments and analysing the resulting data, we noticed that in all the considered cases, if the average node degree of the graph (computed with some value of K) is greater than $K + 2$, one usually gets shortcuts which do not follow the surface of the manifold. Therefore we choose the upper bound of the optimal value of K to be the maximum value of K for which (3.5.1) holds. Equation 3.5.1 was obtained heuristically, taking into account Euler's formula for a connected planar graph [35], to avoid extra connectivity and to reduce computational time. Our experiments demonstrate that this K_{max} choice leads to good results.

At the second step we calculate the values for the cost function E (see (4.3.1)) for parameters from the chosen interval. E is a function of the dimension of the embedding space, and we assume that the embedding dimension is known a priori. This dimension can be estimated using one of the numerous methods for intrinsic dimension estimation [19, 29, 73]. Note that in practice in all our experiments the best low dimensional embedding always has the same dimension as the embedding with K_{min}.

Equation (3.5.2) was used in [75] to measure how well the high dimensional structure is represented in the embedded space. Tenenbaum used the residual variance in

the original Isomap paper to determine the dimension of the embedding space ("elbow" technique). The idea of the minima search in this equation is very similar to the Tenenbaum's idea about the "elbow".

3.6 Experimental Results

In this section we present the results of experiments with seven different data sets. Here we evaluate how the proposed algorithm from the previous Section processes different data types, artificial and real. The first considered data set in Section 3.6.1 consists of artificial images of a face rendered with different poses and lighting directions which can be described by a limited number of features. Sections 3.6.2 and 3.6.3 utilise two synthetic two-dimensional manifolds – Swissroll and S-curve, lying in a three-dimensional space. In these data sets the underlying structure is known and thus the quality of the embeddings, obtained by Isomap with different parameters and by PCA and LLE, can be measured by Pearson's correlation coefficient and Spearman's ρ of their resulting lower-dimensional coordinates with the actual coordinates. The next Section further demonstrates our algorithm's ability to work with real world data set, which is the ear data. Section 3.6.5 utilises the Olivetti face database in which different faces vary slightly in pose as well as scale. The last Section 3.6.6 uses a set of handwritten digits from the MNIST database. The Olivetti and MNIST databases were used for the classification experiments.

3.6.1 Sculpture Face Data Set

First, we tried to select the parameter for the sculpture face data set automatically. This data set was used by Tenenbaum et. al in the original Isomap paper [118]. The

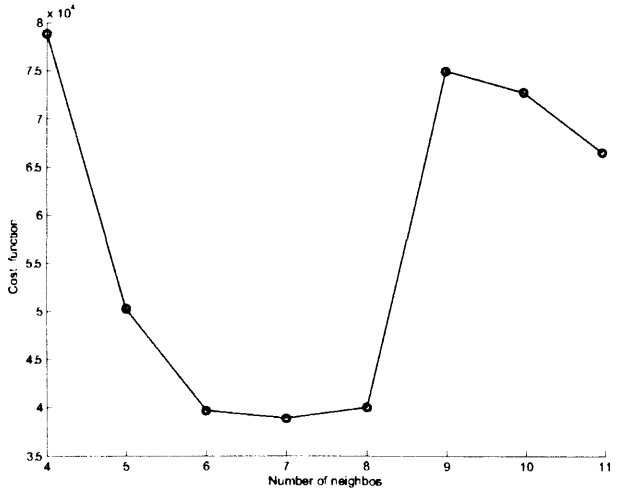

Figure 3.2: Cost function for sculpture data set

sculpture face data set consists of 698 gray-scale images, 64 × 64 pixels each. We represent each image by a raster scan vector and form the Isomap input space X.

Following the first step of our algorithm, we choose the interval for K_{opt}, which according to our algorithm is [4, 11]. Then we reduce the number of possible optimal parameter values. In Figure 3.2, one can see the plot of the cost function $E(K)$. The optimal value of K determined by our method for this data is 7.

Tenenbaum et. al in [118] used $K = 6$. Indeed, the cost function values for $K = 6$ and $K = 7$ are sufficiently close, as are the residual variance values at these points. On the other hand, if we calculate the Spearman ρ between the input and output data with $K = 6$ and $K = 7$, then we get $\rho_{K=6} = 0.557$ and $\rho_{K=7} = 0.5645$. This means that the magnitude of the correlation between the input and output data is greater with $K = 7$.

Having applied Isomap to the sculpture face set, we obtained a three-dimensional embedding of initial data. Each dimension of embedding represents one degree of freedom of the underlying original data: left-right pose, up-down pose, and lighting

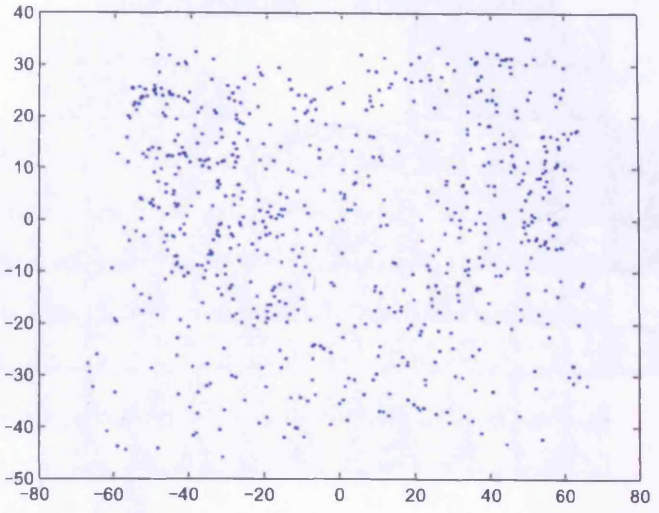

Figure 3.3: Two-dimensional Isomap embedding of sculpture data set with optimal parameter $K = 7$

Figure 3.4: Two-dimensional Isomap representation of sculpture data set with optimal parameter $K = 7$

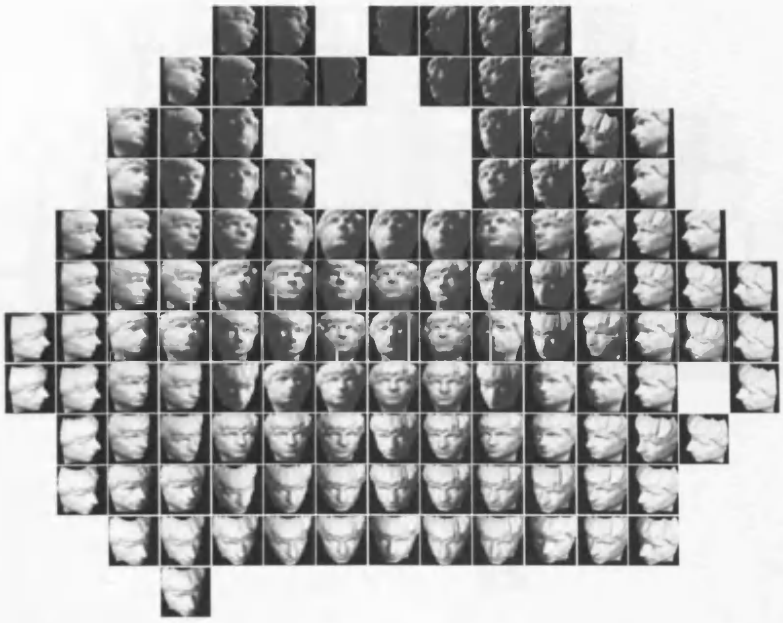

Figure 3.5: Two-dimensional Isomap representation of sculpture data set with suboptimal parameter $K = 15$

direction. Figure 3.3 demonstrates the two-dimensional projection of the sculpture data set embedding, obtained by running Isomap with the optimal parameter value of $K = 7$. To illustrate this embedding, we draw the original sculpture face images instead the points (with the space scaling to get more compact representation), as shown in Figure 3.4. Same as Tenenbaum in the original Isomap paper, we got the left-right pose changes on the horizontal axis, and up-down pose changes on the vertical axis. The third dimension, which is lighting direction, is less significant than the first two.

We include two more pictures of the Isomap embedding with suboptimal parameters for this data set in Figure 3.5 and Figure 3.6 to illustrate the problem of the parameter optimality. Clearly, it is difficult to understand main features of the data hidden in these mappings.

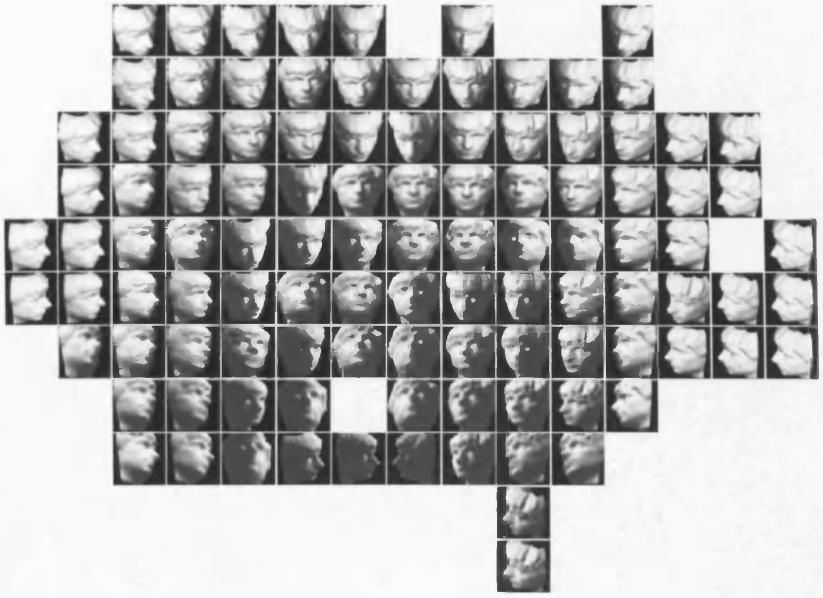

Figure 3.6: Two-dimensional Isomap representation of sculpture data set with sub-optimal parameter $\epsilon = 1.2$

3.6.2 Swissroll

Next, we proceed applying different methods to the Swissroll data set, which is a synthetic example of a non-linear manifold. The Swissroll was used by Tenenbaum et. al in the original Isomap paper [118] to illustrate how the method works. Figure 3.7 shows its shape as well as 2000 points sampled from the manifold.

We applied our algorithm to numerous Swissroll data sets, each containing from 200 to 2500 randomly chosen points. For each set we ran 50 to 100 experiments with different values of Isomap parameter ϵ. The cost function plot here is similar to those for Swissroll data sets with different numbers of points (see Figure 3.8), with minima in the same point $\epsilon = 6.2$. This ϵ value was chosen as optimal for all the Swissroll data sets. With the next neighbour region size, $\epsilon = 6.3$, the neighbourhood graph becomes over connected, which leads to rapid growth of the cost function.

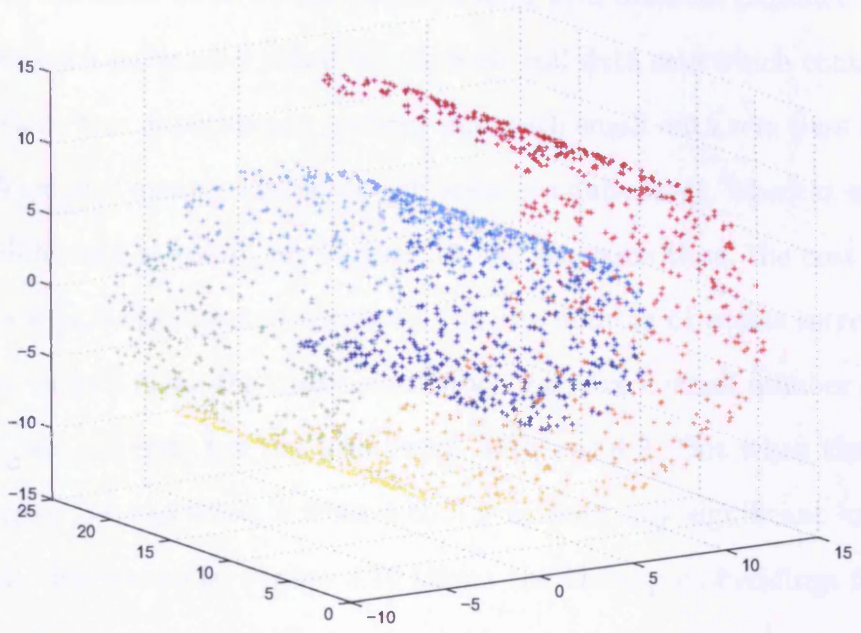

Figure 3.7: Swissroll data set

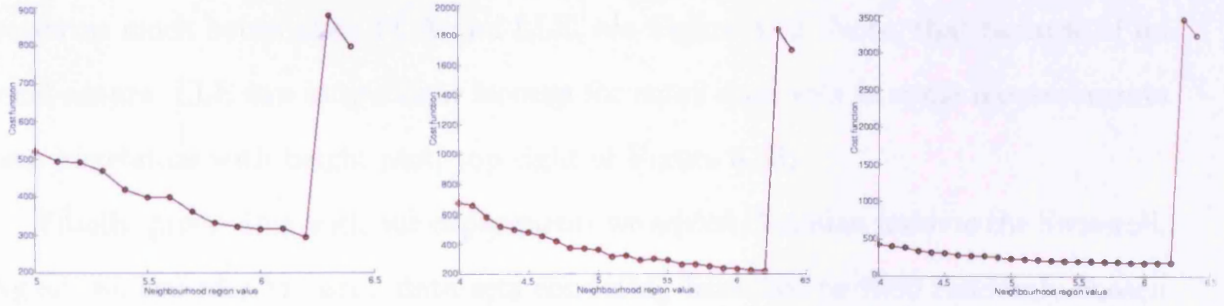

Figure 3.8: Swissroll cost functions at 200, 1000 and 2000 points

To demonstrate the optimality of the chosen value of ϵ we calculate the following measures: residual variance, correlation coefficients and Spearman's ρ with angle and height. First, we calculate these measures for Isomap with different parameter values, $\epsilon = 4, 5, 6.2$. For each parameter value we use Swissroll data sets which contain from 200 to 2500 points. Our experiments confirm that with small data sets (less than 700 points) the difference of measure values for different ϵ is quite large, whereas with large data sets this difference is small, see Figure 3.9. At the same time, the cost function in the interval $\epsilon \in [4, 6.2]$ tends to be smoother as the number of points increases (see Figure 3.8). So, we can make the conclusion: when we have a small number of points (less than 700), we will only get the best result with $\epsilon = 6.2$. But when the number of points increases, we can select ϵ from 4 to 6.2 without any significant loss in the quality of result. For example, Figure 3.10 shows the Isomap embeddings for $\epsilon = 4$, $\epsilon = 5$, $\epsilon = 6.2$ – they look quite similar and good in contrast to the mapping obtained with $\epsilon = 6.3$ (Figure 3.11).

Next, we compare Isomap (with selected optimal parameter) with other techniques: PCA and LLE. Analogously, we measure residual variance, correlation coefficients and Spearman's ρ with angle and height. Our results show that Isomap performs much better than PCA and LLE, see Figure 3.12. Note, that because of its local nature, LLE can outperform Isomap for small data sets in some measurements (see correlation with height plot, top right of Figure 3.12).

Finally, proceeding with our experiments we added Gaussian noise to the Swissroll. Again, we consider Swissroll data sets consisting from 200 to 2500 randomly chosen points, and run Isomap 50 to 100 times. However, finding the optimal parameter for data with noise is a difficult task – at a given noise level our experiments failed to

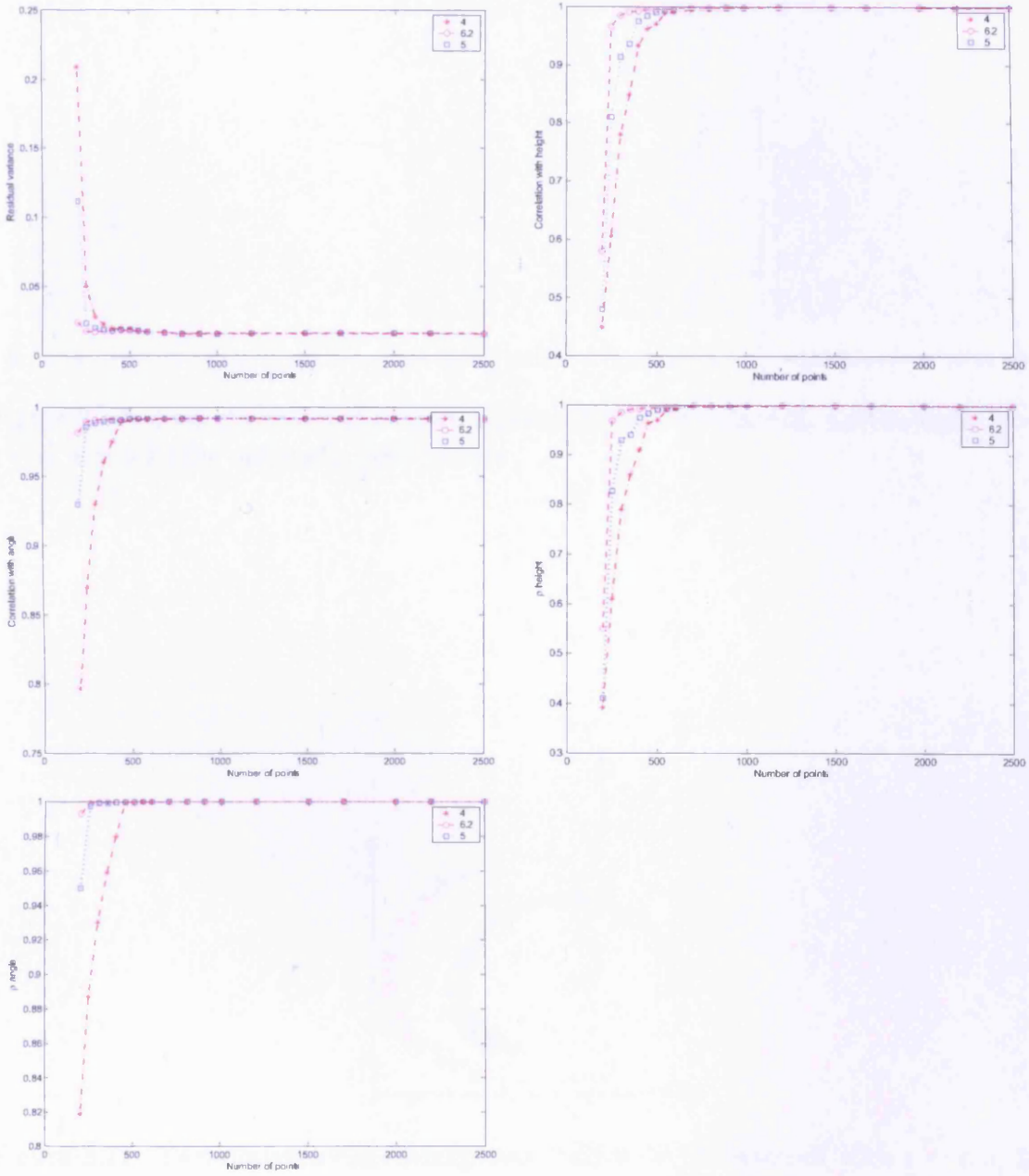

Figure 3.9: Residual variance, correlation coefficients and Spearman's ρ with height and angle (left to right) for the Swissroll data. At each plot these values are shown for $\epsilon = 4, 5, 6.2$

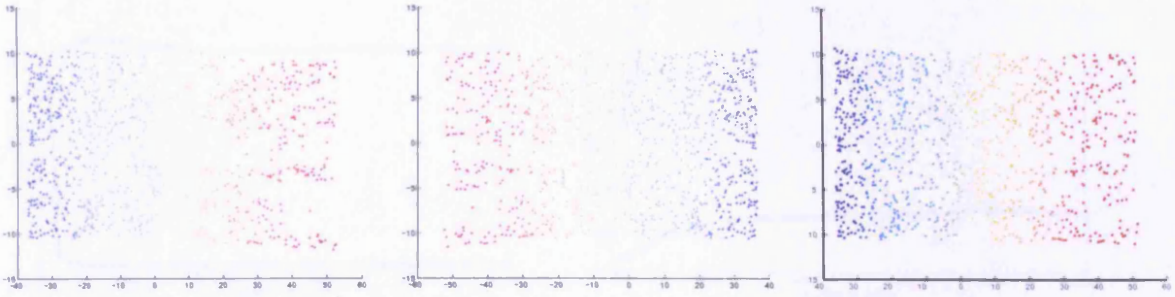

Figure 3.10: Two-dimensional Isomap representation of Swissroll. Left to right: $\epsilon = 4$, $\epsilon = 5$, $\epsilon = 6.2$ (the optimal), 1000 points

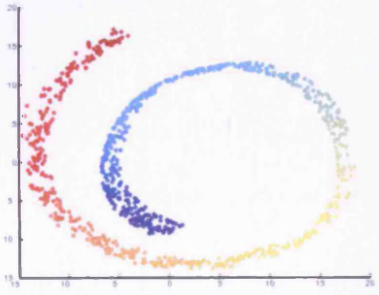

Figure 3.11: Two-dimensional Isomap representation of Swissroll with $\epsilon = 6.3$, 1000 points

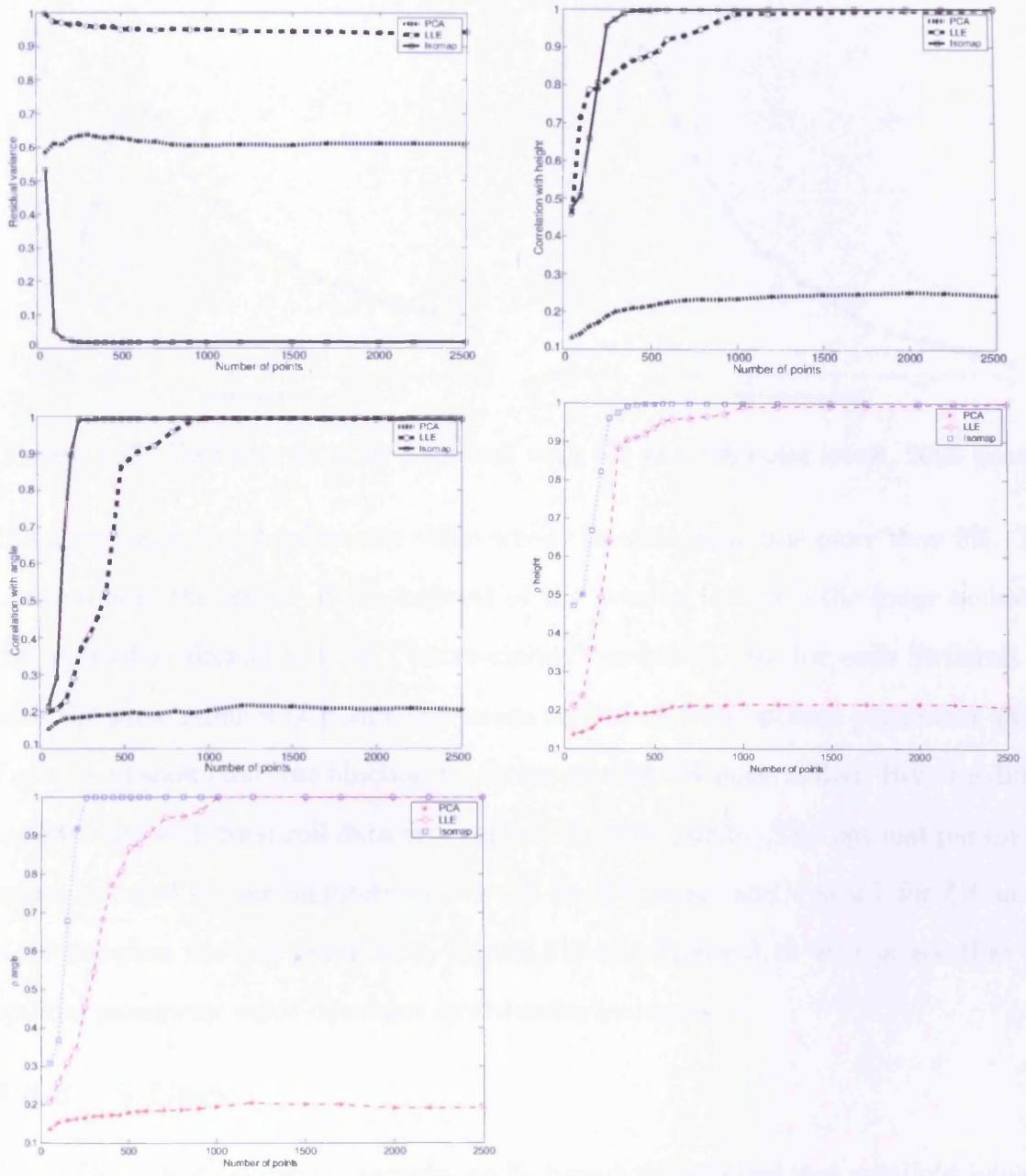

Figure 3.12: Residual variance, correlation coefficients and Spearman's ρ with height and angle (left to right) with PCA, LLE ($K = 8$) and Isomap ($\epsilon = 6.2$) for the Swissroll data

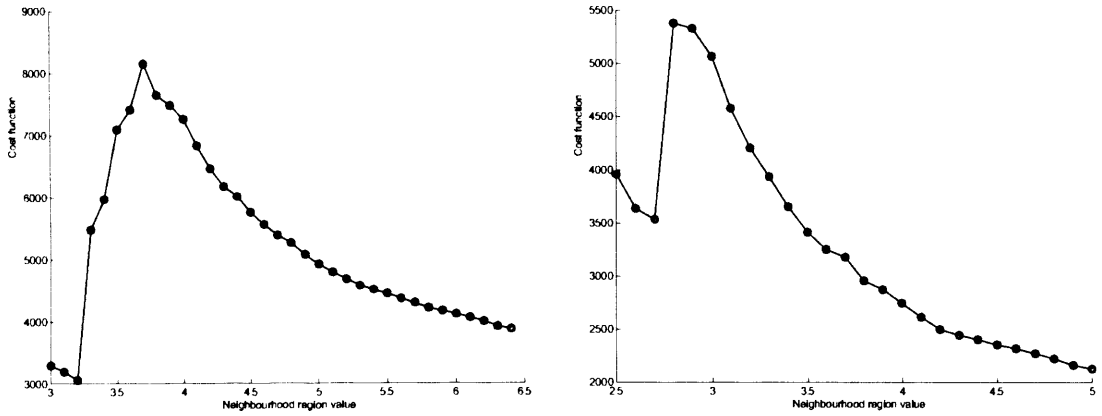

Figure 3.13: Cost functions for Swissroll with 4% and 5% noise levels, 2000 points

find a general optimal parameter value when the noise ratio was more than 5%. The problem is in the choice of the interval of the possible values – the lower bound of the interval is already too big ("short-circuit" problem). So, for each Swissroll set with the same number of points one needs to find its own optimal parameter value. Figure 3.13 shows the cost function for Swissroll with 4% noise added. Here we did 70 experiments with Swissroll data sets containing 2000 points. The optimal parameter values selected by our method are $\epsilon = 3.3$ for 4% noise, and $\epsilon = 2.7$ for 5% noise. If we compare the last graph from Figure 3.8 and Figure 3.13 we can see that the optimal parameter value decreases as the noise increases.

3.6.3 S-Curve

The S-Curve is a synthetic example, an S-shaped two-dimensional manifold lying in a three-dimensional space, see Figure 3.14. Analogously to the Swissroll example, we run our algorithm 50 to 100 times with different S-Curve data sets, containing from 50 to 2500 randomly chosen points.

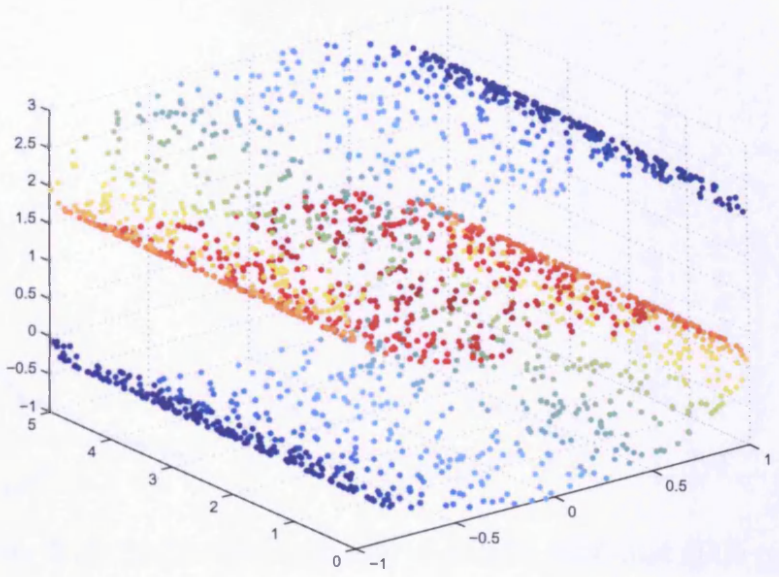

Figure 3.14: S-Curve data set

S-Curve cost function plots are all similar (see Figure 3.15) for the data sets with different number of points, but the location of the minima decreases as the number of points increases. As seen in Figure 3.15, $\epsilon_{min} = 1$ when number of points is equal to 200, $\epsilon_{min} = 0.8$ with 1000 points, and $\epsilon_{min} = 0.4$ with 2000 points.

To compare Isomap with other techniques (PCA and LLE) we used the same measures as in the case of Swissroll (see Figure 3.16). It turned out that the optimal parameter value for Isomap depends on the number of points sampled from the S-Curve. Again, on average, Isomap results are better than both PCA and LLE results.

Figure 3.17 shows the Isomap mapping of the S-Curve, obtained with the optimal parameter. Here the horizontal axis corresponds to angle, and the vertical axis corresponds to height.

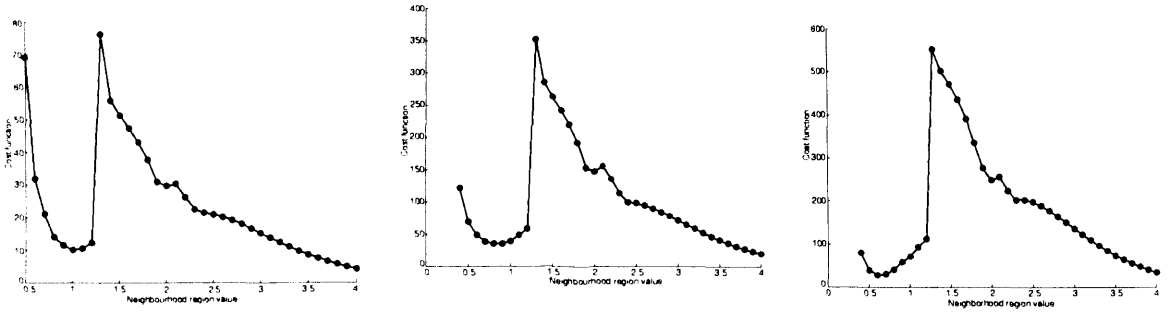

Figure 3.15: S-Curve cost function at 200, 1000 and 2000 points

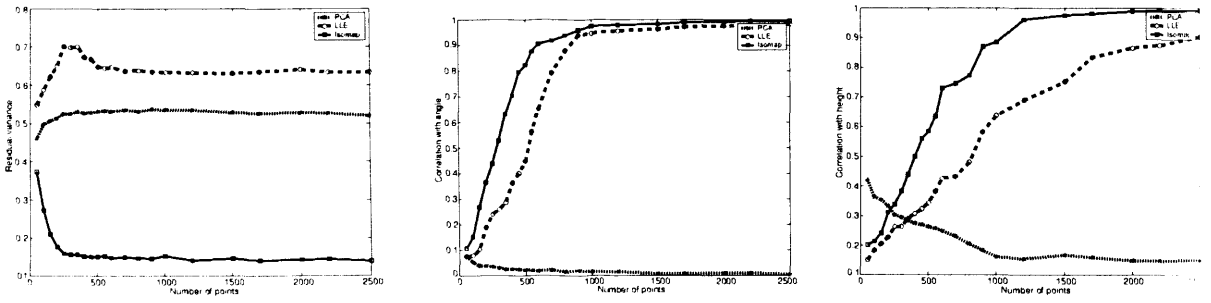

Figure 3.16: Residual variance, correlation coefficients with angle and height (left to right) with PCA, LLE ($K = 18$) and Isomap for the S-Curve data

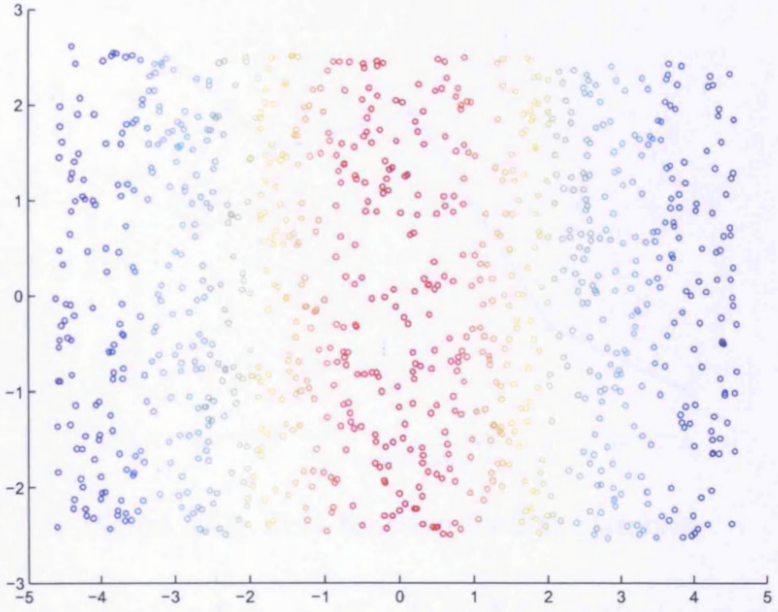

Figure 3.17: Two-dimensional Isomap representation of S-Curve with $\epsilon = 0.8$, 1000 points

3.6.4 Ear Database

The database was created by David Hurley (Southampton University) and consists of 70 grey-scaled ear images of size 201×281, cropped from 5 samples for each of 14 subjects, see [64] for more details. First, we transform each image into a column vector ($N = 56481$). Then we concatenate all vectors to form the input matrix of size 70×56481. Figure 3.18 shows the cost function $E(K)$ plot with the chosen interval $K \in [6, 10]$. There are two minima of this function: with $K = 7$ and $K = 10$, and we choose the K value with minimal residual variance, which is 7.

Figure 3.19 shows the 2-dimensional projection of discovered Isomap embedding, with neighbourhood value $K = 7$, where the axes represent the ear shape. The shape of an ear tends to be dominated by the helix, and also by the shape of the lobe [18]. Therefore these are the most important features for ear biometrics. With our

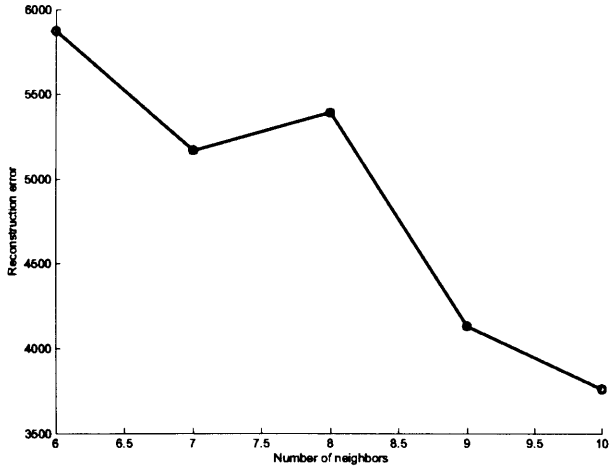

Figure 3.18: Cost function for Ear Database

embedding, we got variations from small to large helix at the horizontal axis, and the ear angular position (from strictly vertical at the top of the figure, to diagonal at the bottom). The result shows the potential of using Isomap instead of the widespread use of PCA in ear biometrics.

3.6.5 Classification Experiments: Olivetti Face Database

The Olivetti face database is a widely used set of images, consisting of 400 images of 40 subjects, which can be found at http://www.uk.research.att.com/ facedatabase.html. For some of the subjects, the images were taken at different times. There are variations in facial expression (open/closed eyes, smiling/non-smiling), and facial details (glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an up-right, frontal position, with tolerance for some sideways movements. The images are greyscale with a resolution of 92×112.

We divide this set into two subsets, a training set (200 images) and a test set (200 images). The basic Isomap method is not optimal from the classification viewpoint, so

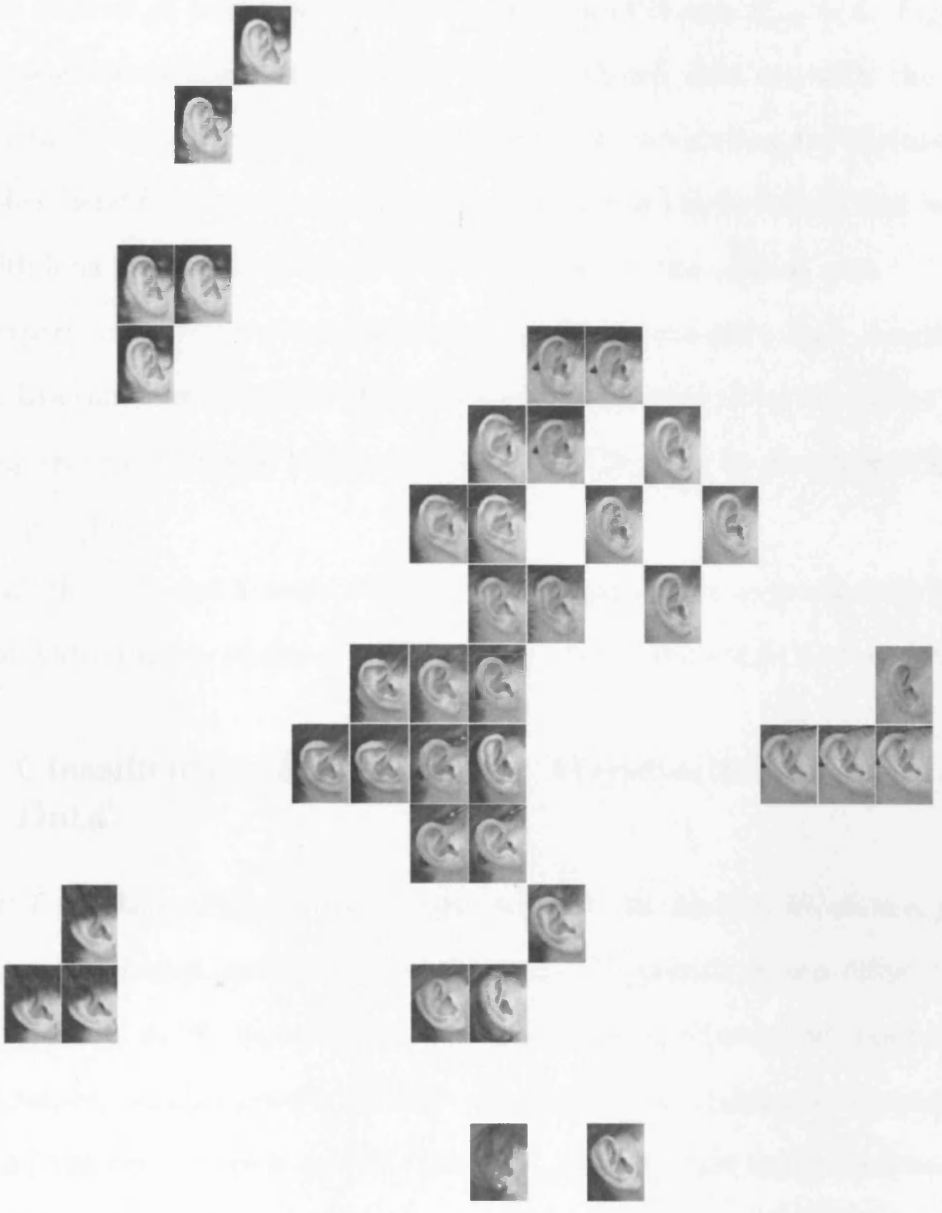

Figure 3.19: Two-dimensional Isomap embedding ($K_{opt} = 7$) for Ear Database

we will use its extended version, Discriminant Isometric Mapping [132, 134]. Applying our algorithm to find optimal parameter for Discriminant Isometric Mapping, we obtain the interval of parameter K values: $[6, 7]$, and choose $K_{opt} = 6$. Figure 3.20 shows a two-dimensional representation of the Olivetti data set with the optimal parameter value. It can be seen that the illumination varies along the horizontal axis, and people's facial features change (from man with beard and without hair in the top to girl with long hair without beard in the bottom) on the vertical axis.

The experimental results of classification are shown in Figure 3.21. Among all the methods, Discriminant Isometric Mapping with the optimal parameter value achieves the lowest error rate. Similar results were obtained by Yang in the original Extended Isomap paper [133].

Overall, the extended Isomap with the optimal parameter outperformed PCA and LLE. It provided more accurate classification, and also resulted in better embeddings.

3.6.6 Classification Experiments: Handwritten Digits, MNIST Data

The MNIST database of handwritten digits was used for this classification experiment (http://yann.lecun.com/exdb/mnist/index.html). It consists of ten different classes of images from '0' to '9', 28×28 pixels each. In our experiments we used a MNIST database subset, which contains 5000 images, where the training set contains 2000 images and the test set contains 3000 images. The optimal neighbourhood region value for the extended Isomap, which was obtained using our algorithm, is $\epsilon = 9.6$. The embedded space formed by the two main dimensions is presented in Figure 3.22. Here the horizontal axis represents the digit shape, and the vertical axis - bottom

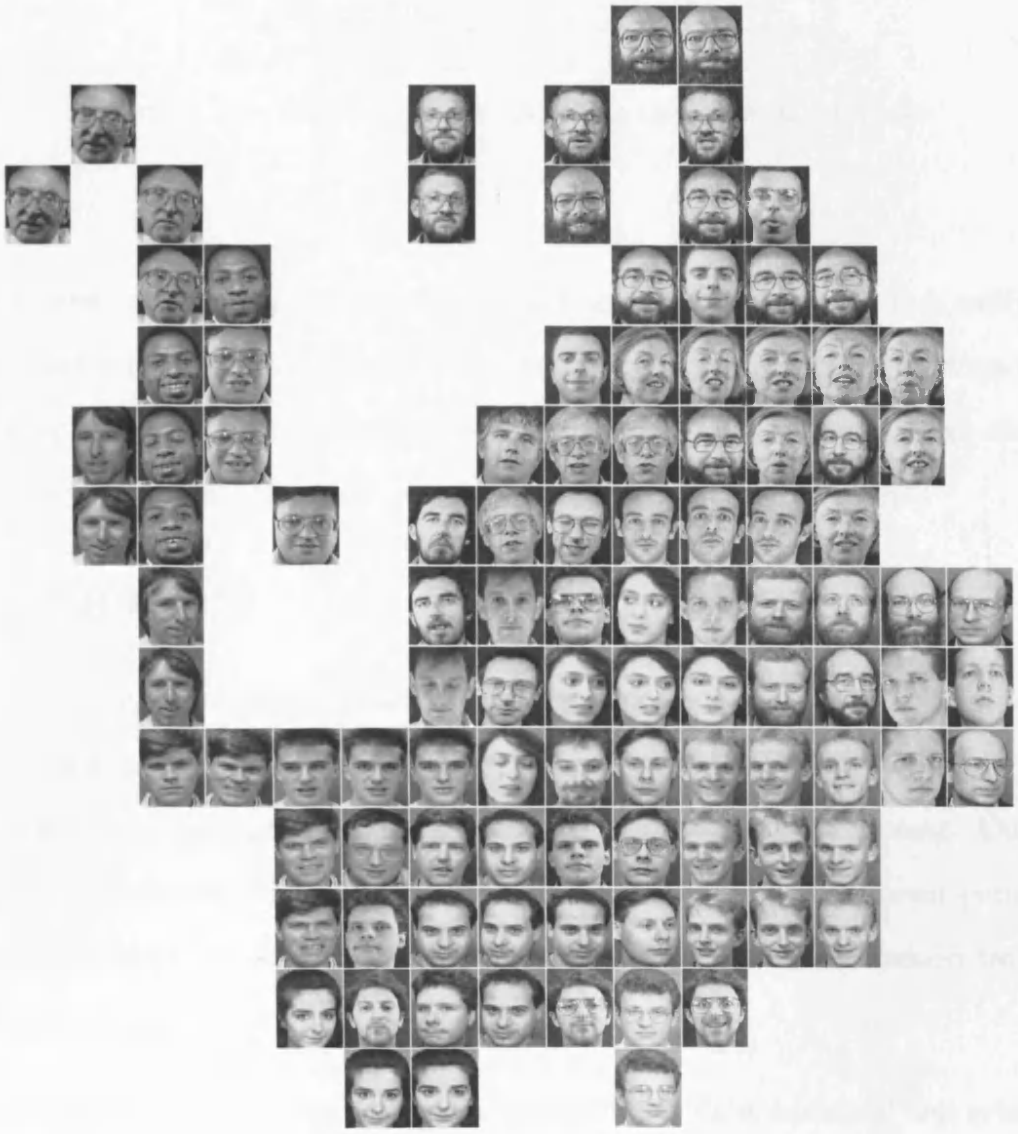

Figure 3.20: Two-dimensional Discriminant Isometric Mapping mapping ($K = 6$) of Olivetti data set

Method	Error rate (%)
PCA	2.4
best LLE, $K = 14$	2
best Ext. Isomap, $K = 6$	1.6
Ext. Isomap, $K = 7$	2

Figure 3.21: Classification results with the Olivetti database

loop.

The results of classification experiments are shown in Figure 3.23. It is easily seen that the extended Isomap with the optimal parameter outperformed PCA and LLE on this set. Note, that with this data set Isomap was less sensitive to the chosen ϵ parameter, when comparing this experiment with using other data sets.

3.7 Summary

Here we presented the dimensionality reduction process using Isomap and considered related problems. The main contribution of this Chapter is a novel method for automatic selection of the optimal neighbourhood parameter value for Isomap. Our idea is based on minimising the Isomap cost function estimation in a chosen parameter interval. We tested the algorithm on various data sets. Results obtained from our experiments suggest a number of conclusions:

- Our method can be used for a wide class of input data, both real and synthetic.
- When a data set is too sparse (for example swissroll built on 50 points), it is difficult to choose the optimal parameter. With $K = K_{min}$ ($\epsilon = \epsilon_{min}$), i.e. when the neighbourhood graph becomes connected, the graph already contains shortcuts and we obtain $K_{max} \leq K_{min}$ ($\epsilon_{max} \leq \epsilon_{min}$). In this case it will

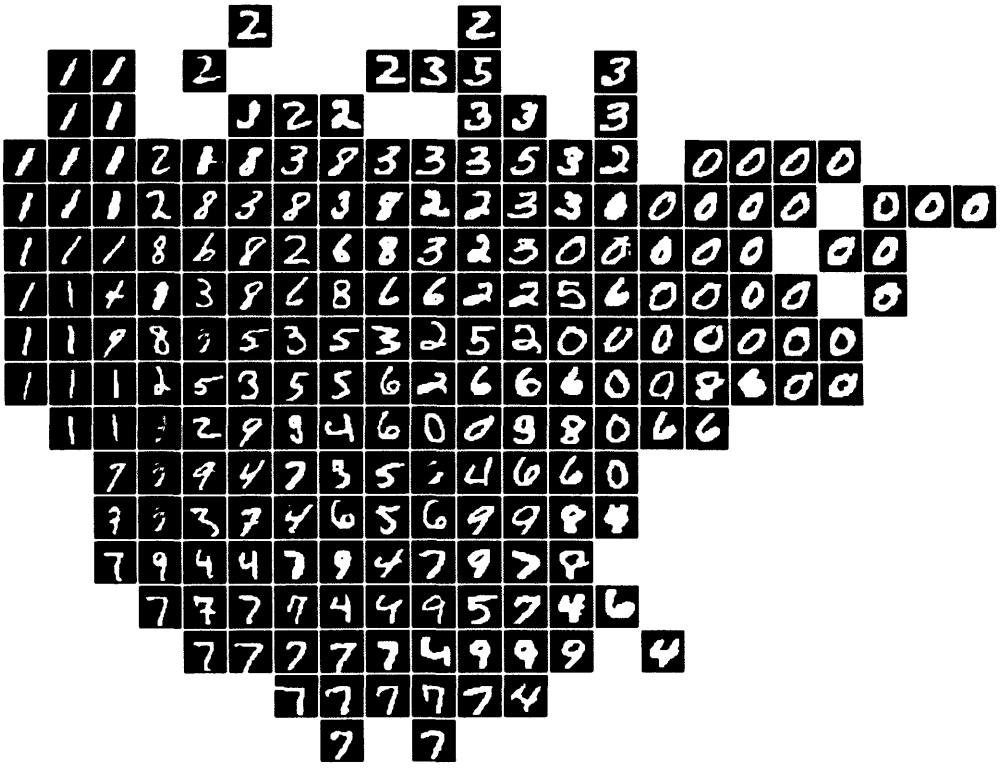

Figure 3.22: Two-dimensional Discriminant Isometric Mapping mapping ($\epsilon = 9.6$) of MNIST data subset

Method	**Error rate (%)**
PCA	3
best LLE, $K = 18$	2.7
best Ext. Isomap, $\epsilon = 9.6$	1.65
Ext. Isomap, $\epsilon = 10$	1.7

Figure 3.23: Classification results with the MNIST database

be better to choose another dimensionality reduction technique, for example Hessian LLE.

- Isomap performs better than PCA and LLE for non-linear data sets.
- The optimal parameter for modified Isomap version, Discriminant Isometric Mapping, can be chosen using our method, in the same manner as for Isomap.

In future work we hope to extend our method to choose the optimal parameter values for other parametric methods for dimensionality reduction of non-linear data, like Hessian LLE.

Chapter 4

Hierarchical Modelling of High Dimensional Data

4.1 Introduction

In the previous Chapter we constructed the low dimensional representation of high dimensional data in the Isomap space. For further data analysis, we need a manageable meaningful structure to see relationships between the data elements.

A hierarchical approach to data structuring allows greater accuracy when representing non-linear data from the real world. Hierarchical clustering builds a hierarchy of clusters which can be represented as a tree. The individual elements of the clustered data set form the leaves, whilst the root of the tree represents the whole data set. One of the hierarchy types is a binary tree, where the tree is represented by a dendrogram. A dendrogram provides a view of the data at different levels of abstraction. The consistency of the dendrogram at different levels allows flat partitions of different granularity to be extracted during the data analysis, making them ideal for interactive exploration and visualisation.

Many papers have been published on hierarchical clustering of data [11, 23, 71, 87].

In the above research the logical hierarchical decomposition of subspaces was known a priori and thus construction of the hierarchical models was relatively simple. In addition, many existing hierarchical clustering algorithms require some input parameters. Incorrect estimation of these parameters leads to poor clustering accuracy, a similar problem to the Isomap parameter estimation from Chapter 3. Furthermore, many clustering algorithms cannot handle different data types at the same time.

The weakness of the existing clustering methods forms the goal of this Chapter: *to provide an efficient, computationally inexpensive hierarchical clustering algorithm for high dimensional data with no a priori knowledge of underlying structure of the data.*

The proposed algorithm can be used in several ways, for data partitioning (as a tree-based classifier) or for data visualisation purposes. Also, having two data sets as a joint input, we can estimate the corresponding data element given a new unseen element using our model. For instance, if we have already built a hierarchial model for talking head data and we are given new audio data which was not used in the construction of this model, we can synthesise new video data, corresponding to the new input audio.

This algorithm is also *fully automated, easy to implement and does not require additional parameters. It can be applied to any number of data sets and any kind of data features (video, audio, motion, physical features, etc.).* In the case when we have several data sets it is important to scale the data spaces. Otherwise we can end up with a model where only the dominating data features are represented.

We consider the construction of our hierarchical model in Sections 4.2, 4.3, 4.4, 4.5. We explain finding the joint data with our model in Section 4.6 and demonstrate

the effectiveness of our algorithm applying it to the different data types: talking head, articulated human motion, MIDI and handwritten digits data sets in Section 4.7. The conclusion is given in Section 4.8.

4.2 Hierarchical Clustering Algorithm Overview

The main drawback of hierarchical clustering is that it is a computationally expensive procedure. Therefore we apply Isomap to the data first, it improves the computational efficiency and is semantically relevant at the same time, since the new reduced dimensions represent the most highly discriminating features of the original data. Then we construct a Gaussian Mixture Model (GMM) [9] in this low dimensional space to group the data. Finally we build the hierarchy using means of the GMM clusters. These means represent the main features in the initial data. Thus at the final step of the algorithm we have a limited set of variables which represent the model.

Suppose we have two sets of data features, denote them as c and m. The outline of our algorithm is shown in Table 4.1.

First of all, we apply the Isomap algorithm to the data features, as explained in Chapter 3. The success of Isomap in data representation and model accuracy depends on being able to choose an appropriate neighborhood size. We use the method for automatic selection of this parameter described in the previous Chapter. As a result of applying dimensionality reduction, we have the new parameter a, which corresponds to the initial data features embedded into the low dimensional space. This parameter contains the knowledge of the whole data set and controls the data variations.

In the following Sections we consider hierarchy construction in more detail.

Input: Two joint data features $c = \{c_i\}_{i=1}^N$ and $m = \{m_i\}_{i=1}^N$
Output: Hierarchical data model, dendrogram

1. Find Isomap parameters using algorithm 3.2.
2. Apply Isomap to $c \in \mathbb{R}^{D_c}$ and $m \in \mathbb{R}^{D_m}$:
   get $I_c \in \mathbb{R}^{d_c}$, $I_m \in \mathbb{R}^{d_m}$, where $d_c \ll D_c$ and $d_m \ll D_m$.
3. Combine I_c and I_m parameters as
   $a = [I_c\ I_m] = \{a_i\}_{i=1}^N$.
4. Find the optimal number of clusters k for a, using the Characteristic Cost Graph approach.
5. Separate a into k clusters, using GMM and EM algorithm: $a'_1, ..., a'_k$.
6. Calculate the mean for each cluster, $\tilde{a}'_1, ..., \tilde{a}'_k$.
7. Apply agglomerative hierarchical algorithm, use the clusters means $\tilde{a}'_1, ..., \tilde{a}'_k$ as initial nodes.
8. Plot dendrogram.

Table 4.1: Hierarchical modelling algorithm

4.3 Selecting an Appropriate Number of Clusters

Finding the correct number of clusters is a difficult general problem even when selecting manually. If the data is labelled, then common information between cluster and class labels can be used to determine the number of clusters. However, more often, little is known about the nature of the data and a method of estimating the number of clusters is required.

Bowden in [12] proposed a simple but effective method for estimating the number of clusters k for unlabelled data. According to this method, we estimate the cost

function

$$D = \frac{1}{N}\sum_{i=1}^{N} d_{min}(a_i) = \frac{1}{N}\sum_{i=1}^{N} min_{z_i \in z}(d(a_i, z_j)) \ , \tag{4.3.1}$$

where $z = \{z_i\}_{i=1}^{k}$ is a set of cluster centers, and $d(a_i, z_j)$ is the squared Euclidean distance between the parameter a_i and the cluster center z_j.

We use this cost function to produce the Characteristic Cost Graph shown in Figure 4.1. Bowden defines the optimal number of clusters by locating a point where increasing the number of clusters does not lead to a significant decrease in the resulting cost. We perform a more accurate and reliable estimation of that point using the procedure for determining thresholds, described in [108]. The threshold point is selected as the number of clusters k' that maximises the perpendicular distance between the line and the point $(k, D(k))$, where $D(k)$ is the cost function for k clusters, see Figure 4.1.

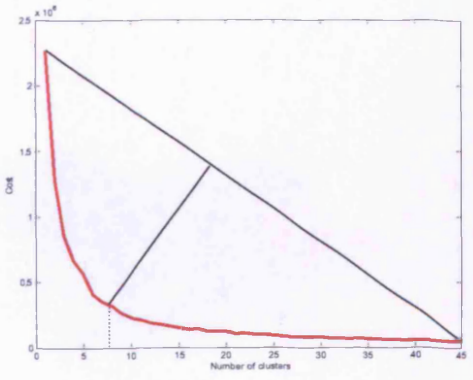

Figure 4.1: Characteristic Cost Graph with the selected threshold

4.4 Data Modelling as a Gaussian Mixture Model

Gaussian Mixture Modelling is a flexible and powerful method for unsupervised data grouping [89]. In addition to grouping, it gives us parameters which we will use

further to initialise our dynamic model in the next Chapter.

A k-component GMM may be defined in terms of the distribution of the initial parameters in the low dimensional space as

$$p(a) = \sum_{i=1}^{k} \alpha_i N(\mu_i, S_i) \ , \tag{4.4.1}$$

where $p(a)$ is the probability that a point in the low dimensional space is generated by the model, α_i is the prior probability and constitutes the mixture coefficients, and $N(\mu_i, S_i)$ describes each Gaussian distribution with mean μ_i and covariance matrix S_i. GMM parameters α_i, μ_i and S_i are estimated from the low dimensional embedded space a through traditional Expectation Maximisation (EM) approaches [9]. An example of a GMM in the embedded Isomap space for the talking head data is shown in Figure 4.2, also see Section 4.7.1.

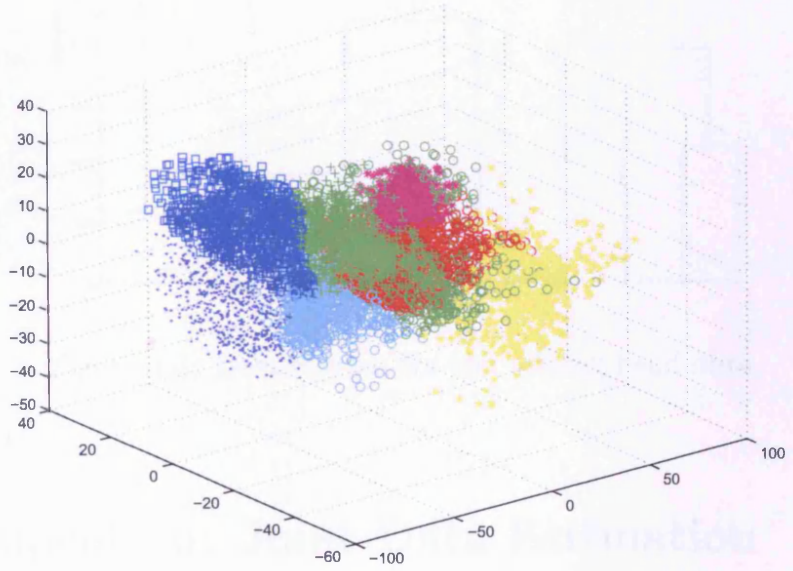

Figure 4.2: GMM in the reduced dimensionality eigenspace (three highest modes of variation) modelling talking head data

4.5 Hierarchical Agglomerative Clustering

To construct a hierarchy over the obtained clusters, we employ the standard agglomerative clustering algorithm [40] because it is a simple, effective, fast and non-parametric method. We use the means of the k clusters $\tilde{a}'_1, ..., \tilde{a}'_k$ as input data for the agglomerative algorithm. This approach attempts to place the input elements in a hierarchy in which the distance within the tree reflects similarity between the elements. The result of the hierarchical clustering is represented as a dendrogram tree, see Figure 4.3 for the example of the talking head data dendrogram.

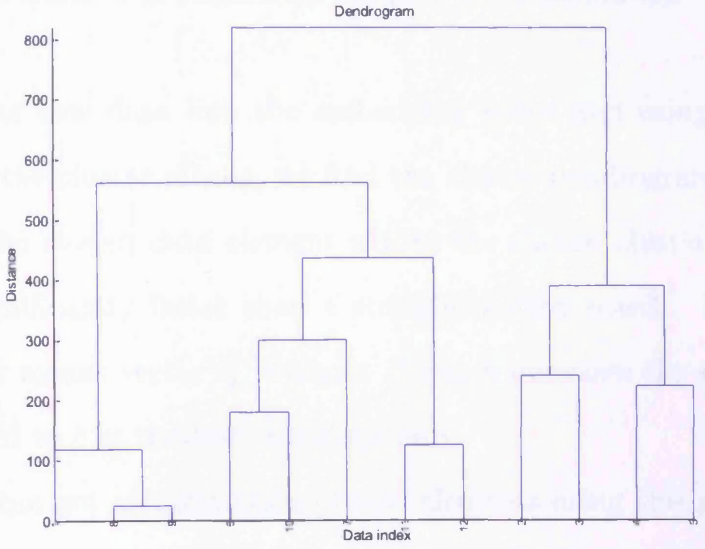

Figure 4.3: Dendrogram for the talking head data

4.6 Application: Joint Data Estimation

Here we give an example of our algorithm's application: given a new unseen data element from one set of the joint data sets m_{new}, we estimate a corresponding data element from the second set c_{new}. The algorithm is presented in Table 4.2.

Input: New element m_{new};
Isomap projections of the training data $a = [I_c\ I_m]$, labelled with the cluster numbers; the dendrogram constructed using algorithm 4.1 with node values equal to the cluster means I'_m.
Output: Estimated c_{new}.

1. Project m_{new} into I_m using kernel function, get $\tilde{I}_m$.
2. Start from the dendrogram root, find the closest to $\tilde{I}_m$ bottom node value $\{I'_m\}_l$.
3. Among the parameters I_m labelled with l, find the closest to $\tilde{I}_m$ parameter, I''_m.
4. Choose $\tilde{I}_c$ equal to I''_c, where I''_c corresponds to the I''_m parameter.
5. Set c_{new} to be a back projection of $\tilde{I}_c$.

Table 4.2: Algorithm for joint data estimation

We project the new data into the embedding space and using the dendrogram constructed over the cluster means, we find the closest dendrogram node value. We then search for the closest data element within the chosen cluster (step 3 in Table 4.2), which is significantly faster than a straightforward search. Note that we use part of the cluster means vector I'_m without I'_c which increases the speed. We employ parameters related to c in the last two steps only.

Although we can get an estimation of new elements using this general algorithm, there are several issues regarding the quality of the final result. Let us consider the talking head example with two data features: speech and images. In the case of a single speech element we get a single image as an output, which is the corresponding image of the closest speech element from the training set. But for the same speech element there may exist several images with opened or closed mouth, blinking, etc. Blinking, a most evident change in the upper part of the face, is a recurrent phenomenon and does not depend on the speech. There are a number of research papers

dedicated to this problem, see [24, 28]. The problem of blinking can be solved using a priori data, which we do not consider in this thesis. Cosker in [27] created a robust algorithm for modelling the talking head. He manually divided the face into regions: mouth, eyes, cheeks, etc. and modelled each face part individually. We propose an automatic algorithm for data segmentation with no a priori knowledge in Chapter 7 of this thesis, which can be used as a preprocessing step for talking head analysis with the Cosker algorithm to make the process automated.

4.7 Experiments with a Real World Data

Here we evaluate how the proposed algorithm processes different data types obtained from the real world. The first data considered, in Section 4.7.1, is the talking head. In this example we show how our algorithm works with large data with several parameter sets. Section 4.7.2 describes the model visualisation ability with 3D coordinates of a walking person. Next, a hierarchical model is built for a MIDI music collection, representing the melodies according to their style. And finally, a model is constructed for the handwritten digits from the MNIST collection. The last two examples, MIDI and handwritten digits, are naturally labelled therefore we use them to test the classification ability of our method. The classification results are compared to the results achieved by other researchers for this data.

4.7.1 Talking Head Data Set

We test the algorithm described above on the data from [28]. Initially, it was 8500 frames video of a speaker reading a text, recorded at 25fps with mono audio sampled at

32KHz. The subject was recorded front-on with as little out of plane head movement as possible. An Appearance Model [25] was built to represent image shape and texture variation. Mel-Cepstral analysis was applied to the corresponding audio data associated with each video frame. Figure 4.4 shows examples of the input data. Thus, we have the normalised set of appearance parameters and associated Mel-Cepstral coefficients to build the hierarchy.

Figure 4.4: Examples of the texture, shape and Mel-Cepstral coefficients for the talking head data

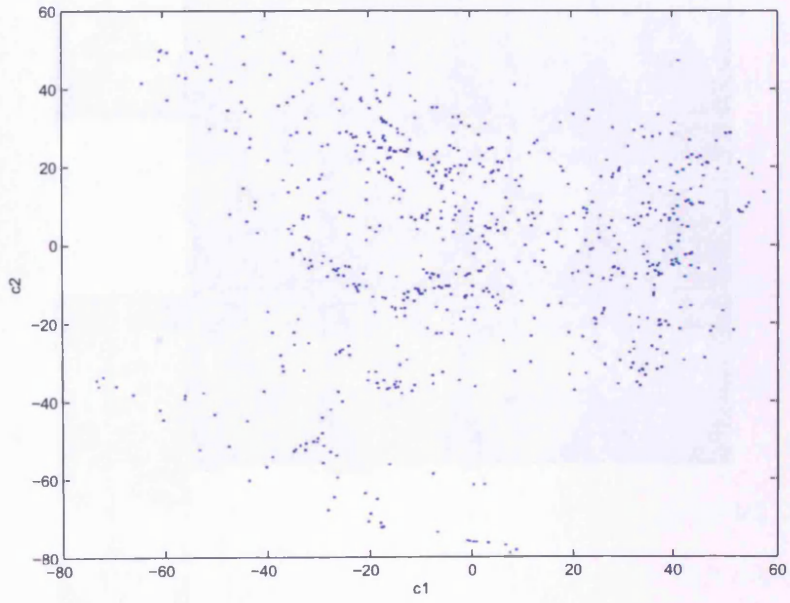

Figure 4.5: Distribution of appearance parameters

From Figure 4.5 it is seen that our parameter distribution is highly nonlinear.

First, the Isomap method is applied to extract major features from the parameter sets, and we obtain a 22-dimensional embedding of the video data with the main variations corresponding to mouth shape changes. Figure 4.6 shows the face changes in the first two dimensions of the embedding obtained from Isomap. It can be seen that the first embedding coordinate corresponds to open-close mouth changes on the horizontal axis, and mouth shape changes on the vertical axis.

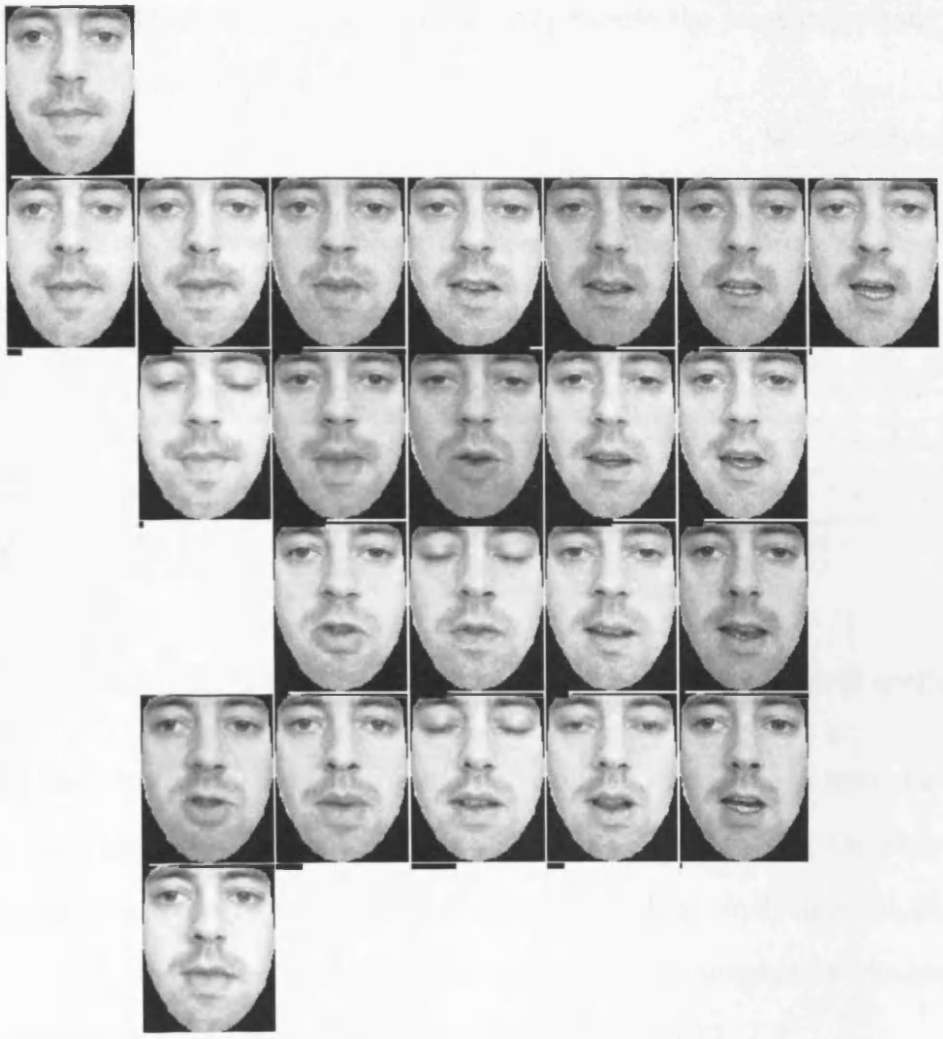

Figure 4.6: Two-dimensional Isomap mapping of the appearance parameters

We embed the speech coefficients into a three dimensional Isomap space. The first embedded audio feature represents the changes from voiced vowels (a, o, i) to unvoiced consonants (p, k, t). For the second feature we have changes from voiced to unvoiced vowels, and changes from unvoiced vowel (e, u) to voiced consonant (b, m, v) for the third feature. Figure 4.7 shows Mel-Cepstral coefficients (represented as a bars) mapped into the first two Isomap embedding coordinates. So far, at this stage we have two sets of parameters, each of them represents the most important features, extracted from the initial data.

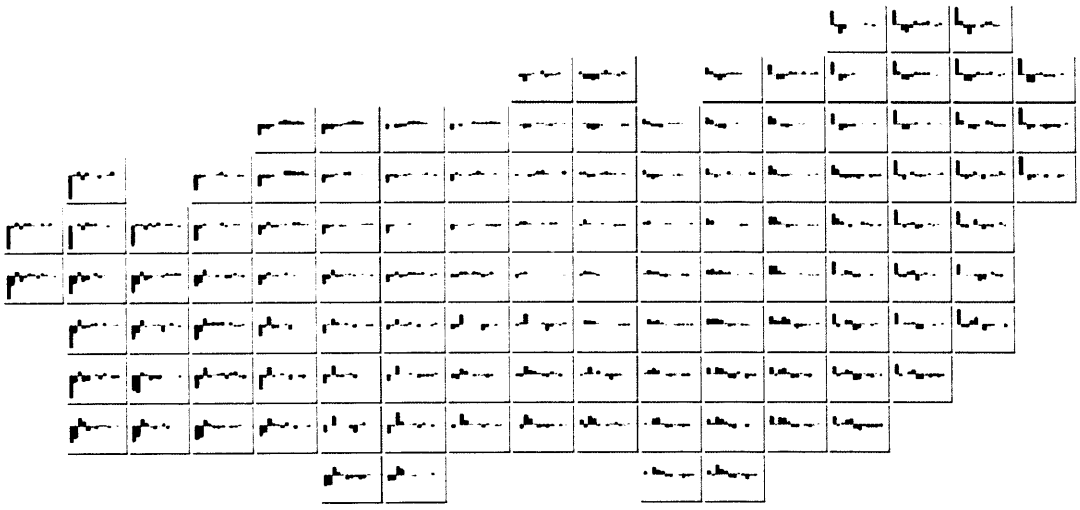

Figure 4.7: Two-dimensional Isomap mapping of the Mel-Cepstral coefficients

During the next step we split the merged Isomap embeddings into clusters. According to our algorithm, we choose the number of clusters using the characteristic graph produced for the model. A value is calculated by analysing the plot of the resulting overall cost of a converged solution against the number of clusters k. The optimal number of clusters for the talking head model is 12.

Using this optimal value of k, a Gaussian mixture model is built for the parameter a in a low dimensional manifold. Figure 4.2 illustrates the mixture of Gaussians in

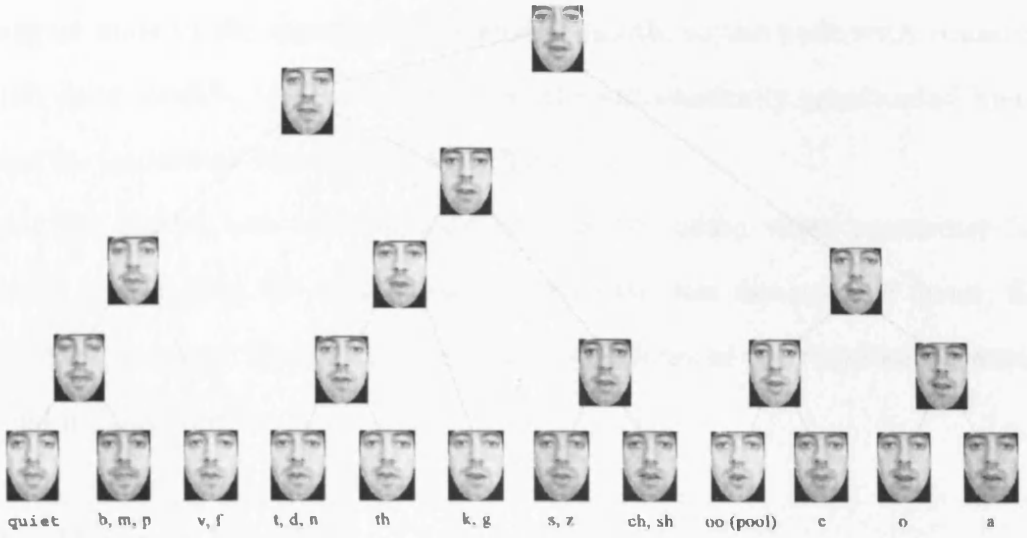

Figure 4.8: Hierarchical model representation of the talking head data

the embedded space. The highest joint modes of variation for this example are the first audio feature (voiced vowels - unvoiced consonants), and the first two appearance parameters features (from Figure 4.6). Each of the obtained clusters represents initial data features: mouth shapes, speech variations; the largest clusters are silence, or "Quiet" (yellow stars), consonants "b, m, p" (red circles), "a" (blue squares), "o" (blue dots), "e" (green circles), and "oo" (cyan circles). Generally, we got the audio feature domination in the clustering result.

For each of the 12 clusters obtained in the previous step we take the mean and use it as the input for the agglomerative algorithm. This produces the hierarchical model shown in Figures 4.3 and 4.8. In Figure 4.8 each face at the bottom level represents one cluster mean.

This hierarchical model has relatively good accuracy. The bottom level of the model presents main phonemes, which are grouped according of their similarity. The images are the corresponding visemes. The bottom nodes change from the Quiet node

(the biggest node in the model) with a closed mouth, to the node with voiced vowel *a* and an open mouth. It can be seen that the automatically constructed hierarchy captures the important features of the talking head.

Using the model, one can estimate the corresponding video parameter for the new audio by mapping the new parameter into the low dimensional space, finding the closest cluster and the low dimensional coordinates of the required parameter as explained in Section 4.6.

4.7.2 Human Motion

This data represents motion of a walking person, consisting of two steps, a right swing and one step. The initial feature parameters represent the coordinates of the human (arms, legs, torso) in 3D space. Each pose is characterised by 17 points. There are 218 poses contained in this data set.

We reduce the dimensionality of the initial data space to 3, and perform clustering. Figure 4.9 illustrates the mixture of Gaussians in the embedded Isomap space. As one can see from the Figure, the data hierarchy should have 10 clusters at the bottom. Figures 4.10 and 4.11 demonstrate the visual representation of the data in the Isomap space. According to these figures, our first dimension corresponds to the motion direction: go left - go right. The second dimension is the leg position: left leg in front - right leg in front. Finally, the third dimension shows the step stages: from the left to the right heel strike (Figure 4.11, vertical axis).

The hierarchical model is shown in Figures 4.12, 4.13 (the visual representation), and Figure 5.2 (the dendrogram). The small pictures in Figures 4.12 and 4.13 represent the cluster means. Let us consider the model in more details below.

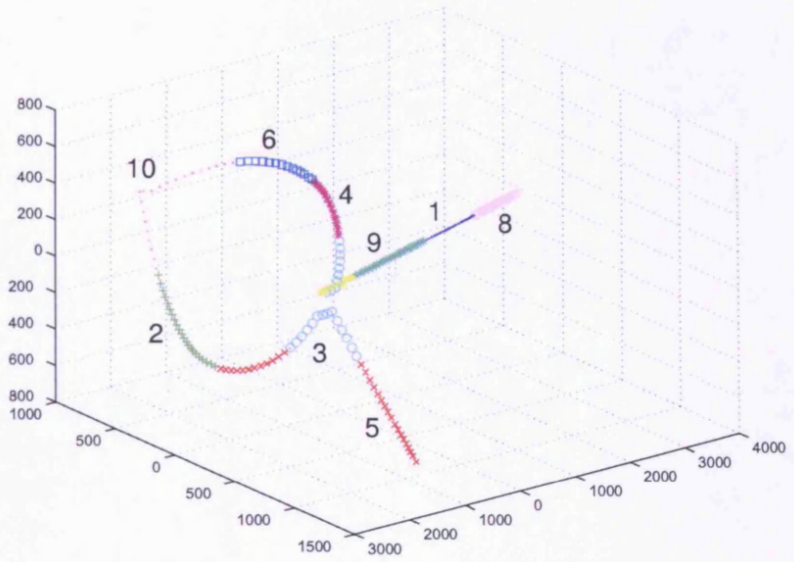

Figure 4.9: GMM in the Isomap space. Different colours correspond to different clusters.

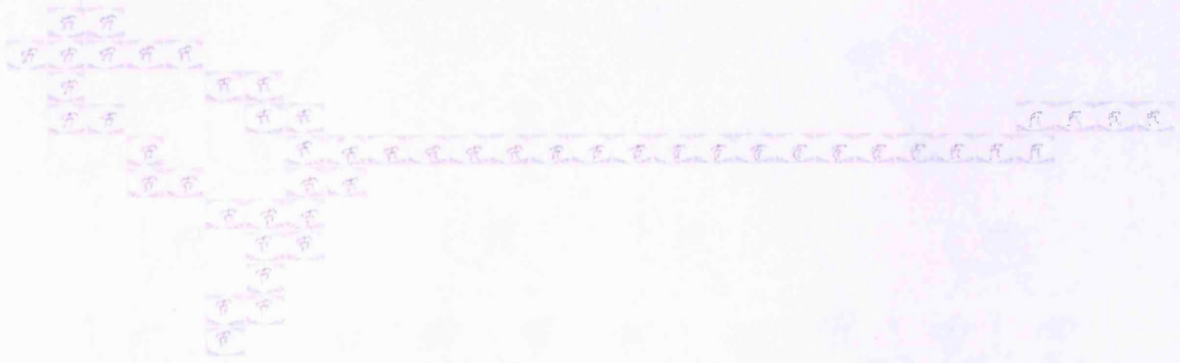

Figure 4.10: Walking data projection into Isomap space (first two dimensions), visual representation.

Figure 4.11: Walking data projection into Isomap space (second and third dimensions), visual representation.

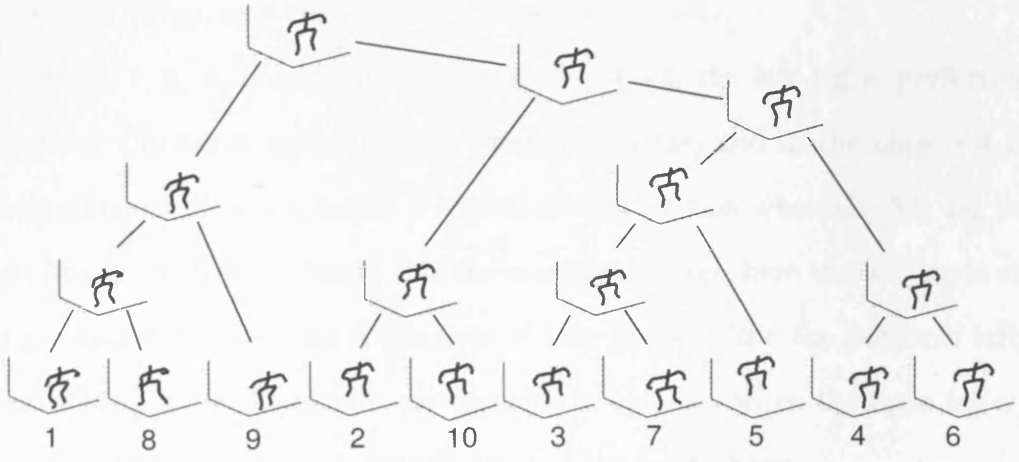

Figure 4.12: Hierarchical model representation of the human motion, 3D view

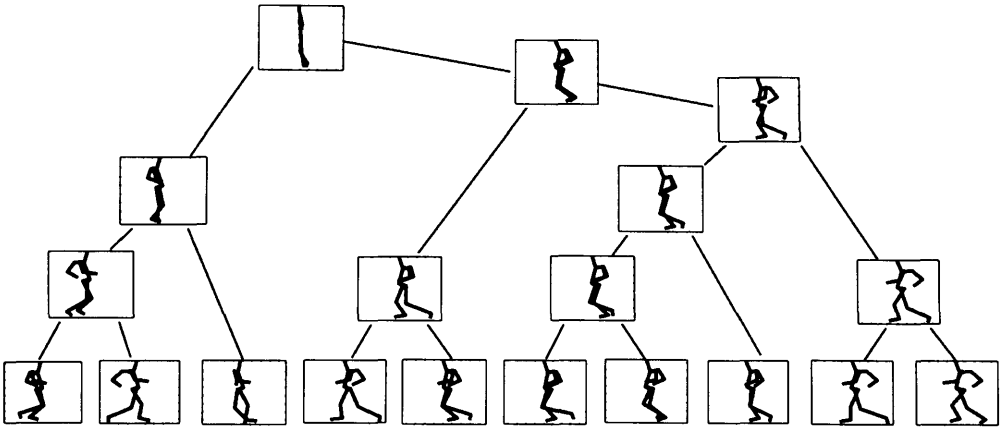

Figure 4.13: Hierarchical model representation of the human motion, side view

Clusters 2, 10, 3, 7, 5, 4, 6 lie in one branch of the hierarchy and represent the first two steps. Clusters 1, 8 and 9 are in another branch. Clusters 1 and 8 represent the last step, and cluster 9 corresponds to the swing and the last step beginning.

Cluster 1 corresponds to the left leg moving down and the right leg lifting up motions, and cluster 8 represents the right leg moving down and the left leg bending motions. Clusters 2 and 10 are similar in meaning to clusters 1 and 8 respectively, but before the swing, and the motion of legs are opposite.

Clusters 3, 7, 5, 4, 6 correspond to the case when the left leg is performing the main motion. Cluster 6 represents the right heel strike, and at the cluster 4 the left leg is beginning to lift up. Cluster 3 represents the motion when the left leg replaces the right one in the front. Cluster 7 is the smallest cluster, here the left leg is starting to swing. And finally, cluster 5 consists of two parts. Here the person's left leg is lifting up. One part of the cluster corresponds to the case when the right leg is in the front, and another part corresponds to the left leg in the front.

This model is a good example of the hierarchical data decomposition. The clusters are disposed logically, one tree branch represents the human moving to the camera,

another branch represents the human moving away from the camera. The bottom nodes introduce phases of a single step. These nodes are merged to represent the movement in a more general way, right leg movement, left leg movement, etc. Each motion is represented quite well, and it will be easy to classify a new motion with such a hierarchy.

4.7.3 MIDI Data

For this experiment we downloaded 178 separate MIDI files selected from a range of classical composers, folk music and some popular music. There are Chopin, Haydn, Beethoven, Bach and Mozart compositions in the first group, Irish and Chinese folk in the second group, and Beatles, Abba, Queen and Jean Michel Jarre in the third group.

To make these MIDI files more uniform, we use the preprocessing procedure described in [23]. The preprocessor only keeps the basic MIDI-track decomposition as well as note timing and duration events. The preprocessed MIDI files are then used as the input for our algorithm.

After applying our algorithm to the input data we obtain the hierarchy shown in Figures 4.14 and 4.15. There are two main branches in the tree. The first branch combines the classical composers and the folk music, another branch represents the popular music. Indeed, there are much greater differences between the classical and popular music than between the classical and folk or between folk and popular. The mixture of classical and folk music at the first branch can be explained by the character of the selected Bach and Mozart pieces. The bottom nodes of the hierarchy represent the composers. Using the name of the composer as a natural class label we perform

classification with the MIDI data.

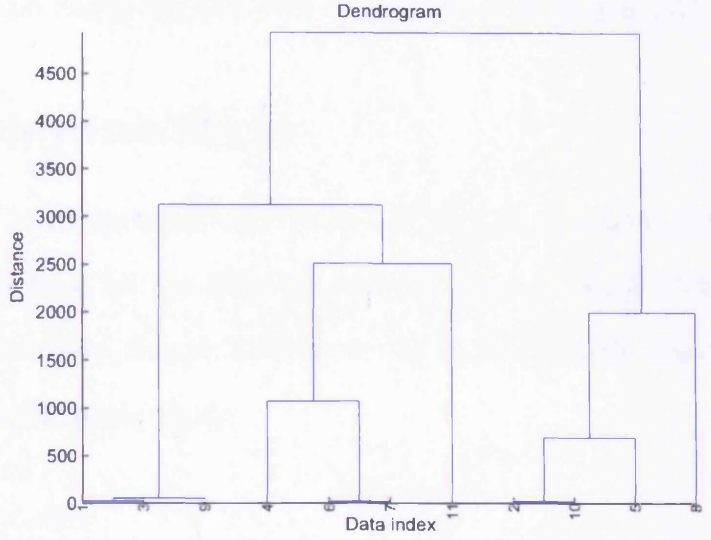

Figure 4.14: Dendrogram for the MIDI data

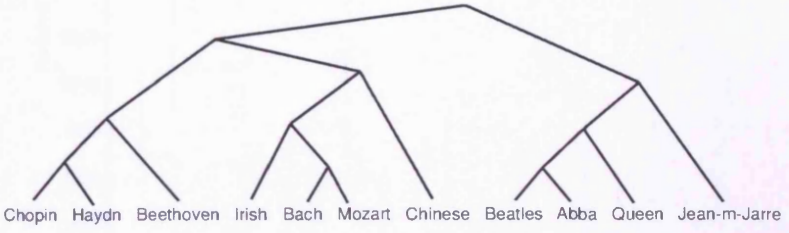

Figure 4.15: Hierarchical model representation of MIDI data

To test the effectiveness of our hierarchical model we classify new pieces of music. The test set consists of 110 MIDI melodies, with 10 melodies in each category. After the preprocessing we project them into the embedded Isomap space as explained in Chapter 3.4. Then we use these projections to find the closest cluster. Starting from the tree root, represented by the training data mean, we descend down the tree and find the closest cluster at the bottom level. The achieved accuracy is 92.73% (8

music pieces are misclassified). The obtained result is similar to the best performance reported by Li and Sleep in their melody classification paper [91].

4.7.4 Handwritten Digits

We constructed a hierarchical model for the MNIST database of handwritten digits subset, which we used for the Isomap experiments in Chapter 3.6.7. With our algorithm we get 10 clusters, Figure 4.16 shows the automatically constructed hierarchical model of the handwritten digits.

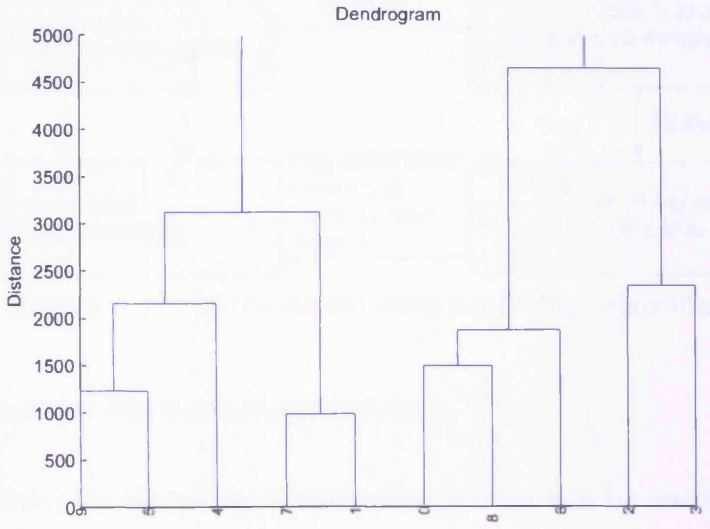

Figure 4.16: Hierarchical model representation of MNIST data (digits from 0 to 9 in the bottom line)

We use a new set of 1000 images for the classification test. In this experiment the recognition rate was 93.5% (65 images were misclassified). The resulting error rates on this data set vary from 0.7% to 12% as reported in [79]. This shows that although we use a generic algorithm, we achieve quite a low error rate.

4.8 Summary

We presented a novel method for the automatic construction of a hierarchical model using clustering and non-linear mapping outlined in Figure 4.17. During the first step of our algorithm we reduce the high dimensional data parameter space into a low dimension manifold using Isomap. Then we construct a GMM in this manifold. Next, agglomerative clustering is applied to the set of Gaussian means and the clustering structure is represented as a dendrogram.

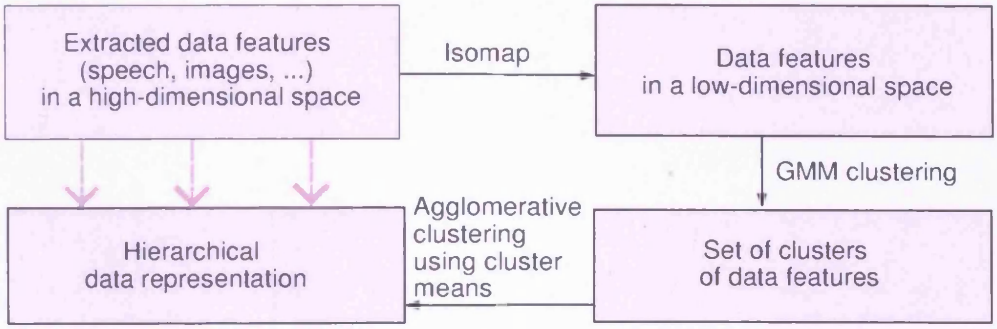

Figure 4.17: Hierarchical data modelling algorithm

We now summarise our main contributions:

- A new method for hierarchy construction which can be used with a wide range of high dimensional data.
- The initial data can be represented as several corresponding data sets.
- Our algorithm is fully automated and is easy to implement. There are no additional configuration parameters required.
- The data representation as a small set of Gaussian means allows efficient use of the clustering algorithm to divide the prototypes into groups. The reduction

of the computational cost is especially important for hierarchical algorithms allowing clusters of arbitrary size.

- The algorithm is useful for different purposes: data visualisation, as a data classifier and for generation of new data.

Chapter 5

Developing a Dynamic Framework for the Automatic Hierarchy Construction Algorithm

5.1 Introduction

In Chapter 4 we constructed a hierarchical model of the high dimensional non-linear data with an unknown structure. The constructed model is a *static* model, *i.e.* it does not contain any explicit information on the temporal relationship between the data points. Understanding the temporal structure of the initial data sequence is important for recognising and representing dynamic data reliably. In this chapter we introduce a new two-stage algorithm for dynamic data modelling. We use the hierarchical model obtained from Chapter 4 as the first stage of the proposed algorithm, and at the second stage of the algorithm we introduce dynamics by involving a Hierarchical Hidden Markov Model (HHMM).

In our work, we employ a HHMM as a technique for data analysis that can work across a range of applications. The ability to automatically learn structures and models from many kinds of input data feature sequences using a single algorithm

gives great advantage for use in many applications.

The HHMM is a generalisation of the widely used Hidden Markov Model (HMM), where HMMs form a hierarchy. The hidden state of an HHMM can emit a single observation like an HMM, or a string of observations by entering lower level HMMs. The main problem with HHMMs is their topology definition, i.e. how to set the number of HMMs, the number of states in each HMM, the allowable horizontal transitions within each HMM, and the allowable vertical transitions between levels. Usually, the hierarchical structures (topology) of HHMM is defined manually to be used only for particular data. Here we present an automated HHMM topology definition algorithm, which is independent from the initial data.

We consider our algorithm as *blind content processing*, in which the first stage is unsupervised data content characterisation, and the second stage is 'supervised' pattern discovery based on characterisation from the first stage. We find that this is more effective than using a purely unsupervised approach for the pattern discovery problem. The intention is that unsupervised learning provides a better initialization for supervised learning, and hence a better final local minimum. *Our model is fully automated and does not require additional parameters.* To the best of our knowledge, no significant work has been reported for our problem of interest which is able to work as quickly and effectively.

The algorithm is based on Isomap's ability to preserve data geometry at all scales [114]. Using this property, we find the data trajectory in the embedded space. We also employ the assumption of data trajectory smoothness during analysis of the space. Under these conditions we propose that utilising hierarchical clustering we can construct a HHMM topological structure, and the obtained HHMM is efficient in

exploring semantic patterns in the input data. We build a HHMM using our algorithm from the test data and learn the model with Dynamic Bayesian Networks (DBN) [33]. After training the model we can explore observed patterns from the new unseen data.

In the following Sections we summarise briefly the theory of HHMM's and build a HHMM with the structure determined from our hierarchical algorithm. The following Chapter 6 will demonstrate the effectiveness of our algorithm by applying it to different motion data types.

5.2 HHMM: Definition and Representation as a DBN

Here we give a formal definition of HHMM, explain the meaning of each HHMM parameter and discuss its analysis and estimation. Since HHMM consists of HMMs and contains all HMM parameters, we start by considering the basics of HMMs.

5.2.1 Hidden Markov Model

A Hidden Markov model consists of a Markov chain with a finite number of states, a state transition probability matrix and an initial state probability distribution. Although the states are hidden (not directly observable), each state generates observations that are drawn according to some probability distribution (either discrete or continuous) [104]. A HMM is characterised by the following elements:

- The number of states m in the model. The states are hidden, and may be represented by Gaussian mixtures for the continuous data. The states are defined $Q = Q_1, Q_2, ..., Q_m$, and the state at time t is Q_t.

- The state transition probability distribution $A = \{a_{ij}\}$, where

$$a_{ij} = P(Q_{t+1} = j | Q_t = i), 1 \leq i, j \leq m \tag{5.2.1}$$

- The observation probability distribution in state j, $B = \{b_j(O)\}$, where

$$b_j(O) = P(O_t | Q_t = j), 1 \leq j \leq m \tag{5.2.2}$$

- The initial state distribution $\Pi = \{\pi_i\}$, where

$$\pi_i = P(Q_1 = i), 1 \leq i \leq m \tag{5.2.3}$$

Using a shorthand notation, a HMM is defined as the triplet

$$\lambda = (A, B, \Pi) \tag{5.2.4}$$

5.2.2 Hierarchical Hidden Markov Models

A Hierarchical Hidden Markov Model (HHMM) was first introduced by Fine in [44] as a natural generalisation of HMM with a hierarchical control structure. The difference between a standard HMM and a HHMM is that states in the hierarchical model can produce a sequence of observations, whereas each state in the standard model produces a single observation.

Figure 5.1 shows an example of a HHMM state transition diagram. Basically, it consists of two HMMs with three and two states at the bottom level respectively, and one two-state HMM at the top level. We start from the top level HMM, which calls its sub-HMM from the bottom level, this emits observations and returns control to the top-level HMM again. Now, according to the transition matrix, we can go to the another state of the top level HMM or enter the end state, which terminates the process.

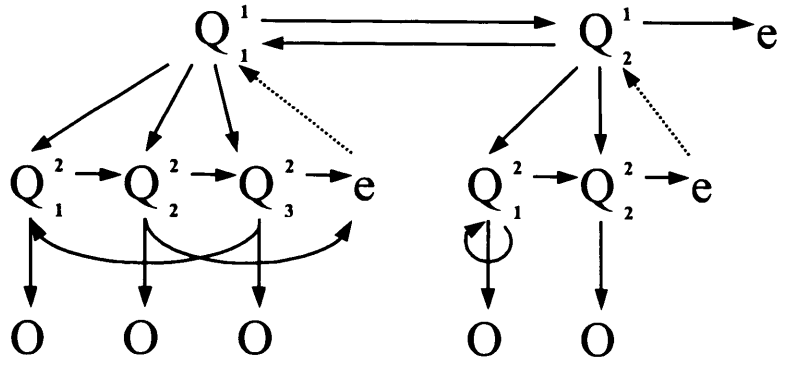

Figure 5.1: A two-level HHMM with observations at the bottom. Black edges denote vertical and horizontal transitions between states and observations. Dashed edges denote returns from the end state of each level to the level's parent state.

There are two types of hidden states: "production states", which emit single observations (Q_1^2, Q_2^2, Q_3^2 in Figure 5.1 are production states), and "abstract states", which contain production states or other abstract states (represented in Figure 5.1 as Q_1^1 and Q_2^1). Each production state is associated with an observation vector, which maintains distribution functions for each observation defined for the model (denoted by O in Figure 5.1). Each abstract state is associated with a horizontal transition matrix, and a vertical transition vector. The horizontal transition matrix of an abstract state defines the transition probabilities among its children. The vertical transition vectors define the probability of an abstract state to activate any of its children. Each abstract state is also associated with a child called an end state which returns control to its parent. The end states (e in Figure 5.1) do not produce observations and cannot be activated through a vertical transition from their parent.

Let us give the formal HHMM definition.

- **States.** The difference between a standard HMM and a hierarchical HMM is that individual states in the hierarchical model can contain sequences of nested

states or observations, whereas each state in the standard HMM can contain only linear sequences of observations, due to its linear nature. Therefore in the formal HHMM description instead of hidden states sequence $Q = Q_1, Q_2, ..., Q_m$ we have a vector $\overrightarrow{Q} = (Q_1^d, Q_2^d, ..., Q_m^d)$, where $d = 1, ..., D$ is a hierarchy level.

- **The state transition probability distribution.** For each abstract state $Q^d, d \in 1, ..., 1 - D$, the probability of making a horizontal transition from the ith state to the jth, where both are siblings of Q^d, is defined by the state transition probability matrix $A^d = \{a_{ij}^d\}$, where

$$a_{ij}^d = P(Q_j^{d+1} | Q_i^{d+1}), 1 \leq i, j \leq m \tag{5.2.5}$$

- **The observation probability distribution.** The production states are parameterised solely by the output probability vector $B^D = b^D(k)$, which is the probability that the production state Q^D will output the observation O_k. Here we have

$$b^d(k) = P(O_k | Q^D) \tag{5.2.6}$$

- **The initial state distribution.** The probability that state Q^d will initially activate the state Q_i^{d+1} $\Pi^d = \pi^d(Q_i^{d+1})$, where

$$\pi^d(Q_i^{d+1}) = P(Q_i^{d+1} | Q^d), 1 \leq i \leq m \tag{5.2.7}$$

The entire set of HHMM parameters, by analogy with Equation 5.2.4, is denoted by

$$\lambda = \{\lambda^d\}_{\{d \in 1,...,D\}} = (\{A^d\}_{\{d \in 1,...,D-1\}}, \{B^D\}, \{\Pi^d\}_{\{d \in 1,...,D-1\}}) \tag{5.2.8}$$

To define a HHMM fully, we should also set a topological structure ζ and an observation alphabet Σ. The topology ζ specifies the number of levels D, the state space at each level, and the parent-children relationship between levels d and $d+1$. The observation alphabet Σ consists of all possible finite observation strings $\overrightarrow{O} = O_1, O_2, ..., O_m$. A full HHMM is defined as a 3-tuple

$$H = \langle \lambda, \zeta, \Sigma \rangle \tag{5.2.9}$$

5.2.3 Analysis and Estimation of HHMM

There are several key issues in the analysis and estimation of the HHMM parameters in Equation 5.2.9. First, we should solve three classical problems of the Hidden Markov Models analysis for λ from Equation 5.2.8:

- **The evaluation problem:** given the HHMM parameter set λ and the observation sequence $O = O_1, O_2, ..., O_T$, how can we efficiently compute $P(O|\lambda)$, the probability of the observation sequence given the model?

- **The decoding problem:** given the HHMM parameter set λ and the observation sequence $O = O_1, O_2, ..., O_T$, how can we find the state activation sequence, that is most likely to generate the observation sequence?

- **The estimation problem:** given the structure of an HHMM ζ and the observation sequence $O = O_1, O_2, ..., O_T$, how can the HHMM parameter set λ be found which optimise $P(O|\lambda)$?

Solutions for the above HHMM problems are more complicated than for the standard HMM because of the HHMM's hierarchical structure. For the original HHMM

the solutions are given by using the well-known forward-backward, Baum-Welch and Viterbi algorithms [44]. The overall complexity of the original algorithm is $O(QT^3)$, where Q is the total number of states and T is the length of observation sequence.

Murphy in [92] proposed to convert the HHMM into a DBN, and apply general DBN inference to the model to estimate the parameter set λ and to achieve the overall complexity $O(TD^2Q^2)$, where T is the length of the observation sequence, D is the depth of the hierarchy and Q is the total number of states. We use the efficient Murphy solution briefly described in Section 5.5, full details of which can be found in [94].

We assume that observation alphabet Σ from Equation 5.2.9 is represented by the input data. To define the HHMM parameters in Equation 5.2.9 entirely, we need to specify another HHMM parameter, ζ. Commonly, this parameter is specified manually to be used only for a particular data set. In this thesis we consider algorithms for data with no *a priori* knowledge, therefore we create a general solution, which is able to work with many kinds of data automatically. We present this solution in the following Section.

5.3 Automatic Discovery of HHMM Hierarchical Structure

The automatic discovery of a HHMM's hierarchical structure ζ from initial data is an important yet complicated problem. Some work in this area has been done by Xie et al. [130], and Youngblood and Cook [136]. The structure discovery algorithm by Xie et al. [130] employs the Markov Chain Monte Carlo (MCMC) method to determine the structure parameters for a HHMM in an unsupervised manner. This

approach is used to discover patterns in video, namely play and break, and is based on a bespoke procedure to select features from the video. Youngblood and Cook [136] examine the problem of automatic learning of a human inhabitant behavioral model. They extract sequential patterns from inhabitant's activities using the Episode Discovery (ED) sequence mining algorithm. A HHMM is created using low-level state information and high-level sequential patterns. This is used to learn an action policy for the environment. This model, as with [130], was created to represent specific features inherent in the data.

Using the hierarchical clustering algorithm from Chapter 4 as a basis, we now aim to construct a two-level HHMM which shows different parts of the initial data as separate submodels. The top HHMM level corresponds to the data patterns, and at the bottom we have the initial data sequence divided into subsequences according to the patterns they represent.

The choice of the number of levels is natural: every data sequence has patterns. It is difficult to say which patterns are contained in data with an unknown structure, and how we can define them. Even manually, for a known structure, this can be done in several ways for the same data, depending on the application. We can construct HHMM using more dendrogram levels, but this can be unnecessary in most applications and it also increases the computation time.

In a HHMM, every higher-level state corresponds to a stream produced by lower-level sub-states, a transition at the top level is invoked only when the lower-level HMM enters an exit state. Therefore it is natural to construct a Hierarchical Hidden Markov Model in a "bottom-up" manner.

After the hierarchical clustering algorithm is applied to the initial data, we get

the data dendrogram representation as in Figure 5.2. We use this representation to construct the HHMM structure. The outline of the proposed HHMM structure construction algorithm is presented in Table 5.1.

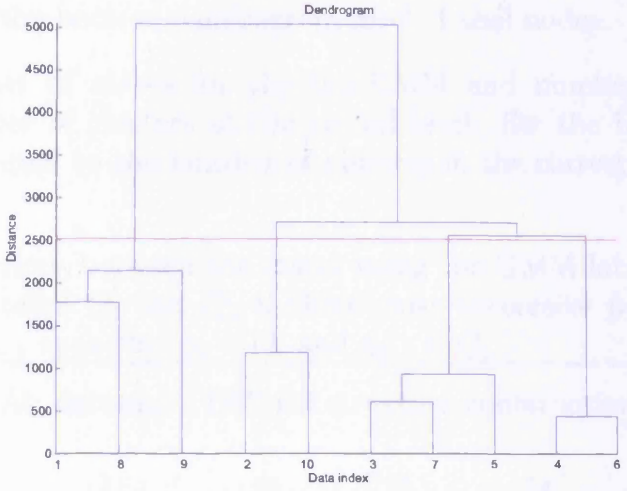

Figure 5.2: A dendrogram for the walking data. The red line indicates the cut-off level. Four clusters are formed in this example

We apply a heuristic approach to the construction of the HHMM structure. We cut the dendrogram using the most popular measure from the clustering theory, mean distance, and use the clustering specified by the dendrogram at that cut-off level. The purpose of this is to provide clusters that are similar enough to be grouped together, and the sizes of these groups are large enough to construct a regular HMM with them. Figure 5.3 illustrates the hierarchy for the motion data, obtained from the dendrogram shown in Figure 5.2. Here the cut-off level clusters are denoted by their own numbers. Using this representation we can set the number of states for each HMM (for this particular example these numbers are 4 for the top-HMM, and 3, 2, 3 and 2 for the bottom level HMMs).

Figure 5.4 shows the resulting HHMM structure, as a state transition diagram.

Input: Data with an unknown structure, hierarchically clustered in a dendrogram using algorithm 4.1.
Output: HHMM parameter ζ.

1. Find the dendrogram cut-off level using the mean distance between clusters. Remove all dendrogram levels above the cut-off and intermediate levels between this level and the bottom dendrogram level. Label nodes.
2. Set the number of states for the top-HMM and number of bottom-HMMs equal to number of clusters at the cut-off level. Set the bottom-HMMs number of states equal to the number of clusters in the corresponding dendrogram branches.
3. Identify transitions between the states using the GMM labels. Set a transition between the states Q_i and Q_j if there exist successive points from the data sequence y_t, y_{t+1} such that $y_t \in Q_i$ and $y_{t+1} \in Q_j$.

Table 5.1: An automatic HHMM structure construction algorithm

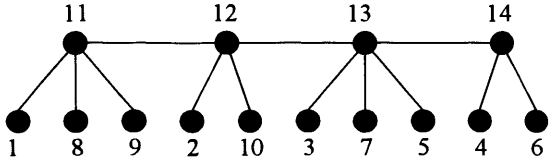

Figure 5.3: A hierarchical representation for the motion data

Black arrows denote state transitions, and dotted arrows denote returns to the parents states. Our HHMM does not have a fully connected topology because we limited the transitions by using the GMM labels obtained at the first stage. This reduces the number of non-zero transition matrix elements and simplifies calculation of the HHMM parameters.

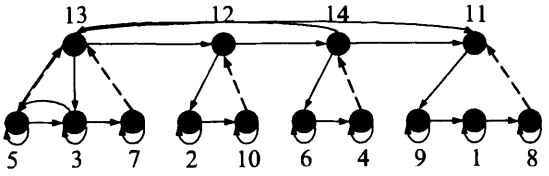

Figure 5.4: A HHMM state transition diagram for the motion data

5.4 Representing the HHMM as a DBN

A HHMM is a special case of Dynamic Bayesian Networks (DBN) [92]. It is common to convert an HHMM to a DBN, because the time complexity of computation can be greatly reduced.

We represent the obtained HHMM as a DBN as shown in Figure 5.5. Our HHMM has two state levels, and the production states are at the bottom of the hierarchy. The state of the HHMM at level d and time t we represent as Q_t^d and observed state at time t as O_t. The state of the whole HHMM we encode by the vector $\vec{Q}_t = (Q_t^1, Q_t^2)$. This vector represents the contents of the stack, which specifies the full path from the root to the leaf state. F_t^d is an indicator which is "on" if the HMM at level d and time t has just finished (i.e. is about to enter an end state), otherwise it is "off".

There are two transition types in a DBN: intra and inter. States within one time step are connected by intra transitions ($Q_t^d \rightarrow Q_t^{d+1}$, $Q_t^d \rightarrow O_t$ $Q_t^d \rightarrow F_t^d$, $F_t^{d+1} \rightarrow F_t^d$), between the time steps they are connected by inter transitions ($Q_t^d \rightarrow Q_{t+1}^d$ and $F_t^d \rightarrow Q_{t+1}^d$). The vertical arrow between Q variables represents a transition from a state to its sub-state. The arrows between the F variables enforce the fact that a higher level HMM can only change state when the lower-level HMM is finished.

We now define the Conditional Probability Distribution (CPD) for each of the node types from parameters of the original HHMM, which will complete the definition of the model. We review the first, middle and last time slices, as well as the top and bottom layers of the hierarchical structure separately, since they have different topology.

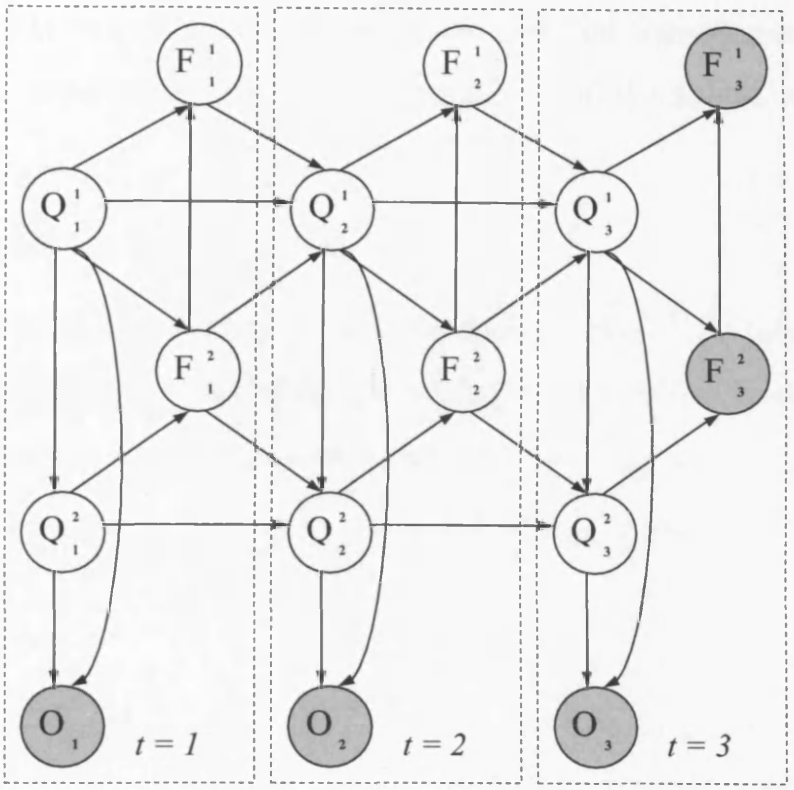

Figure 5.5: An HHMM represented as a DBN. Q_t^d is the state at time t, level d; $F_t^d = 1$ if the HHM at level d has finished (entered its exit state), otherwise $F_t^d = 0$. Shaded nodes are observed; the remaining nodes are hidden.

5.4.1 Definition of the CPDs

- **Bottom level:** $d = 2, t = 2...T-1$.

  Define CPDs for Q_t^2 and F_t^2. The equation for F_t^2 is

$$P(F_t^2 = 1 | Q_t^2 = i, Q_t^1 = k) = A_k^2(i, end) \tag{5.4.1}$$

  where $A_k^2(i, end)$ is the termination probability for sub-model of the original HHMM. We assumed $i, j \neq end$. Definition of Q_t^2 CPD is

$$P(Q_t^2 = j | Q_{t-1}^2 = i, F_{t-1}^2 = f, Q_t^1 = k) = \begin{cases} \widetilde{A}_k^2(i,j), f = 0 \\ \pi_k^2(j), f = 1 \end{cases} \tag{5.4.2}$$

where $\widetilde{A}_k^2$ is rescaled version of the original HHMM transition matrix A_k^2, and π_k^2 is the initial distribution for level 2 given that the parent variables are in state k.

- **Top level:** $d = 1, t = 2...T - 1$.

  The difference here is that we now also get a signal from below, F_t^2, which specifies whether the sub-model has finished or not. If it is finished, we change state; otherwise we remain in the same state. The equations for these cases are

$$P(F_t^1 = 1 | Q_t^1 = i, F_t^2 = f) = \begin{cases} A_1^1(i, end), f = 1 \\ 0, f = 0 \end{cases} \tag{5.4.3}$$

$$P(Q_t^1 = j | Q_{t-1}^1 = i, F_{t-1}^2 = b, F_{t-1}^1 = f) = \begin{cases} \delta(i, j), b = 0 \\ \widetilde{A}_1^1(i, j), b = 1, f = 0 \\ \pi_1^1(j), b = 1, f = 1 \end{cases} \tag{5.4.4}$$

- **Initial slice:** $t = 1, d = 1, 2$.

  The CPDs for the nodes in the first slice are: $P(Q_1^1 = j) = \pi^1(j)$ for the top level and $P(Q_1^2 = j | Q_1^1 = k) = \pi_k^2(j)$ for the bottom level.

- **Final slice:** $t = T, d = 1, 2$.

  To ensure that all sub HMMs reached their end states by the end of the sequence, we set $F_T^d = 1$ for the all d.

5.4.2 DBN: Inference and Learning

To estimate the DBN parameters, we use the maximum-likelihood (ML) criterion. Let us denote the DBN parameters using $\tilde{\lambda}$ by analogy with λ from Equation 5.2.8. Then the goal of learning is to compute

$$\tilde{\lambda}^* = \arg\max_{\tilde{\lambda}} P(O|\tilde{\lambda}) = \arg\max_{\tilde{\lambda}} \log P(O|\tilde{\lambda}) \tag{5.4.5}$$

where $\log P(O|\tilde{\lambda})$ is the log-likelihood of the training data. We use the Expectation-Maximisation (EM) technique to learn the parameters. Note that EM uses inference as a subroutine, and hence efficient inference is essential for efficient learning.

There are several kinds of inference in DBNs, which include filtering, prediction, and smoothing. Filtering is used to estimate the current state based on all observations to date. Prediction is computing the posterior distribution over future states given all observations. Smoothing is computing a posterior distribution over a past state given all observations. We use the junction tree algorithm to perform inference.

The junction tree algorithm works by performing a forwards-backwards sweep through a chain of junction trees (jtrees). Each jtree is formed from a "1.5 slice DBN", which is a DBN that contains all the nodes in slice 1 but only inter nodes from slice 2. We perform inference in each tree separately, and then pass parameters between them via the inter nodes, first forwards and then backwards, using the frontier algorithm, see [94] for the full details.

5.5 Dynamic Framework Construction Summary

5.5.1 Dynamic Framework: Training Process

The goal of the HHMM training process is to form a model that is capable of exploiting hierarchical structure (in contrast to HMMs) and to estimate parameters from Equation 5.2.9 for that model. The steps involved are shown in Table 5.2.

1. Model Initialisation: already pre-processed training data (e.g. some motion features such as the angular coordinates).
2. Perform Hierarchical Clustering as described in the algorithm from Table 4.1, Chapter 4.
3. Using the obtained result, find HHMM structure by applying the algorithm shown in Table 5.1.
4. Convert HHMM into DBN and find parameter λ from Equation 5.2.9.

Table 5.2: Dynamic framework: training process

5.5.2 Dynamic Framework: Testing Processes

Testing the HHMM includes assigning labels to the new unseen data (a classification task) and sampling from the model.

To perform classification of new data, we follow steps from Table 5.3.

1. Embed the test data into the Isomap space using the kernel function given in Chapter 3.
2. Given HHMM structure ζ, find new parameter λ' from Equation 5.2.9.
3. Using λ', allocate the new data to HHMM states (i.e. label them).

Table 5.3: Dynamic framework: testing processes

To verify how well the obtained HHMM describes the data, we generate random samples using the HHMM parameters. Markov Chain Monte Carlo (MCMC) methods [2] produce a stream of samples from the posterior distribution of the hidden variables given the observations. To generate samples from the model, we apply the Gibbs sampler [46], most common case of MCMC methods, which can be thought of as a stochastic version of EM [94]. We reconstruct the original data features for these

samples from the Isomap space using GRBF to compare them with the training sequence.

We give a general conclusion and discuss the results in the following Chapter, where we demonstrate the applications of our dynamic model.

Chapter 6

Dynamic Framework: Applications

6.1 Introduction

In this Chapter we demonstrate that the automatic dynamic modelling algorithm described in the previous Chapter is able to perform with various data features obtained from a variety of video sources. We evaluate how the proposed algorithm processes different types of motion data, represented by 3D coordinates, angular coordinates and silhouettes extracted from video sequences. The data we use here contains a wide range of variation in both upper and lower body parts, and sequences from several hundreds to several thousands of frames. We are not linked to the fact that this data is human motion and are not using this information during the model's construction. First we extract semantic patterns and evaluate the model for a variety of actions: the swordplay data (Section 6.2.1), the soccer ball kicking data (Section 6.2.2), the gymnastic exercises data (Section 6.2.3), and dance data (Section 6.2.4). These examples are taken from the CMU motion capture database [31] where motion is represented by angular coordinates. In Section 6.3 we address the dynamic model construction and its classification ability with 3D coordinates of a walking person. And finally, a model is constructed with silhouette sequences from the IXMAS motion database

[32]. To evaluate the obtained models, we perform data classification and synthesis experiments. In each experiment we compare our classification results to the results achieved by applying an unsupervised HHMM and a standard HMM.

6.2 CMU MoCap data

We have tested our algorithm on a number of motion capture sequences from the CMU motion capture database. The algorithm input data here is represented by 32 angular coordinates (the video preprocessing and feature extraction details can be found on the CMU website [31]). The motion sequences we used comprise of 573-5357 frames and show swordplay, soccer ball kicking, gymnastic exercises and dancing sequence examples. Each example is described in more details below.

6.2.1 Swordplay Data

The swordplay data sequence contains 1500 frames, and consists of two stabbing movement from the standing position, another two stabs from the half bent knees position, and finally the sword lowering movement.

We reduce the dimensionality of the original space to 3 using our automatic method described in Section 3.5, and perform clustering. Figure 6.1 shows the obtained clustering in the embedded space.

In our experiments we assume that data has smooth transformations between points in the embedded space and therefore clusters include chains of neighbouring elements from the original data sequence, i.e. data subsequences. Also note that in the Isomap space similar clusters are in close proximity to each other - we use this property in constructing our model. Figure 6.1 clearly illustrates our idea: the

embedded data forms a trajectory according to the original sequence. There are two loops in the Figure, patterns "3-7" and "4-10-2-6", which represent stabs from the straight position and stabs from half bend the knees position respectively. The first stabbing is represented mainly by two clusters: the stub and the sweep. The second one is more complicated because of the knees bending movement and greater stub amplitude, and presented by double the number of states. Clusters "8" and "9" illustrate the start of the first stabbing motion and the end of the last stabbing motion, and clusters "1" and "5" represent the final motion, the sword lowering.

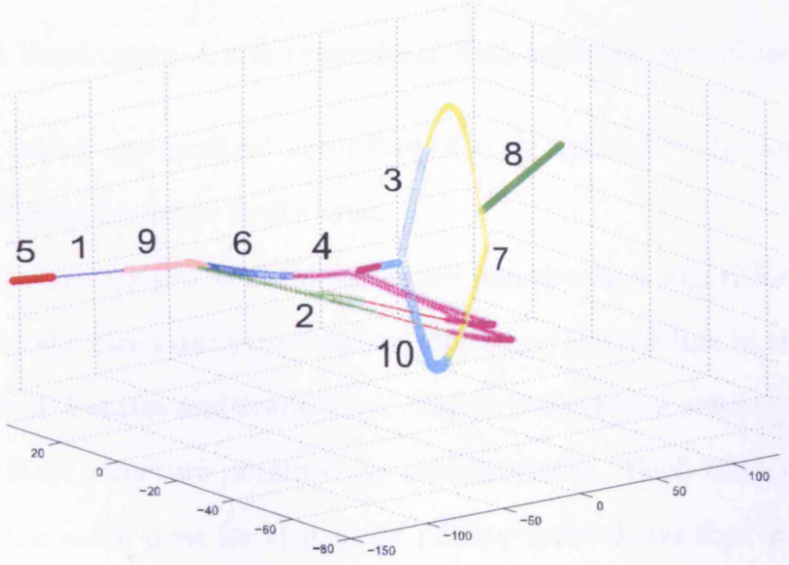

Figure 6.1: GMM clustering of the swordplay data in the Isomap space

After applying the agglomerative clustering algorithm (Section 4.5) to the obtained cluster means we get the dendrogram shown in Figure 6.2. The visual representation of this dendrogram is shown in Figure 6.3. Each image represents the mean pose of the cluster. There are ten levels in the dendrogram, with GMM cluster means

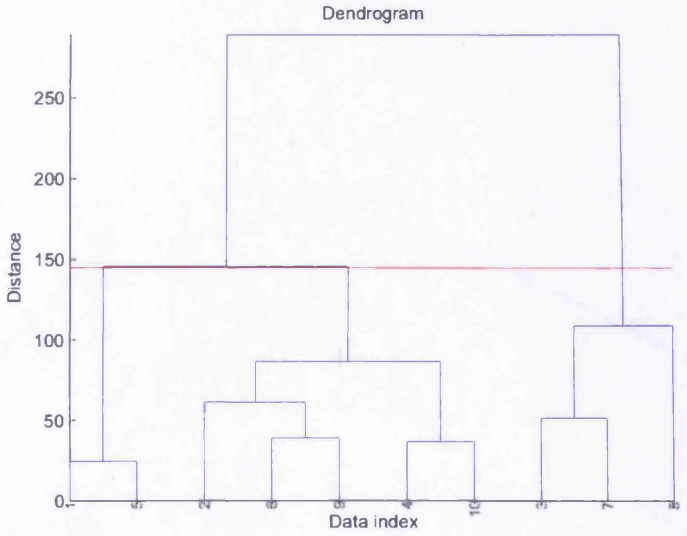

Figure 6.2: A dendrogram for the swordplay data with the cut-off level (red line)

at the bottom which are grouped according of their similarity. It can be seen that there are three main branches in the tree.

We apply our automatic HHMM structure construction algorithm described in Section 5.3, find the mean distance measure (shown as the red line in the Figure 6.2), and set the HHMM states and transitions. Figure 6.4 shows a schematic representation of the HHMM structure obtained by our algorithm. Each hidden state here is illustrated by the mean pose for that state. This Figure shows that we were able to extract the above mentioned three main patterns from the unlabelled data. We get different numbers of states for each of these patterns: the first stabbing is represented by three states, the second stabbing consists of five states as it is a more complicated movement, and the sword lowering pattern is represented by just two states.

First of all we verify that our model is reasonable: we draw samples from the obtained model using Gibbs' sampling technique [46]. Figure 6.5 demonstrates the

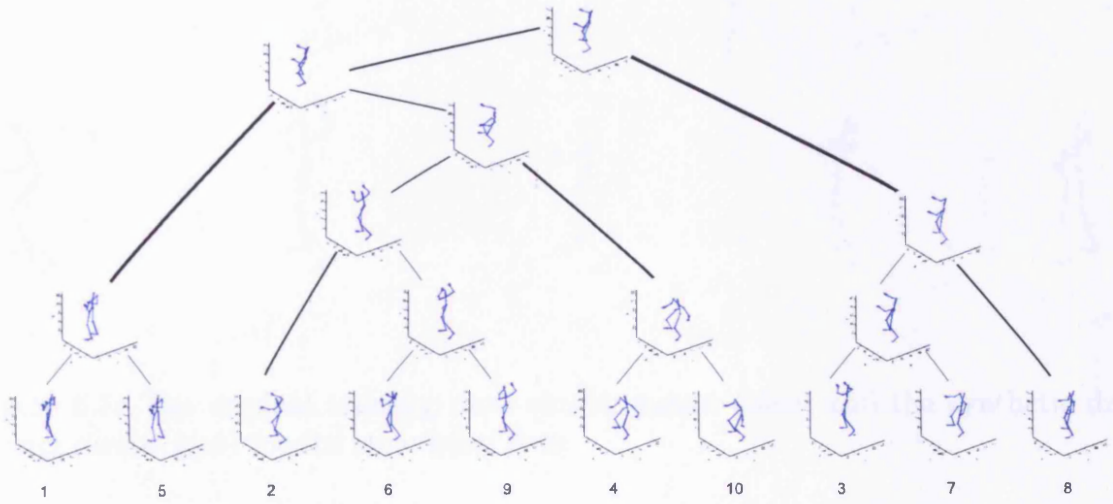

Figure 6.3: Visual dendrogram representation for the swordplay data

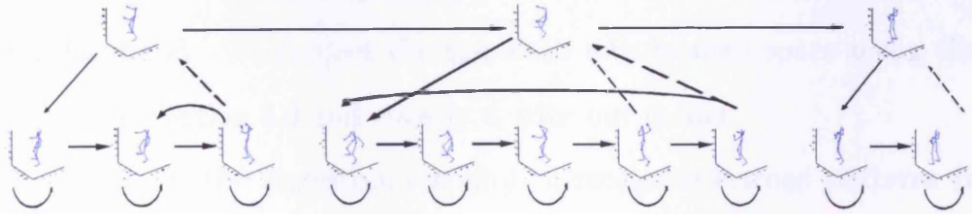

Figure 6.4: A HHMM state transition diagram for the swordplay data

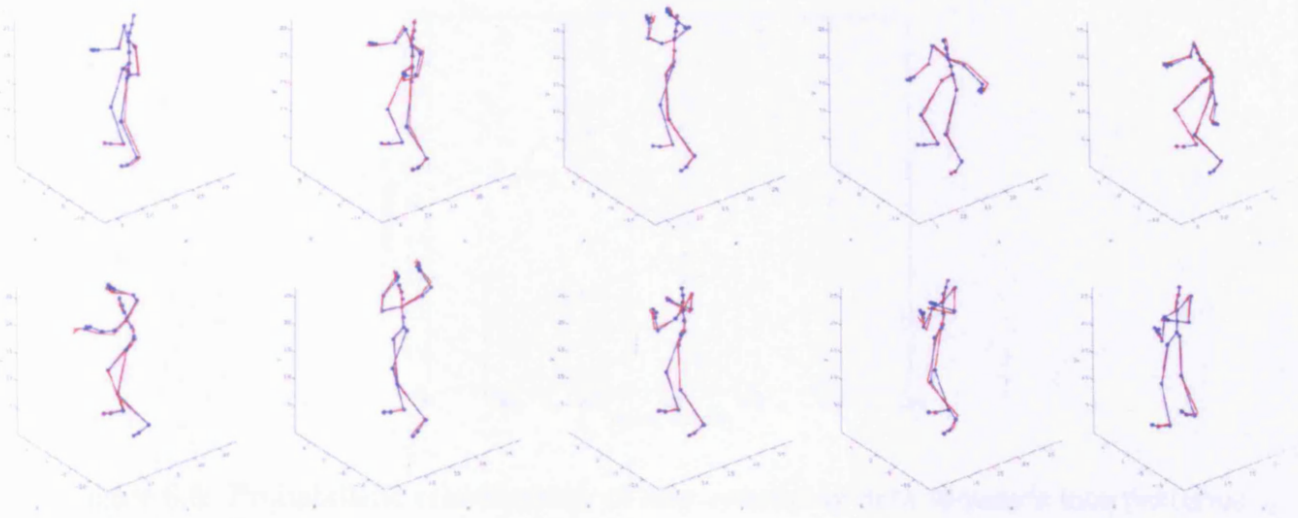

Figure 6.5: The original training data cluster means (blue) and the synthetic data cluster means (red) for the swordplay data

original and the reconstructed mean poses for each cluster which are very close to each other. This confirms that we get good results with the sampling experiment. To get back projection from the Isomap embedding to the original space we used the GRBF algorithm presented in Section 3.4.

To further test the model we perform data classification experiments. We take another unused fragment of swordplay motion, which includes similar data, but with more motion variations such as putting the sword in the other hand, and stabbing with the other hand. We project the test data into Isomap space using the kernel function given in Section 3.3 and classify it with our model.

Figure 6.6 shows the algorithm's ability to recognise learned patterns from the new data containing unknown variations (i.e. model's observation probability). The Figure indicates that most probably at the beginning of the motion there are stabbing movements from the standing position (blue line). In the second part of the test sequence the highest probability is for stabbing from half bend knees position (green

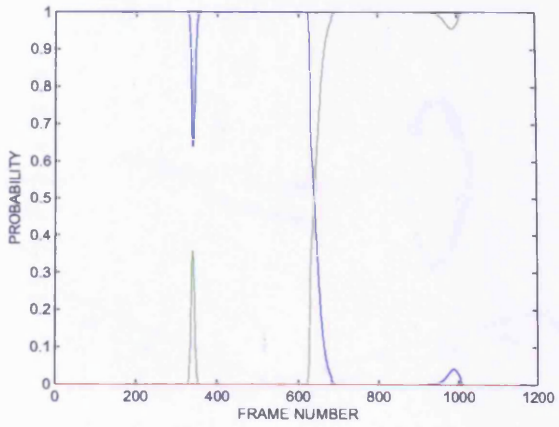

Figure 6.6: Probabilistic classification of new swordplay data sequence into patterns: stabs from the straight position (blue line), stabs from the half bent knees position (green line), sword lowering (red line)

line). The regions where the probabilities get closer indicate the stab's start and finish.

The pattern with the maximum probability is used to label the data. To verify the result and make a comparison with other methods, we manually label train data with these patterns. We consider this labelling as a ground truth, and perform classification evaluation with our semi-supervised model, an unsupervised HHMM and a flat HMM over the test data. We use the same structure for the unsupervised HHMM obtained from our hierarchical algorithm, and the HMM based on the bottom level of our model. The classification results are shown in the first column of Table 6.1. Using our semi-supervised algorithm, we get better accuracy than the other methods.

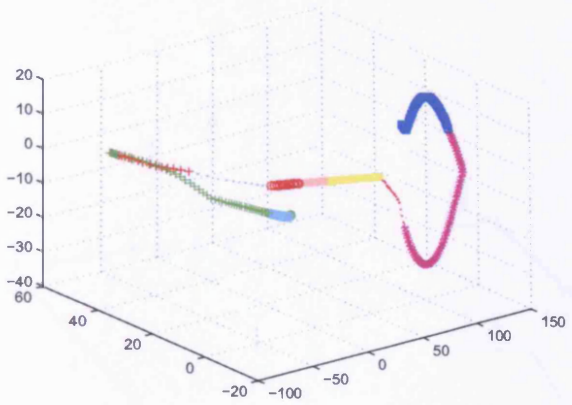

Figure 6.7: GMM clustering of the soccer data in the Isomap space

6.2.2 Soccer Ball Kicking Data

For this experiment we use a 590-frame motion sequence which consists of two steps, ball kicking, turn and step after the turn. We project the data into the three dimensional Isomap space (see Figure 6.7), and perform clustering using the automatic method from Chapter 4. Figure 6.8 demonstrates the visual dendrogram representation obtained from this method.

We construct a dynamic model with our algorithm, and find that we are able to extract three main patterns: two first steps, a kick, and movement after the kick. Figure 6.9 shows the resulting HHMM structure for the soccer data, where the HMM of states "1-3-8-2-11-12" represents the first pattern (steps), the HMM of states "5-9" represents kicking and the HMM of states "7-10-4-6" relates to the movement after kicking.

Now we recover the original data from the obtained model in the same way as for the previous example. Figure 6.10 illustrates the original and the reconstructed cluster means. For this data, the main data variations correspond to the leg movement, and

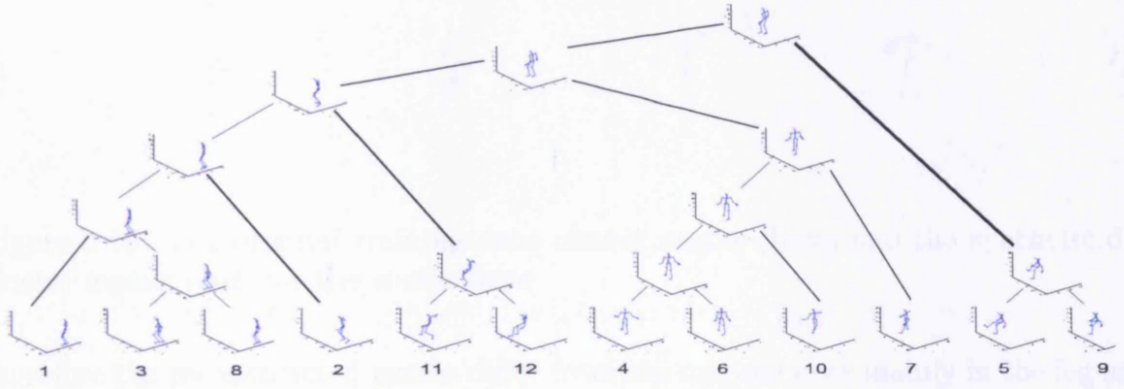

Figure 6.8: Visual dendrogram representation for the soccer data

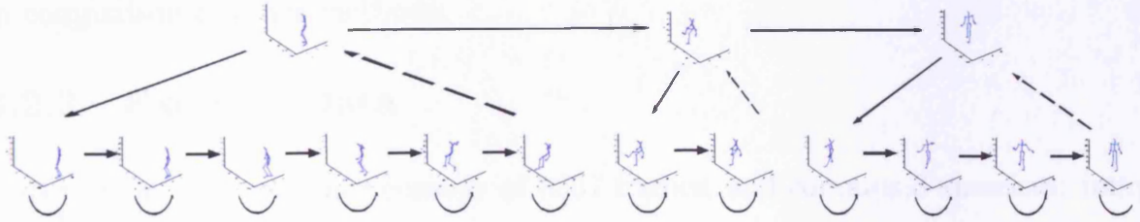

Figure 6.9: A HHMM state transition diagram for the soccer data

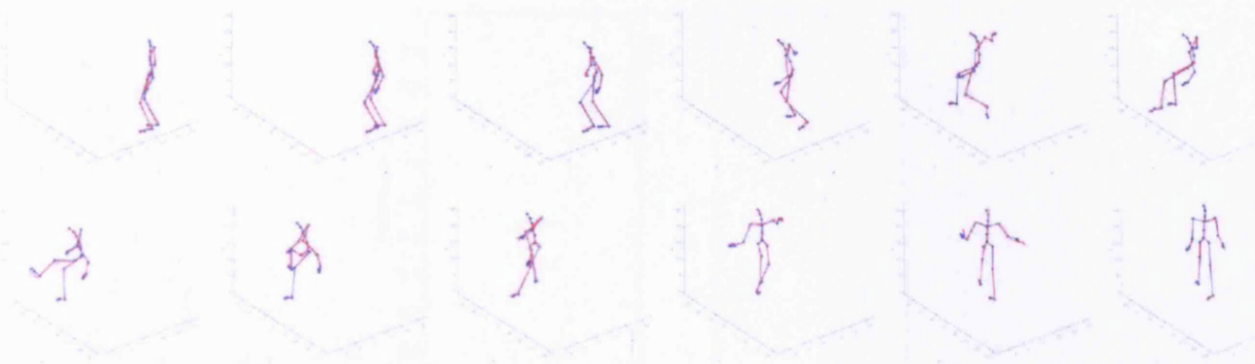

Figure 6.10: The original training data cluster means (blue) and the synthetic data cluster means (red) for the soccer data

therefore the reconstructed means differ from the original ones mainly in the leg area. But still, the original and the reconstructed means look similar.

To test the effectiveness of our model we perform classification of new soccer data with similar patterns. Figure 6.11 demonstrates our algorithm's ability to identify the mentioned patterns from the new data. First there are steps in the test data (blue line), ball kicking (red line), and turning (green line) shown in the Figure as regions with the highest probability. The classification results are presented in the second column of Table 6.1. Again, our method produces the best recognition rate in comparison to other methods.

6.2.3 Exercise Data

This motion data sequence consists of 5357 frames, and contains 8 exercises: jumps, jogging, squats, side twists, lateral bending, side stretch, forward-backward stretch and forward stretch. In this example we demonstrate that our method is able to work with a variety of movements, as well as with long data sequences.

The size of this data set is quite large to use with the original Isomap, therefore

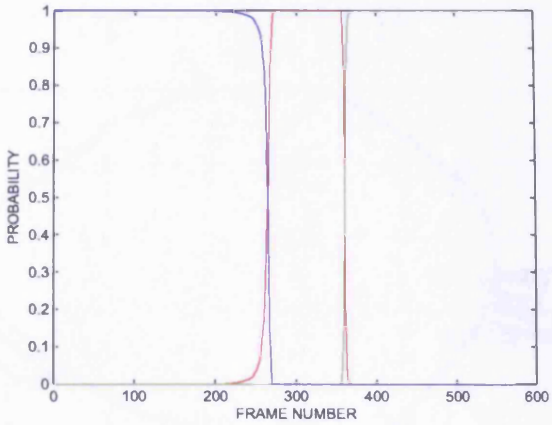

Figure 6.11: Probabilistic classification of the new soccer data sequence into patterns: steps (blue line), kicking (red line), turning (green line)

here we apply landmark Isomap instead as we pointed out in Section 3.3. We take every fifth frame of the data sequence as the Isomap landmark, automatically choose landmark Isomap's parameter by performing our method described in Section 3.5, and get a two-dimensional projection of the original data into the Isomap space. Figures 6.12 and 6.13 show the obtained circle-shaped projection. The blobs at the circle represent the exercise repetition. In Figure 6.12 we manually labelled the projection sections according to a consistent exercise number to explain the embedding. There are jump exercise first (red line), the second exercise is jogging (green line), the next one is squats (yellow line), and so on as above. The horizontal changes in poses are from constriction to side stretch, and vertical changes are from squat to up stretch poses, see Figure 6.13.

After the clustering algorithm is applied to the obtained Isomap coordinates we get 26 gaussian clusters shown in Figure 6.14. Figure 6.15 demonstrates the same embedded space, with the clusters mean poses. The dendrogram with the cut-off level for the HHMM structure construction is presented in Figure 6.16.

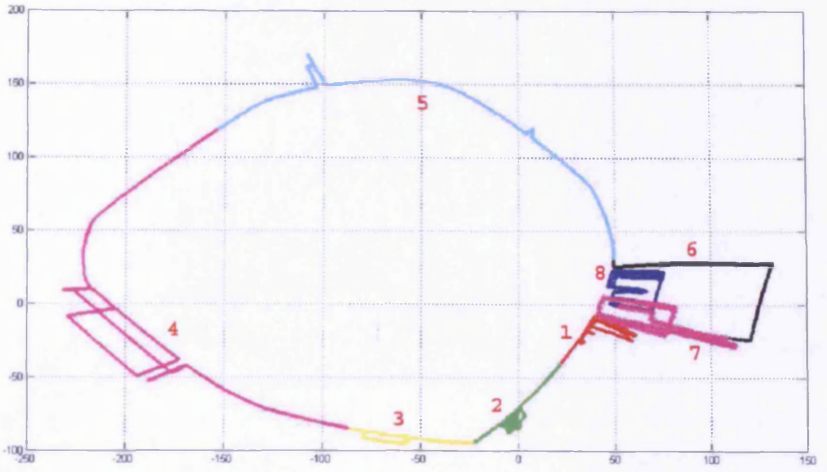

Figure 6.12: Exercise data projection into Isomap space

The resulting HHMM structure obtained by our algorithm is shown in Figure 6.17. There are six states in the top level HMM, which represent jumps, jogging, squats, side twists, lateral bending and stretches (states 27, 28, 29, 30, 31, 32 respectively). At the bottom level we get six HMM's with numbers of states ranging from two (squats HMM: down and up states) to seven (lateral bending). The stretches sub-HMM consists of six states and represents three exercises together: side, forward-backward and forward stretches.

To verify the model we synthesize the original data sequence from the constructed model. Figure 6.18 demonstrates the original and the reconstructed cluster means similarity. Since here we have lots of different motions and the data variations are spread across all the data, we see that the obtained reconstructed means are more different from the original ones in comparison to the previous two examples.

To further test the obtained model we take a new exercise sequence which consists of 2300 frames and includes jumps, jogging, squats and side twists exercises. We

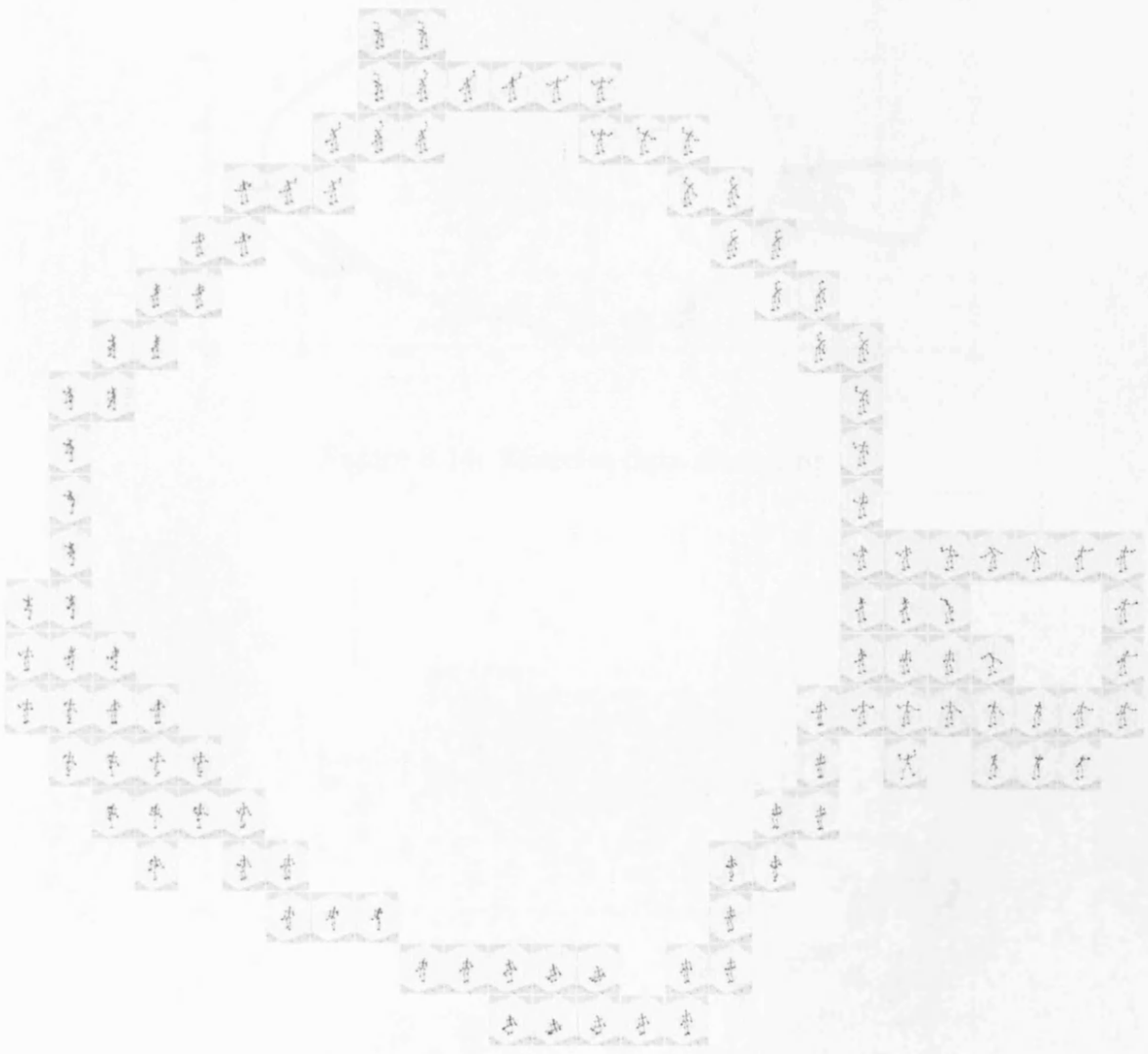

Figure 6.13: Exercise data projection into Isomap space, visual representation

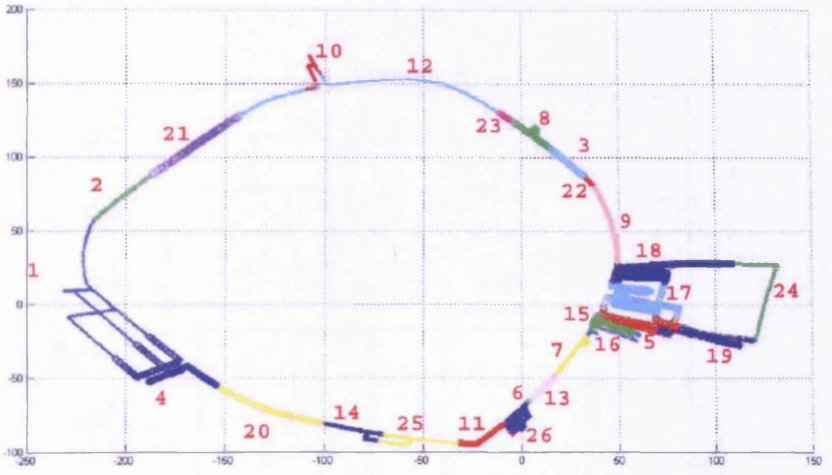

Figure 6.14: Exercise data clustering

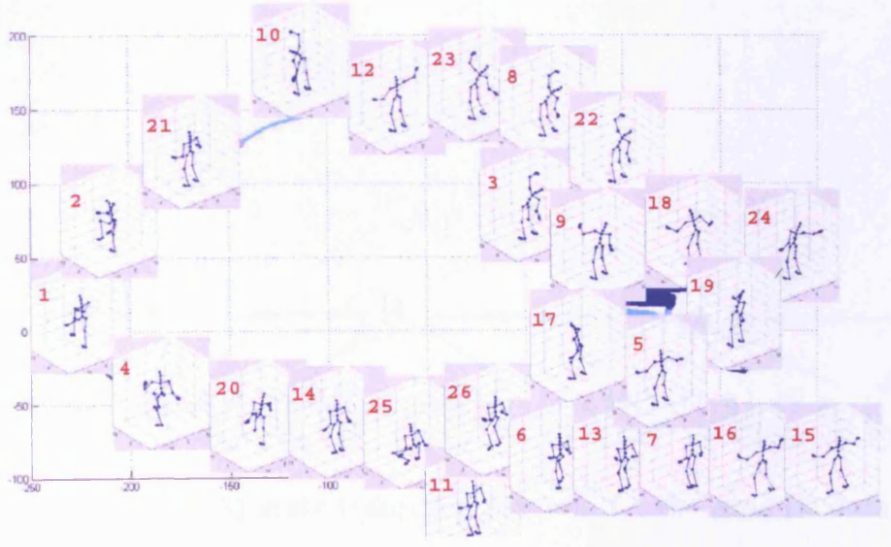

Figure 6.15: Exercise data, cluster mean poses

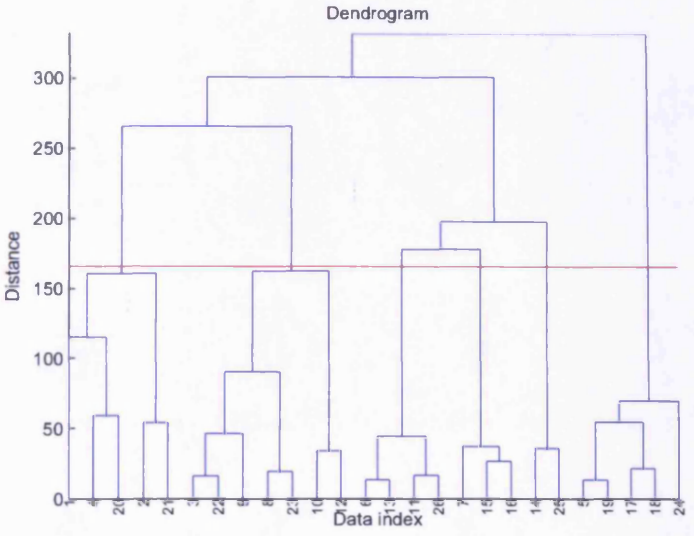

Figure 6.16: Exercise data: dendrogram with the cut-off level

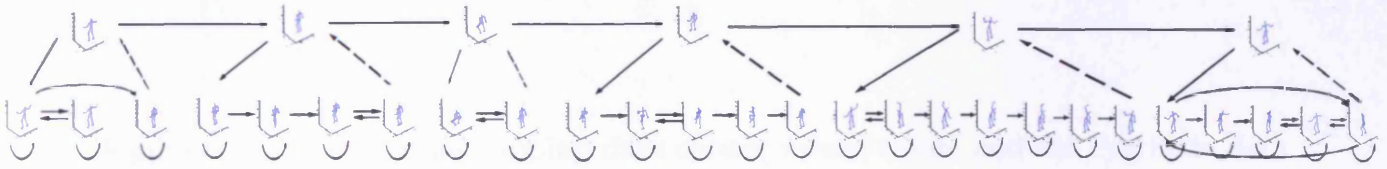

Figure 6.17: A HHMM state transition diagram for the exercise data

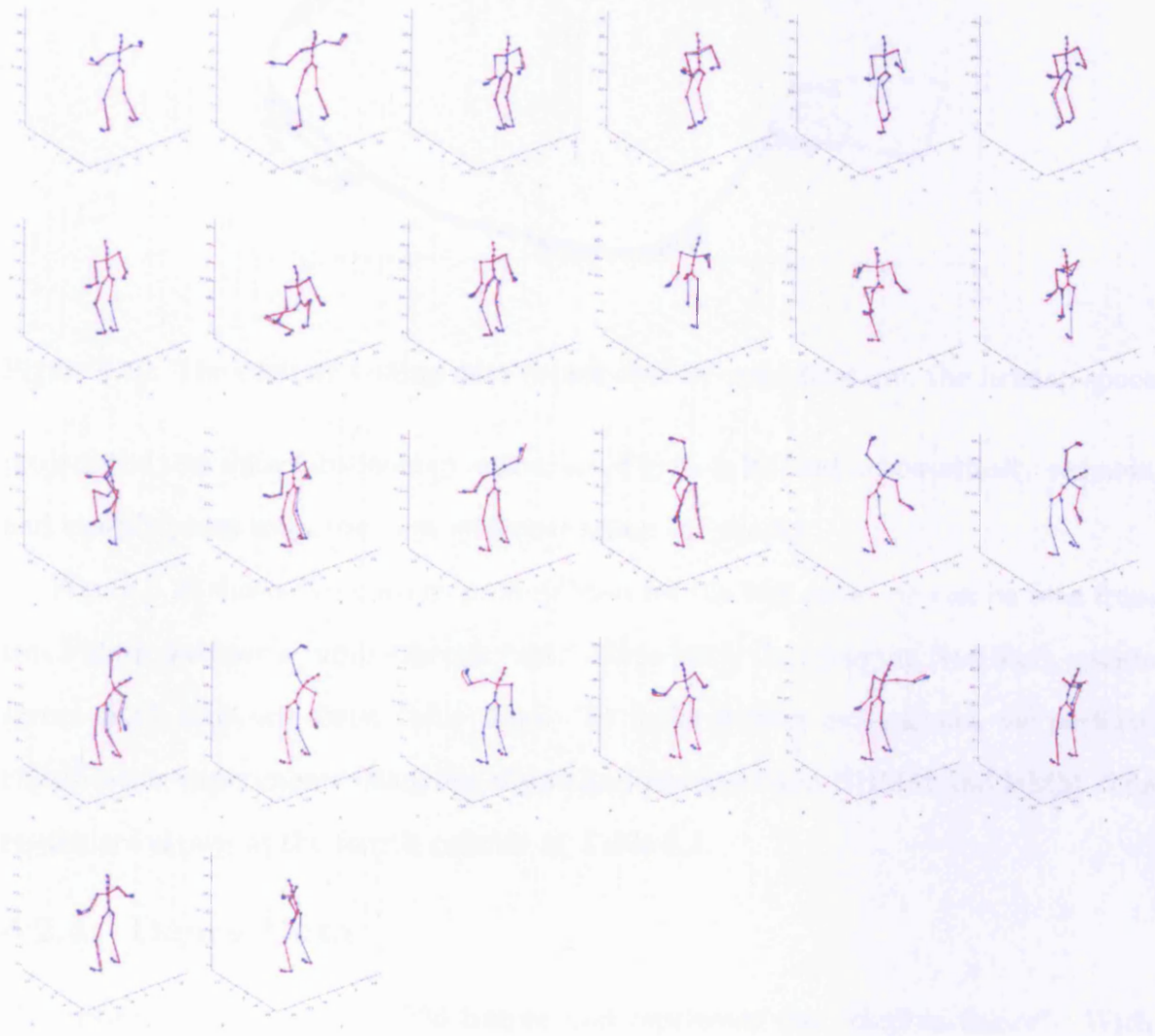

Figure 6.18: The original training data cluster means (blue) and the synthetic data cluster means (red) for the exercise data

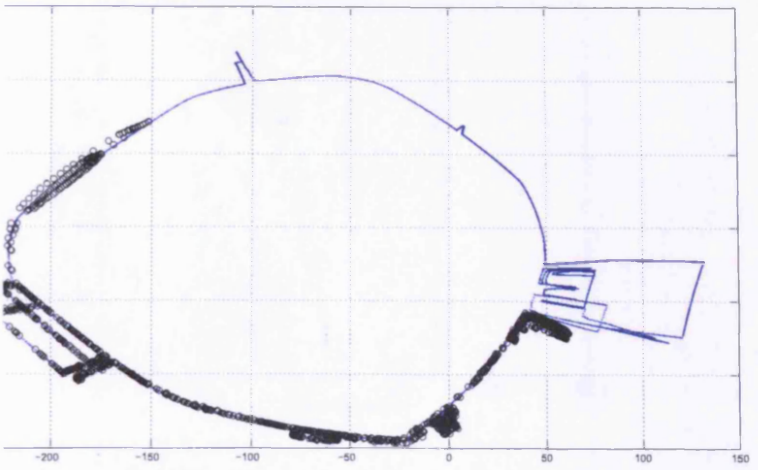

Figure 6.19: The exercise testing data (black circles) embedded into the Isomap space

project the test data into Isomap space, see Figure 6.19, and automatically segment and classify poses from this new sequence using the model.

Figure 6.20 shows the pattern probabilities for the test data. As can be seen from this Figure, we have a jump exercise first (yellow line), then jogging (red line), squats (green line), and side twist (blue line). To make further evaluations, we perform classification experiments using our algorithm, unsupervised HHMM and HMM. The results are shown in the fourth column of Table 6.1.

6.2.4 Dance Data

The training data contains 1536 frames and represents the "chicken dance". With this example we show the algorithm's ability to work with periodic data. Figure 6.21 illustrates dendrogram constructed for the dance data by our hierarchical clustering algorithm. We construct the dynamic model of the dance data by taking this dendrogram as a base. We allow all transitions at the top level HMM, see Figure 6.22.

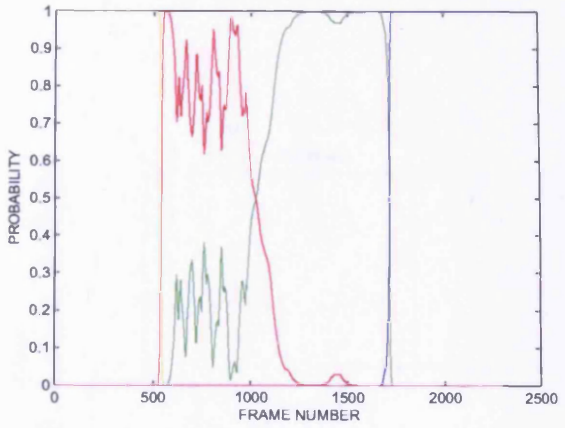

Figure 6.20: Probabilistic classification of new exercise data sequence into patterns: jumping (yellow line), jogging (red line), squats (green line), side twist (blue line), lateral bending (cyan line), stretches (magenta line)

There are two patterns (two HMMs at the bottom level) for this data: one which deals with the legs movement with five states, and another one which deals with the hands.

To verify the model we synthesize the original data sequence and perform the classification experiments. Figure 6.23 shows that the difference between the original and synthesised data is small and we can say that the data generated by our dynamic model is close to the original data. Figure 6.24 demonstrates the pattern probabilities for the new data. This Figure shows oscillation of probability for the test subsequence where the person is standing up and performing the hand movement at the same time, but the general classification result is sufficient. The classification results for the test data using labelling obtained from the training data are shown in the third column of Table 6.1.

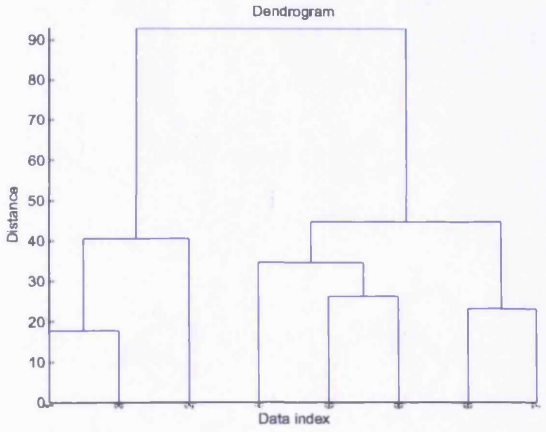

Figure 6.21: Dendrogram for the dance data

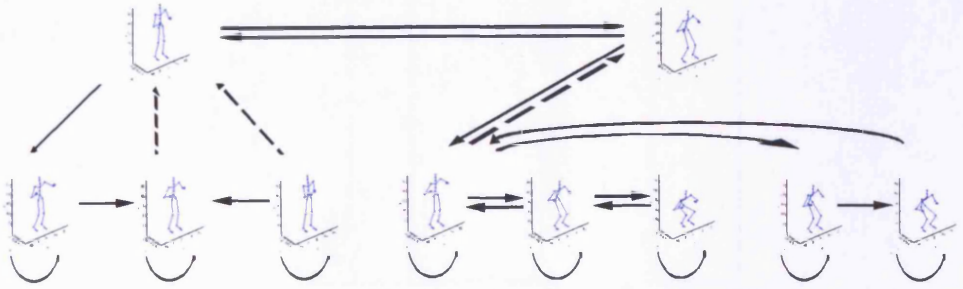

Figure 6.22: An HHMM state transition diagram for the dance data

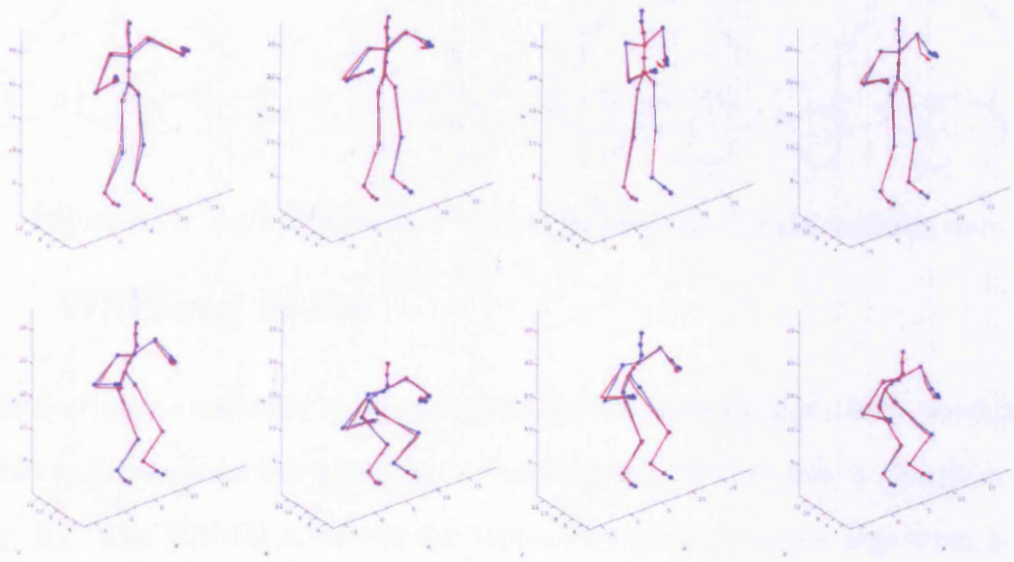

Figure 6.23: The original training data cluster means (blue) and the synthetic data cluster means (red) for the dance data

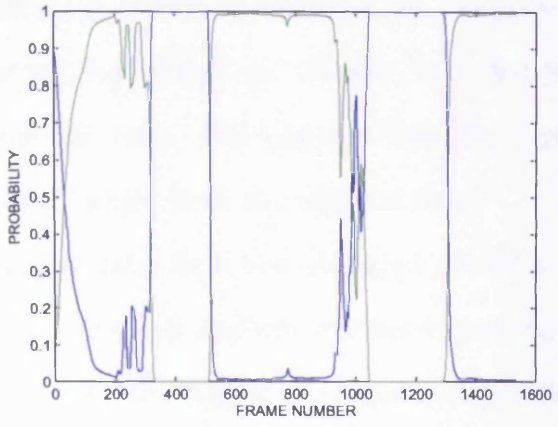

Figure 6.24: Probabilistic classification of new dance data sequence into patterns: hand movements (green line), leg movements (blue line)

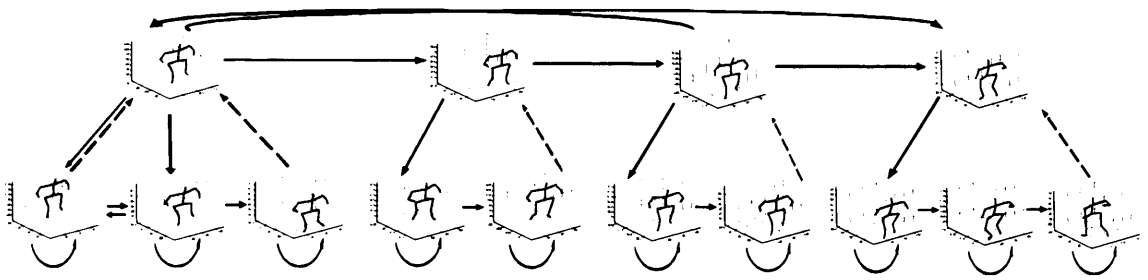

Figure 6.25: An HHMM state transition diagram for the walking data

6.3 Walking Data

In this Section we continue to consider the motion data example from Section 4.7.2. The hierarchical model visualisation, a dendrogram, for this data is demonstrated in Figure 5.2. The HHMM structure for this data obtained by our algorithm is shown in Figure 5.4 and Figure 6.25. We get one top level HMM, which includes four sub-HMMs. the HMM of states "5-3-7" corresponds to the pattern which represents the motion beginning and start of the turn. The HMM of states "2-10" corresponds to the sequence pattern with the left leg moving down and the right leg lifting up. The "6-4"-state HMM movement is the opposite of the previous pattern: the right leg moving down and the left leg lifting up. Finally, the "9-1-8" HMM represents the turn and the step after the turn. We can say that the automatically constructed HHMM extracts "natural" units from the original data.

We recover the original data from the obtained model to verify the result. Figure 6.26 demonstrates the original and the reconstructed cluster means. The reconstructed mean poses here correctly repeat the original ones with some small variations.

In order to test the performance of the model, we automatically segment and classify poses from another walking data sequence of 131 frames, consisting of two steps. We project the new data into Isomap space using the kernel function given in

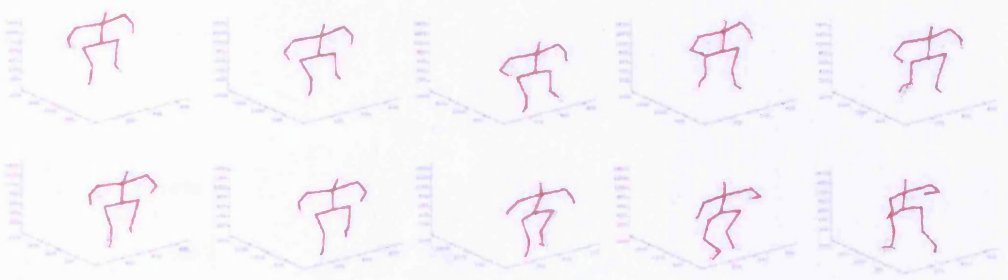

Figure 6.26: The original training data cluster means (black) and the synthetic data cluster means (red) for the walking data

Section 3.3. The result is shown in Figure 6.27. The new data projections are close to the original ones in the embedded space. As expected, it is concentrated at the bottom and the middle of the Isomap space, which corresponds to the first two steps in the original data.

Figure 6.28 shows the HHMM's ability to automatically recognise motion segments (find semantic patterns) "5-3-7", "2-10", "6-4" and "9-1-8" from the new data. Here the horizontal axis corresponds to the frame number, and the vertical axis corresponds to the probability distribution of each pattern. There is a high probability for the "5-3-7" pattern at the beginning of the new data sequence (blue line), then we have the "2-10" pattern in the data (red line), followed by the "6-4" pattern (green line), and probably some bits from the "5-3-7" pattern again. The probability for "9-1-8" pattern is small everywhere (yellow line), thus we consider that there is no such pattern (step after turn) in the test data. The comparison of the classification results is shown in the fifth column of Table 6.1. We can say that our semi-supervised HHMM is able to correctly identify the semantic patterns in the test data and classify new motion patterns better than other methods.

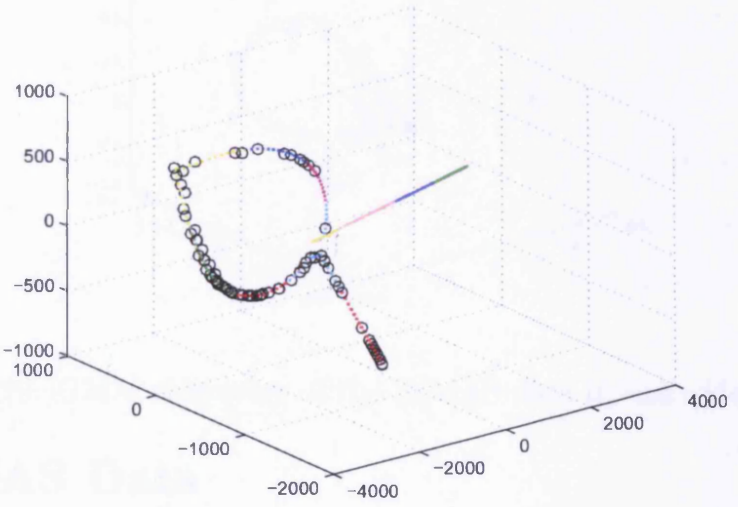

Figure 6.27: New walking data projection in the embedded space (black circles)

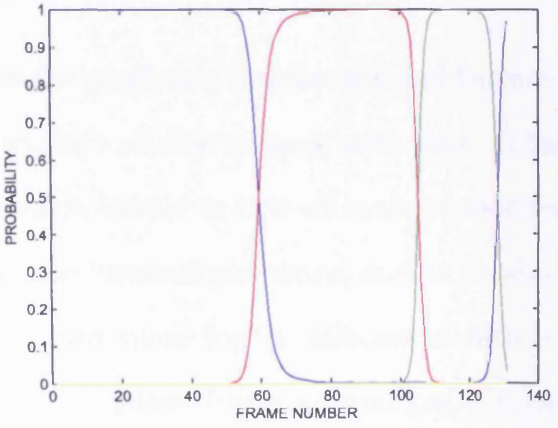

Figure 6.28: Probabilistic classification of new walking data sequence into patterns: "5-3-7" (blue line), "2-10" (red line), "6-4" (green line), "9-1-8" (yellow line)

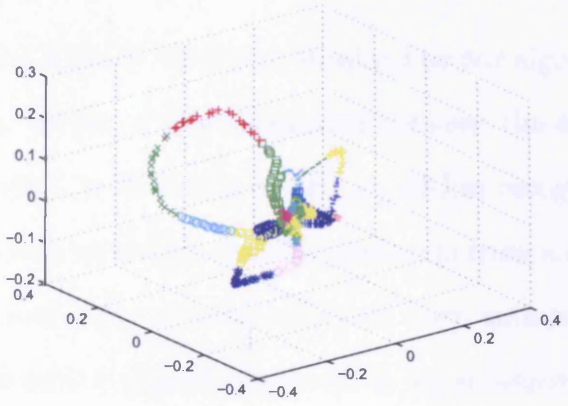

Figure 6.29: GMM clustering of the IXMAS data in embedded space

6.4 IXMAS Data

The Inria Xmas Motion Acquisition (IXMAS) sequence we used here contains 11 actions: check watch, cross arms, scratch head, yoga, turn around, walk, wave, punch, kick, point and throw away. The silhouettes were extracted from the video using a standard background subtraction technique, modelling each pixel as a Gaussian in RGB space.

Although raw silhouette pixels are not the optimal feature for motion analysis, we use it to show the algorithm's ability to work with such difficult input data. For the Isomap algorithm, it is much harder to extract main motion features from silhouettes, to embed them into the low dimensional space, and to produce useful trajectories for further analysis. The Isomap space for the silhouette data is more knotted than for the coordinate data, which makes clustering and pattern extraction more difficult. Figure 6.29 illustrates the embedded space for this example.

To test the algorithm we take the sequence of 1076 frames from the IXMAS database. It shows the front view silhouettes of the person performing the above

actions in the above order.

Figure 6.30 shows the HHMM structure obtained by our algorithm. In comparison to the coordinate data, we get more transitions between the states (because of the space knotting). As shown in this Figure, the algorithm recognised five patterns in this data. The first pattern represents hand movements from a standing position and includes check watch, cross arms, scratch head and wave actions. The second pattern is the yoga action. The next recognised pattern is leg movements and includes turn around, walk and kick actions. The fourth pattern is hand movements with legs apart, this contains punch and point actions. And the last pattern is the throw away action.

We synthesize the original data from the obtained model to verify the result. Figure 6.31 shows the original and reconstructed clusters mean poses. Although the error here is larger than for the previous examples, because of the data's spatial properties, we can clearly see the similarity of the mean poses.

We use another sequence of similar length to test the model. The classification results are shown in Figure 6.32. There are lots of transitions between the first pattern (magenta line), third pattern (yellow line) and the fourth pattern (red line). The second (blue line) and the last (green line) patterns are each represented by single occurrences. Comparison of the classification accuracy is presented at the last column of Table 6.1. The accuracy is lower compared with earlier experiments because of the increased complexity of its trajectory in the Isomap space, but it can be seen that our algorithm is able to find the meaningful patterns even from such data.

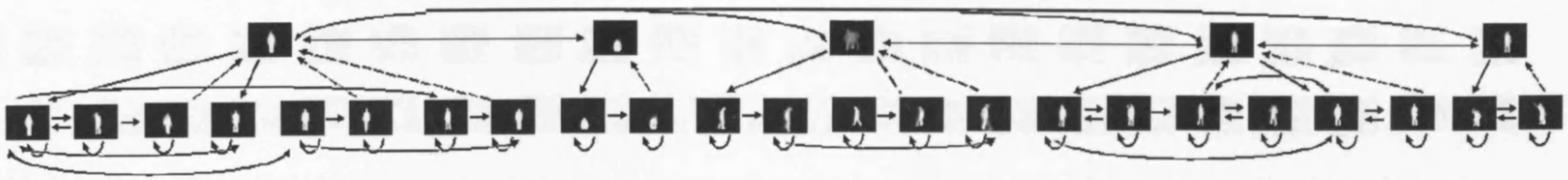

Figure 6.30: An HHMM state transition diagram for the IXMAS data

Figure 6.31: The original training data cluster means (top) and the synthetic data cluster means (bottom) for the IXMAS action data

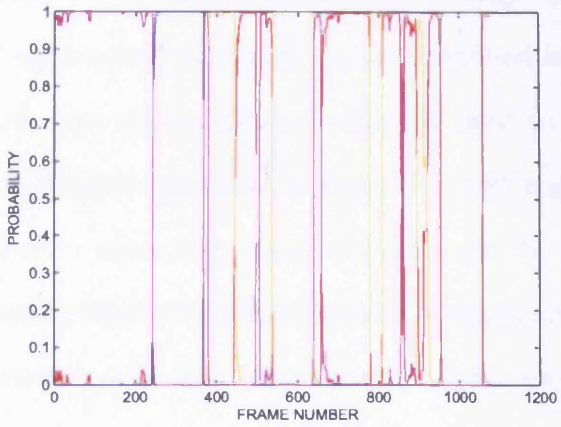

Figure 6.32: Probabilistic classification of new IXMAS data sequence into patterns

Method	**Sword**	**Soccer**	**Exercise**	**Dance**	**Walk**	**IXMAS**
Semi-supervised HHMM	89.85%	96.34%	93.57%	91.86%	96.95%	71.63%
Unsupervised HHMM	83.69%	93.19%	91.13%	89.97%	93.13%	65.43%
Flat HMM	85.78%	90.4%	90.65%	88.28%	92.37%	65.43%

Table 6.1: Classification results (rate of correct classification)

6.5 Summary

We have presented a novel method for automatic dynamic framework construction. First we build a hierarchy for the initial data using Isomap, Gaussian Mixture Modelling and hierarchical agglomerative clustering as described in Chapter 4. From this we obtain a hierarchical data representation which is used to construct the HHMM. Experimental results on different motion features (3D and angular pose coordinates, silhouettes extracted from a video sequence) demonstrate the approach is effective at automatically constructing efficient HHMMs with a structure that naturally represents the underlying motion and that allows for accurate modelling of the data for applications such as tracking and motion resynthesis.

We now summarise our main contributions:

- We have presented a new dynamic framework method which is useful for the extraction of semantic patterns from high dimensional data.
- Since the algorithm is not linked with any a priori information from the data, it can be used with various data types (for example, in DNA sequence analysis).
- Our algorithm is fully automated with no additional configuration parameters required.
- We have shown that patterns extracted by our algorithm have a semantic meaning.
- In pattern classifications experiments our method performs better than other methods.

In order to develop a more detailed knowledge of the strengths and robustness of our algorithm, a more thorough experimental evaluation of our system will be carried out in future work. We plan to make the algorithm more effective against noise by modifying the Isomap algorithm, see Section 8.2.1. Also we hope to improve the robustness and compactness of our model by using alternative clustering algorithms, and we plan to involve probability estimation in the cut-off level detection.

Chapter 7

Automatic Part-Based Data Decomposition

7.1 Introduction

In the previous Chapter we presented promising results of applying a novel dynamic model described in Chapter 5, to real world data. This model is general and does not require any a priori knowledge of the input data. But it is well known that to get maximal accuracy in data modelling we can use an algorithm which is specifically created to analyse a particular data set, i.e. an algorithm based on a priori data knowledge. In this case we can keep as many data features as we want for an application, even if they are not generally significant. For example, in the talking head application, the hierarchy developed there may utilise sets of features in a variety of combinations, depending on the user's purposes. The top level of the hierarchy may seek to capture the main modes of variation of the complete data. Other levels may be used to model specific relationships between certain features, such as specific interactions of speech with facial regions (e.g. lower face, lips, eyebrows). Such a model has proven to be robust in tracking facial features and also resynthesising new

video-realistic faces [28].

The principal difficulty in creating such models is in determining which parts should be used, and identifying examples of these parts in the training data. The task of finding patterns embedded in data is a popular research field in computer science [13, 59, 83].

Nonnegative matrix factorization (NMF) [80] is a promising tool in learning the parts of objects and images. NMF imposes non-negativity constraints in its bases and coefficients. These constraints lead to a parts based representation because they allow only additive, not subtractive, combinations. Later in [62] Hoyer presented Sparse non-negative matrix factorisation (sparse NMF) with an adjustable sparseness parameter. This allows the discovery of parts-based representations that are qualitatively better than those given by the basic NMF. Because of its parts-based representation property, NMF and its variations have been used in image classification [16, 50, 51, 53], face expression recognition [17], face detection [21], face and object recognition [85, 86, 105].

In all of the above papers the number of data parts was quite large and was chosen manually. Here we propose using intrinsic dimensionality estimation (see Section 2.3 for more details) to find the correct number of parts. Thus we catch the most important intrinsic features in order to model them further as data parts in our method.

The novelty of the proposed method is that we use NMF for automatic data mask construction to separate the data into the most significant parts. Our method is fast, efficient, automatic and does not require any additional parameters. We do not use any a priori information about the data. We consider the construction of our model in

Sections 7.2, 7.3, 7.4 and demonstrate the effectiveness of our algorithm by applying it to different data types: talking head data, emotional head data and articulated human motion data in Section 7.5. Finally, the conclusions are given in Section 7.6.

7.2 Data Preprocessing and Parameter Setting

Initial data for our partial data representation algorithm can be represented by raw data vectors (images as an example), or by a parameterised data model. The output generated is a mask identifying different data parts: the data elements labelled with part numbers.

This algorithm works best with aligned data obtained from a sequence of observations. As we explored in our experiments, the method is not very suitable for separation of images of highly articulated objects or objects viewed from significantly different viewpoints into parts because such issues can badly distort the resulting mask.

A normalisation step is needed to make the patterns of interest more evident. Data normalisation is provided as a preprocessing step before NMF, in the same manner as in Li et al. [83]. Basically, we perform a standard data normalisation procedure with additional constraints to avoid having any exact zeros and excessively large data values.

At the first step of our algorithm we estimate the number of data parts. We choose this number to be the same as the intrinsic dimensionality of the data manifold. We use the k-NN (k nearest neighbours) method described in [29] to estimate the intrinsic dimensionality. In this method the dimension is estimated from the length of the minimal spanning tree on the geodesic NN distances computed by the Isomap

algorithm [118]. To automate the k-NN method we choose the number of nearest neighbours using our algorithm described in Section 3.5.

7.3 Constructing Modified Sparse NMF

Classical NMF is a method to obtain a representation of data using non-negativity constraints. These constraints lead to a part-based representation because they only allow additive, not subtractive, combinations of the original data [80]. Given initial data expressed by an $n \times m$ matrix X, where each column is an n-dimensional non-negative vector of the original data (m vectors), it is possible to find two new matrices (W and H) in order to approximate the original matrix:

$$X_{ij} \approx (WH)_{ij} = \Sigma_{l=1}^{r} W_{il} H_{lj} \tag{7.3.1}$$

The dimensions of the factorised matrices W and H are $n \times r$ and $r \times m$ respectively. Each column of W contains a basis vector while each column of H contains the weight needed to approximate the corresponding column in X using the bases from W.

Given a data matrix X, the optimal choice of matrices W and H is defined to be those nonnegative matrices that minimise the reconstruction error between X and WH. Various error functions have been proposed [81], the most widely used one is the squared error (Euclidean distance) function

$$E(W, H) = \|X - WH\|^2 = \Sigma_{ij}(X_{ij} - (WH)_{ij})^2 \tag{7.3.2}$$

However, the additive parts learned by NMF are not necessarily localised, as was pointed out by Li et al. in [83]. To obtain a meaningful partial representation we want to restrict the energy of each NMF basis to the most significant components

only. Therefore we use sparse NMF [62] which proves to be more appropriate in part-based object decomposition than original NMF.

In sparse NMF the objective (7.3.2) is minimised under the constraints that all columns of W and rows of H have common sparseness σ_W and σ_H respectively. The sparseness $\sigma(x)$ is defined by the relation between the Euclidean norm $\|.\|_2$ and 1-norm $\|x\|_1 := \Sigma_i \mid x_i \mid$ as follows

$$\sigma(x) := \frac{\sqrt{n} - \frac{\|x\|_1}{\|x\|_2}}{\sqrt{n} - 1} \tag{7.3.3}$$

if $x \in R^n \setminus 0$. Since $\frac{1}{n}\|x\|_1 \leq \|x\|_2 \leq \|x\|_1$ equation (7.3.3) is bounded $0 \leq \sigma(x) \leq 1$. In particular, $\sigma(x) = 0$ for minimal sparse vectors with equal non-zero components, and $\sigma(x) = 1$ for maximally sparse vectors with all but one vanishing components.

7.3.1 Sparse NMF Modification: Random Acol initialisation

It is well known that good initialisation can improve the speed and the accuracy of the solutions of many NMF algorithms [128]. Langville et al. [77] proposed random Acol initialisation as an inexpensive and effective initialisation technique. Random Acol forms an initialisation of each column of the basis matrix W by averaging p random columns of X. We use the random Acol technique for our modified sparse NMF instead of a random initialisation.

So far, we have three unspecified parameters in our method: initialisation parameter p and sparseness parameters σ_W and σ_H. To automate the algorithm we set $p = [\frac{m}{r}]$. We learn useful features from basis W and leave the sparseness of H unconstrained. For all our experiments we set σ_W to 0.78 for simplicity. For more accurate estimation of the sparseness one can use the method described in [58].

Figure 7.1: Features learned from the ORL database using sparse NMF

7.3.2 Sparse NMF Modification: Earth Mover's Distance.

Figure 7.1 shows the example of sparse NMF basis from the Hoyer paper [62]. It can be seen that there are significant similarities among the learned bases. Guillamet and Vitria proposed that the Earth mover's distance (EMD) is better suited to this problem because one can explicitly define a distance which will depend on the basis correlation [52].

EMD can be stated as follows: let I be a set of suppliers, J a set of consumers and d_{ij} the cost to ship a unit of supply from $i \in I$ to $j \in J$. We define d_{ij} as the Euclidean distance. We want to find a set of f_{ij} that minimises the overall cost:

$$dist(x, y) = \min \Sigma_{i \in I} \Sigma_{j \in J} d_{ij} f_{ij} \tag{7.3.4}$$

subject the following constraints:

$$f_{ij} \geq 0, x_i \geq 0, y_j \geq 0, i \in I, j \in J$$

$$\sum_{i \in I} f_{ij} \leq y_j, j \in J$$

$$\sum_{j \in J} f_{ij} \leq x_i, i \in I$$

$$\Sigma_{i \in I} \Sigma_{j \in J} d_{ij} f_{ij} = min(\Sigma_{i \in I} x_i, \Sigma_{j \in J} y_j)$$

where x_i is the total supply of supplier i and y_j is the total capacity of consumer j.

We use EMD as the distance metric instead the Euclidean distance to avoid similarities between bases.

7.4 Data Postprocessing: Mask Construction

After getting the modified sparse NMF basis we need to analyse the results. At this step we produce a mask of the data by construction of boundaries between the basis vectors.

We consider the mask construction for the images as it demonstrates the result most clearly. However, our algorithm can be used on a wide range of data. In the next section we describe postprocessing example for 3D human motion data.

Examples of the modified sparse NMF basis are shown in Figures 7.3 and 7.6. Each of the basis vectors represents a part of the original image. There is substantial noise in each vector, and some vectors contain several separated parts.

To attempt to remedy this we introduce the following procedure. First we consider each vector of the basis separately to reduce the noise. We define a vector element as noise if a 7×7 pixel square centered at this element has any other pixels with zero values. After deleting such components we label each nonzero basis vector component according to its vector number and merge the vectors.

Next we use a region growing technique which is a basic yet effective method. Region growing [1] is a technique which begins with a seed location and attempts to merge neighboring pixels until no more pixels can be added to it. Because of basis sparseness, we have a considerable amount of pixels that were not assigned a label and now need one to be allocated. These errors have to be removed in a second postprocessing step. The most dominant regions, i.e. the regions with largest component values, are selected as seed regions for a region growing process. Region growing is implemented as a morphological operation. A 3×3 square is moved over the merged basis. When a neighbor to the point of interest (the center of the square) has a label assigned, the point of interest is checked for compatibility to that region. In the case where it is found to be compatible (i.e. all point neighbors belong to the same basis label), it is assigned the label of the corresponding region. If there are conflicting regions, i.e. there are different regions adjacent to the point of interest, the largest region is preferred. This is also the case if the center pixel is already labeled.

When this process is completed, every pixel is assigned one of the possible basis labels, this completes our data partitioning algorithm.

7.5 Experimental Results

In this section we evaluate how the proposed algorithm processes several real world data sets with different characteristics. The first data considered in Section 7.5.1 is the talking head. In this example we show how our algorithm works with large images where data variation is concentrated mainly on a small region (mouth). Next we consider facial expression data with the data variation across the whole image. Section 7.5.3 describes the model's partitioning ability with 3D coordinates of a walking person.

7.5.1 Talking Head Data

The algorithm described in the previous section was tested on the data from [28]. Initially, it was the video of a speaker reading a text, recorded at 25fps. The subject was recorded front-on with as little out of plane head movement as possible.

We extract the texture from each frame of the video as described in [28]. Figure 7.2 shows examples of the texture. We perform data normalisation [83] to improve the algorithm's convergence and to make the patterns of interest more evident. The intrinsic dimensionality of the data is chosen by the automated k-NN method is eight. Setting the number of basis vectors to 8, we perform the second step of our algorithm. The result of this step is shown in Figure 7.3. One can see parts of the face there: eyes, cheeks, chin.

We use the postprocessing algorithm described in Section 7.4 for automatic basis analysis. Figure 7.4 shows the mask generated by our algorithm. It can be seen that the automatically constructed partitioning extracts the most important features of the face. We have an eyes region, three mouth regions (upper lip, lower lip, inside

Figure 7.2: Talking head data: examples

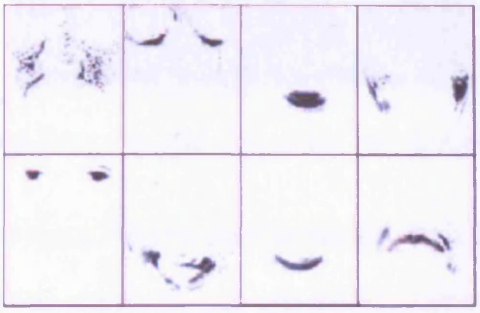

Figure 7.3: Talking head data: modified NMF basis

part of the mouth), cheeks, chin, cheek bones and eyebrow regions. Such partitioning could be very useful for further data analysis offering us a trade-off between keeping fine detail in the data and the large data dimensionality.

7.5.2 Facial Expression Data

For our next experiment we use data sets from two different people. Each person performed several different facial expressions: happiness, sadness and disgust, see Figure 7.5 for the examples.

Figure 7.4: Talking head data: mask

Figure 7.5: Facial expression data one (top) and two (bottom)

Unlike the talking head data, the facial expression data has lots of variation across the whole face. In order to see how our algorithm can deal with expressions, we apply it to each data set. As expected, both sets have the same estimated intrinsic dimensionality, equal to 6. Thus we obtain 6 modified sparse NMF basis vectors which are shown in Figure 7.6. The basis vectors for the facial expression data look similar to the vectors from the previous example. Because the mouth variation is not significant for this example, a vector representing this variation is missed here, while we had 3 mouth variation vectors for the talking head. Instead we have more vectors to represent other face parts which displayed greater variation in these data sets.

To analyse the modified sparse NMF basis we perform data postprocessing, as described in Section 7.4. The results are shown in Figure 7.7. Again, the results are natural and similar to the talking head mask, but with more attention paid to the general face details. For example, we extract a vector which represents a nose. Our algorithm extracts features which are significant for each particular data set.

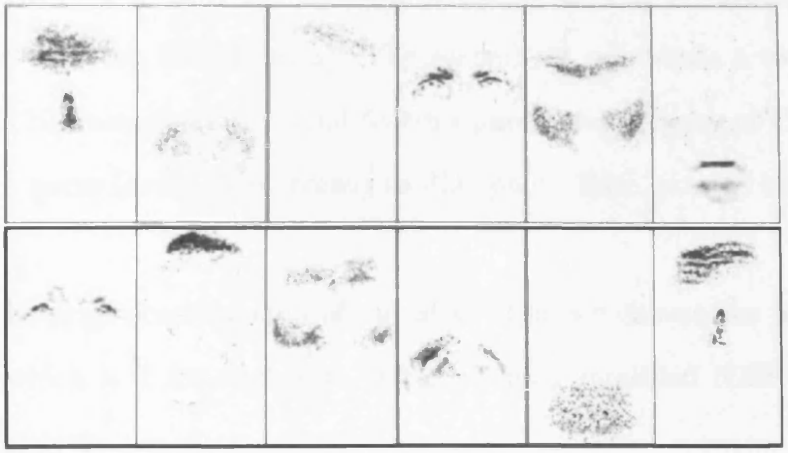

Figure 7.6: Facial expression data one (top) and two (bottom): modified NMF bases

Figure 7.7: Facial expression data one (left) and two (right): masks

7.5.3 Motion Data

Here we test our algorithm with two motion data sets where the data is represented by 3D coordinates. The first set consists of the motion of a walking person: two steps, turn and one step (249 frames). The second set represents a two step motion without turn (179 frames). The initial feature parameters represent the coordinates of human body parts (arms, legs, torso) in 3D space. Each pose is characterised by 17 points.

Following the preprocessing step of our algorithm, we choose the intrinsic dimensionality [29], which is 2 for each set. After running modified NMF we get sparse basis sets for analysis.

We cannot apply the postprocessing step from Section 7.4 because of the data type. Therefore we perform postprocessing in a different way.

We define that a 3D pose junction point belongs to the basis in which this point has maximum summed basis coefficients. Figure 7.8 illustrates the bases for both sets. On the left hand side of Figure 7.8 one can see data partitioning for the walking with turn. The first basis vector here is represented by the torso and forearms, and the second is represented by legs and shoulders. On the right hand side of Figure 7.8 we show data partitioning for the straight walking. Here we have a different partitioning, which consists of two pieces as well. The first basis vector represents a motion of the right arm and the left leg, while the second basis vector represents the left arm, the torso and the right leg. Such partitioning isolates variation in subregions from the rest of the body and provides a high degree of control over different body parts. For example, it is straightforward to find out which part is responsible for moving the legs (the first case), or describes relationships between legs and arms movements (the

second case).

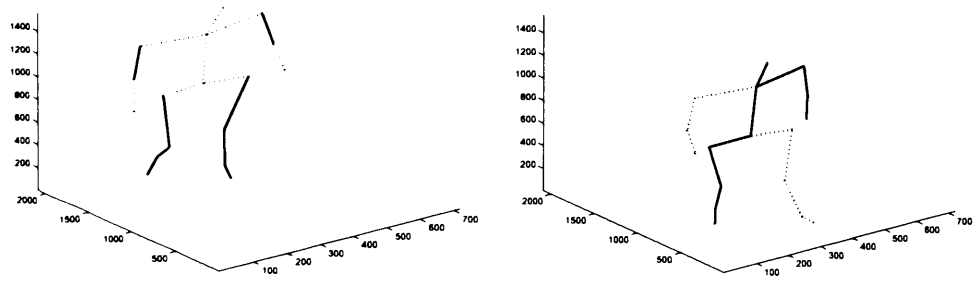

Figure 7.8: 3D motion data partitioning - different bases are represented by different line drawing styles

7.6 Summary

We have described a new algorithm for automatic data decomposition using a modified sparse NMF basis analysis, in which the number of basis vectors is selected to be the same as the estimated intrinsic dimensionality of the data. Segmentation is then performed by applying region growing to the set of basis vectors.

We demonstrate the algorithm's ability to produce good partitioning of real data sets. In these examples we show that our algorithm extracts the most important features for the particular data set. Such partitioning provides a powerful tool for automating construction of parts based data models.

In future work we hope to extend our algorithm to more complex cases, such as highly articulated objects. We also plan to improve the postprocessing step by performing more advanced segmentation.

Chapter 8

Conclusions and Future Work

This thesis investigates low dimensional hierarchical modelling of high dimensional data. Here we give conclusions about our work, along with a discussion of future directions in related research fields.

8.1 Conclusions

The research presented in this thesis was motivated by the desire to develop methods to handle hierarchical modelling of high dimensional data with an unknown structure embedded in a low dimensional subspace. We did not assume any a priori information from the data, so due to the complexity of the task we suggested that the input high dimensional data fully represents its intrinsic topology.

We introduced a number of novel methods, which allow accurate modelling of the high dimensional data. The hierarchical model described here may be used for hierarchical data visualisation and classification, and the dynamic framework allows meaningful pattern extraction and recognition, as well as new data sequence generation. Additionally, we developed a novel data partitioning method to work alongside existing specific models. All of these methods were tested on a variety of data sets.

We began Chapter 3 by describing a non linear low dimensional subspace representation of the high dimensional data. We addressed issues of the optimal input parameter setting for the Isomap algorithm, its back projection and new data sampling in the embedded space.

In Chapter 4 we proposed a hierarchical data modelling method for high dimensional data with no a priori knowledge of the underlying structure of the data. For the data representation we employed the methods presented in the previous Chapter.

In Chapter 5 we developed a dynamic framework for the automatic hierarchy construction algorithm. The method utilised the model presented in Chapter 4 at the first stage and its construction did not involve any information from the input data. At the second stage of the proposed algorithm we introduced dynamics by involving a HHMM which gives us the possibility of extracting semantic patterns (i.e. subsequences) from the data sequence and generating a new data sequence. Experimental results presented in Chapter 6 demonstrate the approach is effective at automatically constructing efficient dynamic models with a structure that naturally represents the underlying data. This allowed for more accurate modelling of the data for applications such as tracking and resynthesis.

In Chapter 7 we presented an additional part to our main methods, where we address the problem of finding patterns embedded in data with an unknown a priori structure. With this representation we can extract data patterns to fit them in the model based methods to make these methods automatic. This algorithm has a broad range of a potential applications, we illustrated this versatility by applying the algorithm to several dissimilar data sets.

In summary, our major contributions are reiterated with respect to their appearance in the main body of the thesis:

- The method for the automatic selection of an optimal parameter value for the Isomap algorithm, presented in Chapter 3, which can be used with a wide class of input data, both real and synthetic. The optimal parameter for modified Isomap and Discriminant Isometric Mapping, can also be chosen using our method, in the same manner as for Isomap. Also we can automate the k-NN method for data intrinsic dimensionality estimation [29] using our method.

- A new method for hierarchy construction from Chapter 4, which can be applied to several corresponding parameter sets of various high dimensional data. This method is fully automated and is easy to implement, there are no additional configuration parameters required. The data representation as a small set of Gaussians means allows efficient use of the clustering algorithm to divide data into groups. In itself the algorithm is useful for different purposes: for data visualisation, as a data classifier and for generation of a new data.

- In Chapter 5 we have presented a new automatic method for extraction of semantic patterns from a data sequence. We have shown that the patterns extracted by our algorithm have a semantic meaning, and in pattern classification experiments our method performs better than other methods. Also we can synthesise a new data sequence using this method, as we have demonstrated in Chapter 6.

- In Chapter 7 we introduced a new automatic method for learning a meaningful sub-part representation from real world data with an unknown a priori structure.

Such partitioning provides a powerful tool for automating construction of parts based data models.

Generally, we consider this research as a first attempt to automatically learn hierarchical dynamic precesses in high dimensional data with an unknown structure through fully automatic HHMM. Rather than using hand crafted approaches to define the hierarchy, we used the data processed through the hierarchical algorithm to set the parameters of the HHMM. We then optimised the model learning routine by introducing DBN with its inference procedures which allowed us to estimate the model parameters more effectively. Additionally, we created the data partial decomposition algorithm, which we plan to incorporate with our main model in future work, see the following Section.

8.2 Future Work

Automatic modelling of data with unknown structure is a challenging task. Whilst this thesis has presented solutions to its important aspects, there are necessarily gaps to be found in the framework that can be addressed in the future. In this section we try to spell out a few extensions that we are interested in pursuing in the near future.

8.2.1 Robust Isomap Algorithm Modification

We constructed our model based on the assumption that the input data is well represented in the high dimensional space. However, in the case of some critical outliers on very noisy data manifolds the results of our algorithm may be unpredictable. In order to overcome this problem, we created a new robust Isomap algorithm modification preliminarily presented in Table 8.1. By applying the network flow we hope to

improve the algorithm's topological stability. At the same time, we improve Isomap's ability to recover the data trajectory by dividing the distances between data points into two categories: local and transitive. This idea was inspired by the ST-Isomap modification reported in [69], the difference is that we introduce a network flow. Thus we try to improve the two above issues with a single algorithm. Also we want to make this algorithm fully automatic and instead of temporal blocks as in [69] we divide the general distance matrix into two: local and transitive. Currently, we choose Perturbed Minimum Spanning Trees (PMSTs) [20] to represent the local distances d_{local}, and work is carried out on choosing robust transitive distance $d_{transitive}$ to make distances between forthcoming clusters more obvious.

Input: N data points in the high-dimensional input space X
Output: Coordinate vectors Y_i in a d-dimensional space Y that best represent the intrinsic geometry of the data.

1. Construct neighborhood graph, G (connect each data point to all points within some fixed radius ϵ, or to all of its K nearest neighbors).
Compute matrix $D_G = d_X(i,j)$

$$d_X(i,j) = \begin{cases} d_{Xlocal}(i,j) & \text{if } d_{Xlocal}(i,j) \leq \tau \\ d_{Xtransitive}(i,j) & \text{otherwise} \end{cases} \tag{8.2.1}$$

$d_{Xlocal}(i,j)$ - Dissimilarity matrix (i.e. $d_{local}(i,j)$ = max weight on the path from x_i to x_j)

threshold $\tau = \sqrt{\chi^2_{N,0.975}}$
2. Network flow

$$I = arg\min \Sigma D_G(x_i, x_j)$$

3. Apply MDS to I, construct d-dimensional coordinate vectors, Y_i.

Table 8.1: Robust Isomap algorithm

8.2.2 Dynamic Modelling of Multi-Source Data

There are many multi-source data applications, like the talking head data with both video and speech, where several data types co-exist in time. In our future work we plan to apply the methods presented in this thesis to dynamic algorithms for dual input data. Cosker in [26] constructed a Dual-Input HMM for talking head data, where he used HMM parameters, obtained after the video data training, to estimate new speech parameters. Since our statistical models are suitable for multi-source data, we can use this idea and apply our algorithms based on HHMM dynamic modelling to estimate a hidden data sequence from a new concurrent observation string.

References

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

[2] C. Andrieu, N. de Freita, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[3] M. Balasubramanian and E.L. Schwartz. The Isomap algorithm and topological stability. *Science*, 295(5552):7-7, 2002.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[5] R.E. Bellman. Dynamic programming. *Princeton University Press*, 1957.

[6] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[7] Y. Bengio, J. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and spectral clustering. *Advances in Neural Information Processing Systems*, 16:177–184, 2004.

[8] M. Bernstein, V.de Silva, J.C. Langford, and J.B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. *Technical Report, Stanford University*, 2000.

[9] J. A. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models. *Technical Report icsi-tr-97-021, UC Berkeley*, 1997.

[10] J. Blackburn and R. E. Human motion recognition using isomap and dynamic time warping. *Human Motion- Understanding, Modeling, Capture and Animation: Lecture Notes in Computer Science*, 4814:285–298, 2007.

[11] D. Blei, T. Gri, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 17–24, 2004.

[12] R. Bowden. *Learning non-linear Models of Shape and Motion.* PhD thesis, Dept Systems Engineering, Brunel University, 2000.

[13] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182, 1999.

[14] M. Brand. Charting a manifold. *Advances in NIPS*, 15:961–968, 2003.

[15] M. Brand. From subspace to submanifold methods. *Proc. of British Mashine Vision Conference (BMVC)*, pages 6–12, 2004.

[16] G. Buchsbaum and O. Bloch. Color categories revealed by non-negative matrix factorization of munsell color spectra. *Vision Research*, 42:559–563, 2002.

[17] I. Buciu and I. Pitas. Application of non-negative and local non-negative matrix factorization to facial expression recognition. In *Proc. of ICPR*, pages 288–291, 2004.

[18] M. Burge and W. Burger. Ear biometrics in computer vision. In *Proc. of ICPR*, pages 822–826, 2000.

[19] F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407, 2002.

[20] M.A. Carreira-Perpinan and R.S. Zemel. Proximity graphs for clustering and manifold learning. *Advances in Neural Information Processing Systems*, 17:225–232, 2005.

[21] X. Chen, L. Gu, S.Z. Li, and H.J. Zhang. Learning representative local features for face detection. In *Proc. of ICPR*, volume 1, pages 1126–1131, 2001.

[22] H. Choi and S. Choi. Kernel Isomap on noisy manifold. In *Proc. of IEEE Int'l Conf. Development and Learning (ICDL)*, pages 208–213, 2005.

[23] R. Cilibrasi, P.M.B. Vitanyi, and R.de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28:49–67, 2004.

[24] M. Cohen, D. Massaro, and R. Clark. Training a talking head. *In Proc. of the IEEE Fourth International Conference on Multimodal Interfaces, (ICMI'02)*, pages 499–504, 2002.

[25] T. F. Cootes, G. Edwards, and C. J. Taylor. Active appearance models. In *Proc. of European Conf. on Computer Vision*, volume 2, pages 484–498, 1998.

[26] D. Cosker. *Animation of a Hierarchical Appearance Based Facial Model and Perceptual Analysis of Visual Speech.* PhD thesis, Cardiff University, 2005.

[27] D. Cosker, D. Marshall, P. Rosin, S. Paddock, and S. Rushton. Towards perceptually realistic talking heads: Models, metrics and McGurk. In *Proc. of ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization (APGV)*, volume 2, pages 270–285, 2004.

[28] D.P. Cosker, A.D. Marshall, P.L. Rosin, and Y. Hicks. Video realistic talking heads using hierarchical non-linear speech-appearance models. In *Proc. of Mirage 2003, INRIA Rocquencourt*, pages 20–27, France, March.

[29] J.A. Costa and A.O. Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Trans. on Signal Processing*, 25(8):2210–2221, 2004.

[30] T.M. Cover and J.A. Thomas. Elements of information theory. *New York*, 1991.

[31] CMU Motion Capture Database. Carnegie-Mellon motion capture database. http://mocap.cs.cmu.edu.

[32] The IXMAS database. http://charibdis.inrialpes.fr.

[33] T. Dean and K.Kanazawa. A model for reasoning about persistence and causation. *Artificial Intelligence*, 93(1-2):1–27, 1989.

[34] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B*, pages 1–38, 1977.

[35] R. Diestel. Graph theory. *Springer-Verlag, Heidelberg*, 2005.

[36] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts. In *Proc. of NIPS*, volume 401, pages 759–760, 2003.

[37] D. L. Donoho and C. E. Grimes. Local Isomap perfectly recovers the underlying parametrization of occluded/lacunary libraries of articulated images. *Technical Report 2002-27, Department of Statistics, Stanford University*, 2002.

[38] D. L. Donoho and C. E. Grimes. When does Isomap recover natural parameterization of families of articulated images? *Technical Report 2002-27, Department of Statistics, Stanford University*, 2002.

[39] D. L. Donoho and C. E. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. In *Proc. of the National Academy of Arts and Sciences*, volume 100, pages 5591–5596, 2003.

[40] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern classification. *Wiley-Interscience Publication*, 2000.

[41] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis: Probabilistic models of proteins and nucleic acids. *Cambridge University Press*, 1998.

[42] V. Estivill-Castro. Why so many clustering algorithms: a position paper. *SIGKDD Explorations*, 4(1):65–75, 2002.

[43] N. Fanizzi, C. dAmato, and F. Esposito. A multi-relational hierarchical clustering method for datalog knowledge bases. In *Proc. of ISMIS*, pages 137–142, 2008.

[44] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[45] T. Friedrich. Nonlinear dimensionality reduction with locally linear embedding and isomap. Master's thesis, Department of Computer Science, The University of Sheffield, September 2002.

[46] A.E. Gelfand and A.F.M. Smith. Sampling-based approaches to calculating marginal densities. *American Statistical Association*, 85:398–409, 1990.

[47] J. Ghosh. Handbook of data mining, chapter scalable clustering methods for data mining. *Lawrence Erlbaum Assoc*, 2003.

[48] E. Gokcay and J.C. Principe. A new clustering evaluation function using renyi's information potential. In *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 6, pages 3490–3493, 2000.

[49] J.J.de Gruijter and A.B. McBratney. A modified fuzzy k means for predictive classification. *Classification and Related Methods of Data Analysis*, pages 97–104, 1988.

[50] D. Guillamet, M. Bressan, and J. Vitria. A weighted non-negative matrix factorization for local representations. In *Proc. of CVPR01*, volume 1, pages 942–947, 2001.

[51] D. Guillamet and J. Vitria. Non-negative matrix factorization for face recognition. *Proc. of CCIA*, pages 336–344, 2002.

[52] D. Guillamet and J. Vitrià. Evaluation of distance metrics for recognition based on non-negative matrix factorization. *Pattern Rocognition Letters*, 24(9-10):1599–1605, 2003.

[53] D. Guillamet, J. Vitria, and B. Schiele. Introducing a weighted non-negative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003.

[54] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering algorithms and validity measures. In *Proc. of the SSDBM Conference*, pages 3–22, 2002.

[55] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: part ii. *SIGMOD Rec.*, 31(3):19–27, 2002.

[56] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

[57] J. Hartigan. Clustering algorithms. *New York*, 1975.

[58] M. Heiler and C. Schnorr. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *JMLR*, 7:1385–1407, 2006.

[59] G. Hinton, Z. Ghahramani, and Y. Teh. Learning to parse images. In *Proc. of Advances in Neural Information Processing Systems*, volume 12, pages 463–469, 2000.

[60] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[61] P.O. Hoyer. Non-negative sparse coding. In *Proc. of IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[62] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[63] M. Hu, C. Ingram, M. Sirski, C. Pal, S. Swamy, and C. Patten. A hierarchical HMM implementation for vertebrate gene splice site prediction. *Technical report, Dept. of Computer Science, University of Waterloo*, 2000.

[64] D.J. Hurley, M.S. Nixon, and J.N. Carter. Force field energy functionals for image feature extraction. In *Proc. of British Machine Vision Conference (BMVC)*, pages 604–613, 1999.

[65] Y.A. Ivanov and A.F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):852–872, 2000.

[66] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[67] A.H. Jazwinski. Stochastic processes and filtering theory. *Academic Press, New York*, 1970.

[68] F. Jelinek. Statistical methods for speech recognition. *MIT Press*, 1997.

[69] O. C. Jenkins and M. J. Mataric. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proc. of the International Conference on Machine Learning (ICML 2004)*, pages 441–448, 2004.

[70] I.T. Jolliffe. *Principal component analysis.* Springer Verlag, New York, 1986.

[71] J. Karaulova, P.M. Hall, and A.D. Marshall. A hierarchical model for tracking people with a single video camera. In *Proc. of British Machine Vision Conference*, volume 1, pages 352–361, 2000.

[72] J. Karhunen and J. Joutsensalo. Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, 7:113–127, 1994.

[73] B. Kegl. Intrinsic dimension estimation using packing numbers. *Advances in NIPS*, 15:681–688, 2002.

[74] R. Kindermann and J.L. Snell. Markov random fields and their applications. *American Mathematical Society*, 1980.

[75] O. Kouropteva, O. Okun, and M. Pietikainen. Selection of the optimal parameter value for the locally linear embedding algorithm. In *Proc. of the 1 st International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'02)*, pages 359–363, 2002.

[76] O. Kouropteva, O. Okun, and M. Pietikainen. Classification of handwritten digits using supervised locally linear embedding algorithm and support vector machine. In *Proc. of the 11th European Symposium on Artificial Neural Networks (ESANN'2003)*, pages 229–234, Bruges, Belgium, April 2003.

[77] A. N. Langville, C. D. Meyer, R. Albright, J. Cox, and D. Duling. Initializations for the nonnegative matrix factorization. In *Proc. of the 12 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 20–27, 2006.

[78] M.H. Law, N. Zhang, and A.K. Jain. Nonlinear manifold learning for data stream. In *Proc. of SIAM Data Mining*, volume 100, pages 33–44, 2004.

[79] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proc. of the IEEE*, volume 86, pages 2278–2324, 1998.

[80] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[81] D.D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*, pages 556–562, 2000.

[82] H. Li and X. Li. Gait analysis using Isomap. In *Proc. of Machine Learning and Cybernetics*, volume 6, pages 3894–3898, 2004.

[83] S. Li, X. Hou, and H. Zhang. Learning spatially localized, parts-based representation. In *Proc. of CVPR*, volume 1, pages 207–212, 2001.

[84] I.S. Lim, P.d.H. Ciechomski, S. Sarni, and D. Thalmann. Planar arrangement of high-dimensional biomedical data sets by isomap coordinates. In *Proc. of CBMS*, pages 50–55, 2003.

[85] W. Liu and N. Zheng. Learning sparse features for classification by mixture models. *Pattern Recognition Letters*, 25(2):155–161, 2004.

[86] W. Liu and N. Zheng. Non-negative matrix factorization based methods for object recogni-tion. *Pattern Recognition Letters*, 25:893–897, 2004.

[87] L. Luo, Y. Wang, and S-Y. Kung. Hierarchy of probabilistic principal component subspaces for data mining. In *Proc. of the IEEE Neural Networks for Signal Processing*, pages 497–506, 1999.

[88] P. McCullagh and J. Yang. How many clusters? *Bayesian Analysis*, 1:101–120, 2008.

[89] G.J. McLachlan and K.E. Basford. Mixture models: Inference and applications to clustering. *Marcel Dekker*, 1988.

[90] M. Meila and D. Hecherman. An experimental comparison of model-based clustering methods. *Microsoft Research, Machine Learning*, 42:9–29, 2001.

[91] Li. Ming and M.R. Sleep. Melody classification using a similarity metric based on kolmogorov complexity. In *Proc. of Conference on Sound and Music Computing*, pages 126–129, 2004.

[92] K. Murphy and M. Paskin. Linear time inference in hierarchical hmms. In *Proc. of Neural Information Processing Systems*, pages 833–840, 2001.

[93] K.P. Murphy. Representing and learning hierarchical structure in sequential data. *Unpublished manuscript, at http://wwwanw. cs.umass.edu/ cs691t/SS02/readings.html*, 2001.

[94] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference, and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

[95] D.J. Navarro and M.D. Lee. Spatial visualisation of document similarity. In *Proc. of Defence Human Factors Special Interest Group Meeting*, pages 39–44, 2001.

[96] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[97] J. Pearl and S. Russell. Bayesian networks. *MIT Press*, 2000.

[98] K.W. Pettis, T.A. Bailey, A.K. Jain, and R.C. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Trans. on PAMI*, 1:25–37, 1979.

[99] J. Platt. Fastmap, MetricMap, and Landmark MDS are all Nystrom algorithms. In *Proc. of 10th International Conference on Artificial Intelligence and Statistics*, pages 261–268, 2005.

[100] R. Pless. Images spaces and video trajectories: Using Isomap to explore video sequences. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 1433–1440, 2003.

[101] R. Pless and R. Souvenir. A survey of manifold learning for images. *IPSJ Tracsactions on Computer Vision and Applications*, 1:83–94, 2009.

[102] T. Poggio and F. Girosi. Network for approximation and learning. In *Proc. of the IEEE*, volume 78, pages 1481–1497, 1990.

[103] M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In *Proc. of NIPS*, volume 14, pages 1258–1262, 2001.

[104] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proc. of the IEEE*, volume 77, pages 257–286, 1989.

[105] M. Rajapakse, J. Tan, and J. Rajapakse. Color channel encoding with nmf for face recognition. In *Proc. of ICIP*, volume 3, pages 2007–2010, 2004.

[106] D.de Ridder and R.P.W. Duin. Locally linear embedding for classification. Technical Report PH-2002-01, Pattern Recognition Group, Imaging Science & Technology Department, Faculty of Applied Science, Delft University of Technology, 2002.

[107] D.de Ridder, O. Kouropteva, O. Okun, M. Pietikainen, and R.P.W. Duin. Supervised locally linear embedding. In *Artificial Neural Networks and Neural Information Processing, ICANN/ICONIP 2003 Proceedings, Lecture Notes in Computer Science 2714*, pages 333–341. Springer.

[108] P. L. Rosin. Unimodal thresholding. *Pattern Recognition*, 34:2083–2096, 2001.

[109] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000. (see http://www.sciencemag.org/cgi/content/full/290/5500/2323).

[110] L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[111] L. K. Saul, F. Sha, and D. D. Lee. Statistical signal processing with nonnegativity constraints. In *Proc. of EuroSpeech*, pages 1001–1004, 2003.

[112] A. Saxena, A. Gupta, and A. Mukerjee. Non-linear Dimensionality Reduction by Locally Linear Isomaps. In *Proc. of ICONIP*, pages 1038–1043, 2004.

[113] V.de Silva and J. Tenenbaum. Unsupervised learning of curved manifolds. In *Proc. of the MSRI workshop on nonlinear estimation and classification*, pages 453–466. Springer Verlag, 2002.

[114] V.de Silva and J. Tenenbaum. Local versus global methods for nonlinear dimensionality reduction. *Advances in Neural Information Processing Systems*, 15:705–712, 2003.

[115] V.de Silva and J.B. Tenenbaum. Sparse multidimensional scaling using landmark points. *Stanford Mathematics Technical Report*, 2004.

[116] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[117] T. Starner and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. on Pattern Analysis and Machine Intelligene*, 20:1371–1375, 1998.

[118] J.B. Tenenbaum, V.de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[119] J.B. Tenenbaum, V.de Silva, and J.C. Langford. The Isomap algorithm and topological stability - response. *Science*, 295(5552):7, 2002.

[120] F.J. Theis, K. Stadlthanner, and T. Tanaka. First results on uniqueness of sparse non-negative matrix factorization. In *Proc. of EUSIPCO*, pages 18–27, 2005.

[121] W. S. Torgerson. Multidimensional scaling: Theory and method. *Psychometrika*, 17:401–419, 1952.

[122] F. S. Tsai, Y. Wu, and K. L. Chan. Nonlinear dimensionality reduction techniques and their applications. *EEE Research Bulletin*, pages 54–60, 2004.

[123] A. Ultsch and C. Vetter. Self-organizing feature maps versus statistical clustering methods. *Technical Report 0994, Universiy of Marburg*, 2001.

[124] R.E. Walpole, R.H. Myers, and S.L. Myers. Probability and statistics for engineers and scientists. *Prentice Hall International*, 1998.

[125] J. Wang, C. Zhang, and Z. Kou. An analytical mapping for lle and its application in multi-pose face synthesis. In *Proc. of BMVC*, pages 38–46, 2003.

[126] Y. Wang and Y.Jiar. Fisher non-negative matrix factorization for learning local features. In *Proc. of Asian Conference on Computer Vision*, pages 27–30, 2004.

[127] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[128] S. Wild. Seeding non-negative matrix factorizations with the spherical k-means clustering. Master's thesis, University of Colorado, 2003.

[129] A.D. Wilson and A.F. Bobick. Recognition and interpretation of parametric gesture. In *Proc. of ICCV*, pages 329–336, 1998.

[130] L. Xie, S. Chang, A. Divakaran, and H. Sun. Feature selection and order identification for unsupervised discovery of statistical temporal structures in video. In *Proc. of IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 29–32, 2003.

[131] L. Xie, S. Chang, A. Divakaran, and H. Sun. Unsupervised discovery of multilevel statistical video structures using hierarchical hidden markov models. In *Proc. of IEEE Intl. Conf. Multimedia and Expo (ICME)*, volume 3, pages 29–32, 2003.

[132] M. Yang. Extended Isomap for classification. In *Proc. of 16th International Conference on Pattern Recognition*, volume 3, pages 615–618, 2002.

[133] M.H. Yang. Face recognition using extended Isomap. In *Proc. of ICIP*, volume 2, pages 117–120, 2002.

[134] M.H. Yang. Discriminant isometric mapping for face recognition. In *Proc. of Computer Vision Systems (ICVS)*, volume 2626, pages 470–480, 2003.

[135] S.J. Young, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P.C. Woodland. The HTK book. *Cambridge University Engineering Department, at http://htk.eng.cam.ac.uk*, 2002.

[136] G. M. Youngblood and D. J. Cook. Data mining for hierarchical model creation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(4):561–572, 2007.