Learning Non-linear Models of Shape and Motion

A Thesis submitted for the degree of Doctor of Philosophy

By

Richard Bowden

Department of Systems Engineering, Brunel University

October 1999

Abstract

Deformable models have been an active area of research in computer vision for a number of years. Their ability to model non-ridgid objects through the combination of geometry and physics has proven a valuable tool in image processing. More recently a class of deformable objects known as Point Distribution Models or Eigen Models have been introduced. These statistical models of deformation overcome some of the shortfalls of earlier deformable models by learning what is 'allowable' deformation, for an object class, from a training set of examples. This semi-automated learning procedure provides a more generic approach to object recognition, tracking and classification. Their strength lies in their simplicity and speed of operation, allowing the robust ability to model complex deformations in cluttered environments. However, the automated construction of such models leads to a breakdown of the fundamental assumptions upon which they are based. Primarily, that the underlying mathematical model is linear in nature. Furthermore, as more complex objects are considered, these assumptions fail completely and what is produced is an unreliable model.

This work addresses these problems and presents novel techniques for the automated construction and application of non-linear deformable models, which retain the speed, and simplicity of the linear Point Distribution Model. It is further shown how these non-linear models can be augmented with probabilistic temporal constraints, which are essential in object tracking and classification.

This work presents, in essence, three developments to the field. Firstly, a piecewise linear approach to modelling non-linearity is proposed and results demonstrated that show its accuracy in modelling both low and high dimensional datasets with heavy non-linearity. The technique is then extended to the automated construction of models. Secondly, it is shown how the piecewise approach can be augmented with temporal constraints and used in both model prediction, animation and for the support of multiple hypotheses during tracking. It is further shown how these temporal models can be extended to incorporate

information from other sources, providing more reliable tracking in the absence of complete training data. Thirdly, it is shown how elements can be combined statistically and used to infer information about an object from its shape alone. Using human motion capture as an example, it is demonstrated that models can be assembled which allow 3D structural information about body pose and motion to be inferred from a monoscopic image sequence using only natural features of the body as markers.

Acknowledgements

I would like to thank both EPSRC and Foster Findaly Associates Ltd who supported this work financially. Televirtual Ltd for access to the human motion capture data, the Turing Institute for the 3D surface head models and the Centre for Medical Imaging Research at the University of Leeds for access to the MRI datasets of the human hand.

I would like to thank my supervisors Dr Tom Mitchell and Professor Mansoor Sarhadi for having the confidence in my abilities to allow me to pursue the topics which excite me. Tom has been more of a friend than a supervisor during the course of my studies and invaluable as a sounding board for ideas and approaches. Mansoor has been more than generous in both his time and confidence in my abilities and I thank him for the opportunities he has provided me.

I would like to thank my colleagues at Brunel and other institutions for their numerous stimulating conversations and insights into my work, you know who you are. A special word of thanks for Dr Dave Cohen for the many valued discussions related to this work.

Id like to thank my friends and family, especially my parents, for the opportunities they have provided to their children and for teaching us all that our abilities were only limited by our imaginations. For pushing us, understanding us, and helping us to achieve our dreams.

Lastly I would like to thank my wife Beccy for the constant encouragement, proof reading and tolerance of my absence. It is a debt I can never repay. I could not have done it without you.

Declaration

Elements from this manuscript have appeared in, or are about to appear in the following publications.

Non-linear Statistical Models for the 3D Reconstruction of Human Pose and Motion from Monocular Image Sequences. , R. Bowden, T. A. Mitchell, M. Sarhadi, To appear in Image and Vision Computing.

Non-linear Point Distribution Models, R. Bowden, In CVonline: On-Line Compendium of Computer Vision [Online]. R. Fisher (ed). Section 11.3.1.2 , Oct 98.

Reconstructing 3D Pose and Motion from a Single Camera View, R. Bowden, T. A. Mitchell, M. Sarhadi, In Proc. BMVC, John N. Carter & Mark S. Nixon Eds, Uni of Southampton, Vol 2, pp , Southampton, Sept 1998.

Cluster Based non-linear Principal Component Analysis, R. Bowden, T. A. Mitchell, M. Sahardi, IEE Electronics Letters, 23rd Oct 1997, 33(22), pp1858-1859.

Real-time Dynamic Deformable Meshes for Volumetric Segmentation and Visualisation, R. Bowden, T. A. Mitchell, and M. Sahardi. In Proc. BMVC, Adrian F. Clark Ed, Vol 1, pp 310-319, Essex, UK, Sept 1997.

Virtual Datagloves: Interacting with Virtual Environments Through Computer Vision, R.Bowden, A. J. Heap and C. Hart, In Proc. 3rd UK VR-Sig Conference, DeMontfort University, Chris Hand Ed, Leicester, UK, July 1996.

Some elements of this work are similar in nature and content to the work of A. J. Heap. However, this work was done concurrently and in isolation as demonstrated by the publication, Cluster Based non-linear Principal Component Analysis, which was submitted to IEE Electronics Letters on 2nd Sept 1997 and predates the work of the other author.

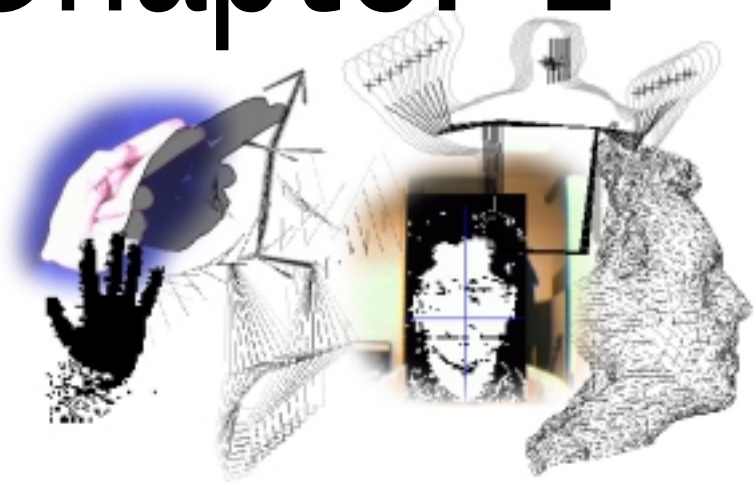
Table of Contents

Table of Figures

List of Abbreviations

ACM	**A**ctive **C**ontour **M**odel
ARM	**A**ctive **R**egion **M**odel
ASL	**A**merican **S**ign **L**anguage
ASM	**A**ctive **S**hape **M**odel
BSL	**B**ritish **S**ign **L**anguage
CBNLPDM	**C**luster **B**ased **N**on-**l**inear **P**oint **D**istribution **M**odel
CSSPDM	**C**onstrained **S**hape **S**pace **P**oint **D**istribution **M**odel
CCD	**C**harge **C**oupled **D**evice
CONDENSATION	**Con**ditional **Dens**ity Propag**ation**
FCM	**F**uzzy **C** **M**eans (k-means) Clustering Algorithm
HSV	**H**ue **S**aturation **V**alue
HSB	**H**ue **S**aturation **B**rightness
HLS	**H**ue **L**ightness **S**aturation
HSL	**H**ue **S**aturation **L**uminosity
HVC	**H**ue **V**alue **C**hroma
$\mathbf{I_{x,y}}$	Intensity of pixel at point (x,y)
ISL	**I**nternational **S**ign **L**anguage
MRI	**M**agnetic **R**esonance **I**maging
NLPDM	**N**on-**l**inear **P**oint **D**istribution **M**odel
PCA	**Principal C**omponent **A**nalysis
PDF	**P**robability **D**ensity **F**unction
PDM	**P**oint **D**istribution **M**odel
RGB	**R**ed **G**reen **B**lue
ROI	**R**egion **o**f **I**nterest
σ	Standard Deviation
Voxel	**Vo**lumetric **El**ement

Chapter 1



1 Introduction

The term *Computer Vision* covers a broad field of research encompassing many techniques, applications and disciplines but is commonly summarised as

"the science of making a computer see...."

However, the goal is often to allow the computer to understand what it sees to some extent, and it is here that the science embraces aspects of artificial intelligence. This artificial understanding, or interpretation, of a scene stems from human perception and our attempt to mimic the functionality of the human visual system. It is natural to attempt to emulate the way in which humans perceive or interpret the world and this approach has been instrumental throughout the course of vision research, with developments such as foveal vision systems and stereoscopic depth reconstruction. The most fundamental of such approaches is that of model based vision.

The image plane of a camera is akin to the retina of the eye, and images projected onto it are the 2D projection of the 3D world. This loss of information presents no obstacle for the human brain which interprets the image seamlessly, constantly updating its model of the world. The ability to judge depth through the

disparity of objects falling upon the retinas provides essential clues to the brain about the structure of the real world. However, even when this stereoscopic information is unavailable, the human brain can still interpret the scene and accurately estimate the position and orientation of objects. This is due to the huge knowledge base the brain accumulates about the 3D world, its laws, and the shape and structure of objects and how they project onto the retina.

If the human brain can achieve such feats for millions of objects, then the rationale of providing a similar knowledge of a small subset of objects to a computer is an obvious solution. This is the premise of *model based vision*, where an internal representation of the world or object is provided to a computer allowing it to locate, recognise, track or interact with real world objects. This *a priori* knowledge about objects can be encapsulated and represented in numerous ways.

Probably the simplest form of *model based vision* is that of template matching [Ballard 82]. Given a known object or feature to be located in an image, a template, representing object features, is applied to the image at every location. By formulating template matching with a scoring mechanism, the fit of the model at any location can be assessed and the probable position of objects or features estimated. Although a relatively time consuming approach, template matching algorithms can provide effective object location for constrained applications and have proven invaluable in areas such as industrial inspection. Hardware implementations are commonplace allowing large numbers of templates to be matched in real time.

Industrial inspection has proven a successful application of real time vision systems as the nature of the problems is typically heavily constrained. If the application of biscuits on a conveyor belt is considered, the problem of object location is greatly simplified by the process and nature of the object. The production line produces only biscuits, so the variability of shape is heavily reduced. Biscuits are typically flat and as such can be assumed to be 2D objects, which adhere to ground plane constraints. In addition, lighting inconsistencies and background clutter can be controlled and modelled accurately. Given a

ridgid internal model of an object, probable locations can be identified within the image by matching the features of the object with the extracted features of an image (such as edges or corners). This is often applied as a hypothesise and test procedure, where possible locations of an object are generated and compared to the image. Each hypothesis is then assessed using some metric where the highest scoring hypotheses correspond to the likely location of objects. As more complex objects are considered, techniques such as geometric hashing [Wolfson 92] can be used to allow affine object transformations. However, when real world objects and less constrained environments are considered these tools are insufficient at modelling object variability.

The problems of recognition are compounded when everyday, unconstrained objects are considered. In addition to the variability of lighting, shading and complex scenes containing cluttered backgrounds, even ridgid 3D objects will produce considerably differing views depending upon their position and orientation. Consider a book. The shape of the book projected onto the image frame will vary immensely as its orientation changes. More complex still is the goal of building a generic model of a book where the 3D shape parameters of the object vary immensely between examples. A common solution to this problem is to represent the object in terms of its 3D structure and use the 2D projection of the internal model to match with the 2D projection of the real world object.

Models that bend or articulate introduce further complexity to the task of object recognition and tracking. In addition to the object variation described above, articulated objects also produce variability of shape and structure in the image. Many researchers have tackled this by extending the 3D internal model to that of articulated geometric primitives with tight joint constraints, which closely mimic the movement of the real world object. However, as these types of models are typically hand-coded they do not offer a generic solution that can be applied to all objects.

Deformable objects which can alter their shape to fit an object under some global shape constraints overcome these problems by encapsulating a large amount of an object’s variability into a constrained deformation of a contour or object. By

learning this deformation from a training set of example shapes, they produce a set of tools which allow models to be easily constructed for any number of objects under a multitude of situations.

This thesis is concerned with the construction of generic models of deformation and their application to the recognition and tracking of complex 3D objects. Chapter 2 will present a review of relevant literature to the work and discuss the shortfalls of current formulations. Chapter 3 will introduce linear Point Distribution Models and describe the Active Shape Model approach to object tracking. Chapter 4 will discuss the use of colour in image segmentation and feature extraction. Chapter 5 will present a non-linear approximation technique based upon a piecewise linear model. Chapter 6 will extend the piecewise linear approach to more complex, high dimensional training sets and demonstrate the use of such models in the classification of American Sign Language. Chapter 7 will discuss the addition of temporal constraints. Using motion capture as an example it is shown how time dependent deformation can be both learnt and reproduced from a model. Its is further shown how these temporal constraints can be used to support multiple hypotheses during tracking. Chapter 8 discusses the extension of PDMs into the 3D domain. Chapter 9 presents a new approach to markerless based motion capture which incorporates many of the previously discussed elements to allow the 3D pose and motion of a human body to be extracted from a monoscopic image sequence. Finally a discussion and conclusions are presented.

This manuscript also contains two appendices. Appendix 1 presents the k-means and fuzzy k-means (FCM) algorithms along with associated techniques. Appendix 2 presents a new approach to the surface segmentation of volumetric data. Although this work is extremely relevant to 3D PDM construction it stands as an individual piece of research and hence is consigned to the appendices.

Chapter 2

2 Literature Review

2.1 Introduction

An initial literature review was performed which surveyed the field of 3D computer vision. The review covered types of image data from 2D images, range data and depth maps to volumetric segmentation. Acquisition methods, reconstruction and image segmentation were also covered and conclusions drawn to support the remainder of the research. This initial survey was too general for inclusion within this manuscript and hence is available as a separate technical report [Bowden 96].

The conclusions of the report were that contour or surface based approximations (specifically statistical contour models) are important for the following reasons:

- Image searching is localised along contour boundaries and hence provides significant computational savings over more traditional low level image processing techniques. This benefit is more apparent where real-time processing of image sequences or large volumetric datasets are considered.
- The ability to introduce *a priori* knowledge about object shape and deformation into a contour provides a robust deformable template

which can be applied to an image where the absence or occlusion of object features and cluttered/complex backgrounds would result in the failure of other techniques.

- The ability to accurately segment objects from images or sequences provides smoothed object boundaries.
- The ability to aid in the classification of objects under affine transformation.

The remainder of this chapter will present a more specific review of related literature, namely in the area of statistical models of deformation and associated approaches.

2.2 Contour Models

The seminal work of Kass *et al* on *Snakes* or the *Active Contour Model* (ACM) presented a class of semi-automatic methods for segmentation using energy minimising curves [Kass, 1988; Kass, 1987]. In these methods, a user draws the approximate boundary of the region of interest in an image. Then, an elastic contour is fitted to the boundary points and the curve is iteratively refined until its internal energy defined by its curvature is minimised while responding to external forces derived by image edges. Many researchers have shown how these active contour models can be used to locate and track an object in an image [Etoh, 1992; Ueda, 1992; Cipolla, 1992].

Zhou and Pycock segment cells from 2D images using statistical models applied like snakes [Zhou, 1995; Zhou, 1995]. Models are built up for different forms of cells; the interpretation process optimises the match between models and the data using a Bayesian distance measure. Lobregt and Viergever extend upon this model, presenting solutions to the problems of unwanted deformation like shrinking and vertex clustering [Lobregt, 1995]. There is a wealth of published work on variations on the basic model proposed by Kass *et al*, all use the same basic model with small constraints added to allow *a priori* knowledge of shape to be imposed upon the model and hence provide better performance.

Terzopolous and Vasilescu [Terzopoulos 91] extended the snake model to include an inflation force that helps remove the need for initial contour placement and thus avoid convergence on local minima. The inflation force drives the snake model outwards towards the object boundary like an inflating balloon. Terzopolous and Vasilescu formulated the model as a finite element mesh and later extended the model to a thin plate spline, demonstrating successful results in the reconstruction of range data and volumetric CT data surface representations [McInery 93]. Bowden *et al* extended this work further and is discussed in more detail in Appendix 2 [Bowden 97].

Several researchers have proposed B-Spline variations of the active contour model [Rueckert, 1995; Schnabel, 1995; Blake 1998]. Schnabel and Arridge looked at the problems associated with high curvature in active contour models, proposing a curvature matching technique for isophoto curvature matching. They look at the applications of using this approach to segment high curvature contours of the brain from medical images. Blake and Isard have combined many of their publications on the subject in the text 'Active Contours' which covers the construction, tracking and applications of B-spline contour approximations [Blake 1998].

It has been shown that these 2D models can be used to reconstruct 3D surfaces from volumetric data by applying snakes to individual slices to extract contours that can then be reconstructed into a 3D model [Carlbom, 1994; Goshtasby, 1995]. A typical implementation of such a system uses the final model from one slice as an initial estimate for the next to reduce user intervention.

Ivins and Porrill presented Active Region Models [Ivins 98], an adaptation to Kass's Active Contour Models where colour regions within an image are used to locate and track the boundarires of regions within the image.

A Neural network approach was proposed by Chiou *et al* called the neural network based stochastic active contour model (NNS-SNAKE) which integrates a neural network classifier for systematic knowledge building, and an active

contour model for automated contour location, using energy functions to stochastically locate the most probable contour.

2.3 Statistical Models of Deformation

A Point Distribution Model (PDM)[Cootes 95] gets its nickname of 'Smart Snake' from its obvious similarity to elastic snakes (Active Contour Models, ACM [Kass, 1987]). The major difference is that while snakes retain shape information in the elasticity and rigidity of their constituent points, a PDM uses a statistical model to specify allowable deformations. This not only makes the PDM less computationally expensive than the ACM but deformation is easier to build into the model.

Since they were proposed by Cootes *et al,* a wealth of research has been undertaken into Point Distribution Models. A PDM (the underlying mathematical model) or Active Shape Model (the model's applied name) is a statistical model which can be constructed from a training set of correctly labelled images. A PDM represents an object as a set of labelled points, giving their mean positions and a small set of modes of variation which describe how the object's shape can change. These modes of variation are gained from Principal Component Analysis (PCA) on the training set and represent the largest eigenvectors of the covariance matrix. An Active Shape Model exploits the linear formulation of PDMs in an iterative search procedure capable of rapidly locating the modelled structures in noisy, cluttered images, even when partially occluded [Cootes, 1995].

Turk and Pentland [Turk 91] present a method for extracting only the number of eigenvectors equal to the number of training examples and not the dimensionality of the set, in a similar manner to that of Cootes *et al* [Cootes 95] and this is discussed in more detail in Chapter 3.

It has been shown by Bowden *et al* that the PDM provides sufficient dimensional reduction inherent to the model to enable the simple classification of static shape [Bowden, 1995; Bowden, 1996]. These authors outline a simple method for using this dimensional reduction to classify shape deformation from the variation

weights from the mean. They show how static gestures can be recognised in real-time for a PDM of the human hand.

Lantis, Taylor and Cootes have also extended their initial work from contour models to shape and grey-level models [Lantis, 1994]. They use a combined PDM that uses both shape and a grey scale maps to locate and identify human faces.

Turk and Pentland use principal component analysis to describe face images in terms of a set of basis functions or 'eigenfaces'. Though valid modes of variation are learnt from a training set, and are more likely to be more appropriate than a 'physical' model, the eigenface is not robust to shape changes, and does not deal well with variability in pose and expression. However, the model can be matched to an image easily using correlation-based methods [Turk 91].

Magee and Bole presented Vector Distribution Models, where points around a connected contour are converted into a vector, and these vectors are concatenated into a final training vector on which PCA is performed [Magee 98]. These authors went on to discuss the use of Canonical Analysis, a similar procedure to PCA where two co-variance matrices are formed, one describing Intra class variation and one Inter class variation. After extraction of a generalised eigen system a new eigen space is extracted. Although this space may not necessarily be optimised for dimensional reduction, it is useful for data classification as the first components of the model represent inter-class variation [Magee 99].

Swets and Weng [Swets 96] presented a technique called a combined eigen-canonical transform which combined canonical analysis with PCA to give data reduction and improved classification. Canonical analysis was performed on data after it had been projected down into the lower eigen space gained from PCA similar to that outlined in section 6.

Initial work of extending the PDM (Active Shape Model) to 3D has already been proposed by [Hill, 1995].

Ferryman *et al* use PCA on 3D rigid models to build a deformable model for various different car shapes which is used to locate and track moving traffic [Ferryman, 1995]. The process is very similar to that of the PDM. However, instead of modelling the object as points that make up the boundaries of the object, points are chosen at landmarks such as corners, and the model built up from the known interconnection of these points.

O'Toole *et al* presented work for 3D models of faces represented as a mean face with weightings that can be used to deform the model [O'Toole 96]. Faces were built up as 3D surfaces from a set of 65 male and 65 female heads. PCA analysis was performed to provide a compact model. They show that the primary mode of variation of the eigenface data set provides the mapping from a male head to a female head.

2.4 Non Linear PDMs

The linear formulation of the PDM relies on the assumption that similar shapes will produce similar vectors. This being the case, it is a fair assumption that the training set will generate a cluster in some shape space. However, it is unfair to assume that this cluster will be uniform in shape and size. As more complex models are considered the training set may even generate multiple, separate clusters in the shape space.

Under these circumstances the linear PDM will begin to fail as non-linear training sets produce complex high dimensional shapes which, when modelled through the linear mathematics of PCA, produce unreliable models. The nature of non-linear shape spaces will be discussed in depth in later chapters but a number of authors have addressed the problems associated with the construction of non-linear PDMs.

Where rotational non-linearity is known to be present within a model this can be removed/reduced by mapping the model into an alternative linear space. Heap and Hogg suggested using a log polar mapping to remove non-linearity from the training set [Heap 95]. This allows a non-linear training set to be projected into a

linear space where PCA can be used to represent deformation. The model is then projected back into the original space. Although a useful suggestion for applications where the only non-linearity is pivotal and represented in the paths of the landmark points in the original model, it does not provide a solution for the high non-linearity generated from other sources.

Higher order non-linearity is often the result of incorrect labelling of training examples. By carefully selecting landmark points by hand, a near optimum labelling can be achieved which will minimise the non-linearity of a training set. However, for all but the most simple of cases this is not a feasible solution. Often semi-automated procedures are used where a user can speed up the process of labelling example shapes for analysis. Fully automated procedures are rarely used due to the problems of correctly assigning landmarks and the highly non-linear models that this produces.

Work done by Baumberg and Hogg goes some of the way to solving non-linearity in deformable models by using a B-Spline representation. Landmark points for the Spline are represented as a PDM [Baumberg, 1995]. The curvature of the B-Spline takes on some of the non-linearity of the model and therefore reduces the problems presented with linear PDM representing non-linear models.

It has been proposed by Kotcheff and Taylor that non-linearity introduced during assembly of a training set could be eliminated by automatically assigning landmark points in order to minimise the non-linearity of the corresponding training cluster [Kotcheff 97]. This can be estimated by analysing the size of the linear PDM that represents the training set. The more non-linear a proposed formulation of a training set, the larger the PDM needed to encompass the deformation. The procedure was demonstrated using a small test shape and scoring a particular assignment of landmark points according to the size of the training set (gained from analysis of the principal modes and the extent to which the model deforms along these modes, i.e. the eigenvalues of the covariance matrix). This was formulated as a minimisation problem, using a genetic algorithm. The approach performed well but at a heavy computation cost [Kotcheff 97].

As the move to larger, more complex models or 3D models is considered, where dimensionality of the training set is high, this approach becomes unfeasible. A more generic solution is to use accurate non-linear representations. As linear PCA is used for linear PDMs, so, non-linear PCA can be used to model non-linear PDMs and many researchers have proposed approaches to this end.

Sozou et al first proposed using polynomial regression to fit high order polynomials to the non-linear axis of the training set [Sozou 94]. Although this compensates for some of the curvature represented within the training set, it does not adequately compensate for higher order non-linearity, which manifests itself in the smaller modes of variation as high frequency oscillations. In addition, the order of the polynomial to be used must be selected and the fitting process is time consuming.

Sozou et al further proposed modelling the non-linearity of the training set using a backpropagation neural network to perform non-linear principal component analysis [Sozou 95]. This performs well, however the architecture of the network is application specific; also, training times and the optimisation of network structure are time consuming. What is required is a means of modelling the non-linearity accurately, but with the simplicity and speed of the linear model.

Several researchers have proposed alternatives, which utilise non-linear approximations, estimating non-linearity through the combination of multiple smaller linear models [Bowden 97; Bregler 94; Cootes 97; Heap 97]. These approaches have been shown to be powerful at modelling complex non-linearity in extremely high dimensional feature spaces [Bowden 97].

The basic principle behind all these approaches is to break up any curvature into piecewise linear patches, which estimate the non-linearity rather than modelling it explicitly. This is akin to the polygonal representation of a surface. A smooth curved surface can be estimated by breaking it down into small linear patches. In the field of computer graphics this technique is performed to reduce render time. There exists, of course, a trade off between visual accuracy and computation

speed (where the minimum numbers of polygons are used to achieve the desired appearance). The same problem is present in non-linear PDM estimation, where the minimum number of linear patches that accurately represent the model must be determined.

Bregler and Omohundro suggested modelling non-linear data sets of human lips using a *Shape Space Constraint Surface* [Bregler 94]. The surface constraints are introduced to the model by separating the space surface into linear patches using cluster analysis. However the dimensionality of these '*lip*' shape spaces is low as is the non-linearity due to the simplified application of the work.

Cootes and Taylor suggested modelling non-linear data sets using a Gaussian mixture model, which is fitted to the data using Expectation Maximisation [Cootes 97]. Multiple Gaussian clusters are fitted to the training set. This provides a more reliable model as constraints are placed upon the bounds of each piecewise patch of the shape space, which is modelled by the position, and size of each Gaussian.

Both of these estimation techniques become unfeasible as dimensionality and training set size increase. However by projecting the training set down into the linear subspace as derived from PCA the dimensionality and therefore computation complexity of the non-linear analysis can be reduced significantly to facilitate statistical and probabilistic analysis of the training set. This projection relies upon the dimensional reduction of PCA while retaining the preservation of the important information, the shape of the training set [Bowden 97; Bowden 98] and will be discussed fully in the following Chapters.

2.5 Tracking

By treating the problem of model fitting and tracking as an optimization technique the problems of discontinuity can be overcome. Hill *et al* proposed using genetic algorithms to model the discontinuous changes in shape space/model parameters [Hill 91][Hill 92]. Cootes *et al* present the use of genetic algorithms for initial image search and initialisation of PDMs within the image

frame [Cootes 95]. The use of genetic algorithms to overcome the complexities of tracking with the piecewise non-linear model has been investigated. However, the performance of such an approach relies largely on the formulation and structure of the genetic algorithm itself.

Blake *et al* emphasised the advantage of using low-parameter descriptions of deformable models in terms of B-Splines [Blake 93]. In this method, a deformable model is regarded as a linear combination of basis templates, and the state of the model is specified by a vector of coefficients for these templates. The mode leads naturally to a Kalman filter formulation in which the model is driven by an explicit local search for edges lying perpendicular to its boundary. These suggested movements are then used to update the model via the Kalman filter. Ivinns and Porrill suggested a similar approach but propsed an alternative to the Kalman filter using an explicit least-squares approximation [Ivins 98].

Numerous approaches and variations exist on the subject of object tracking but a recent development is that of CONDENSATION [Blake 98][Isard 98]. Blake and Isard presented the Stochastic **Con**ditional **Dens**ity Propag**ation** (CONDENSATION) algorithm in which the location of a contour or object is probabilistically tracked over time using a model of the object's dynamics to predict movement. Objects are not represented by a single parameterisation but instead by a probability density function (PDF) which represents all possible parameterisations of the model. By generating multiple hypotheses from this distribution at each iteration, and checking each hypothesis against the image for supporting information, CONDENSATION allows objects to be tracked which exhibit discontinues movement in complex noisy scenes.

Chapter 3

3 Linear Point Distribution Models

3.1 Introduction

The principle behind the Point Distribution Model (PDM) [Cootes 95] is that the shape and deformation of an object can be expressed statistically by formulating the shape as a vector representing a set of points that describe the object. This shape and its deformation (expressed with a training set, indicative of the object deformation) can then be learnt through statistical analysis. The same technique can be applied to more complex models of grey scale appearance or combinations of these techniques [Cootes 93][Lantis 95][Cootes 98]; however, the underlying linear mathematics for model representation remains the same.

This chapter will introduce the principle, construction and application of Point Distribution Models. Section 3.2 will provide an overview of PDM construction. Section 3.3 will discuss the use of PDMs in tracking deformable objects and section 3.4 will briefly discuss the reconstructive ability of models. Lastly conclusions will be drawn.

3.2 Constructing a Point Distribution Model

3.2.1 Overview

To construct a point distribution model the shape of an object is expressed mathematically as a vector. For a simple 2D contour, each pose of the model is described by a vector $\underline{\mathbf{x}}_i \in \Re^{2n} = (x_1, y_1, \ldots, x_n, y_n)$, representing the set of points specifying the path of the contour (see Figure 3.2.1). A training set **E** of N vectors is then assembled for a particular model class. In each example, the points which specify the shape of the contour are selected such that there is a correspondence of features between examples, e.g. in the hand example, if the j^{th} point (x_j,y_j) is the tip of the middle finger, it should remain so throughout all training examples. In order to achieve this it is often necessary to align the examples with each other and resample the contour by identifying landmark points to provide consistency throughout the training set.

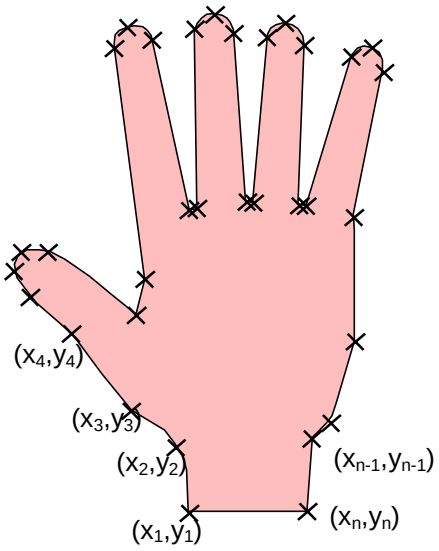


Figure 3.2.1 - 2D Contour of a hand

As the vector, $\mathbf{x_i}$, is effectively a point in a $2n$ dimensional space ($\mathbf{x}_i \in \Re^{2n}$) and each vector is similar in shape, each example will produce a similar point in this $2n$ dimensional shape space. In fact, it would be expected that the training set will form a relatively tight cluster. By analysing the shape of this cluster, the deformation contained within the training set can be learnt and generalised. This is done by making the assumption that the shape of the cluster is hyper-elliptical

and performing Principal Component Analysis (PCA) upon the mean zeroed training set to discover the position and parameters of the ellipsoid in shape space.

PCA projects the data into a linear subspace with a minimum loss of information by multiplying the data by the eigenvectors of the covariance matrix constructed from the training set. By analysing the magnitude of the corresponding eigenvalues, the minimum dimensionality of the space on which the data lies can be calculated and the information loss estimated.

The principle is demonstrated in Figure 3.2.2, where the primary orthogonal axis and its bounds are determined which describe the 3D elliptical cluster. The centeroid of the cluster (i.e. the mean vector) is the mean shape of the training set. The vector $\mathbf{v_1}$ is the primary axis of the cluster with $\mathbf{v_2}$ the secondary orthogonal axis and $\mathbf{v}_3$ the third. Once this analysis has been performed the shape can be restricted to lie within this cluster so constraining the shape of the model. From this *learnt* model of deformation, all shapes that were present in the training set **E** can be reconstructed. In addition, many other shapes (hopefully viable) not present within the original training set can also be constructed i.e. the PDM generalises the shape space contained in **E**.

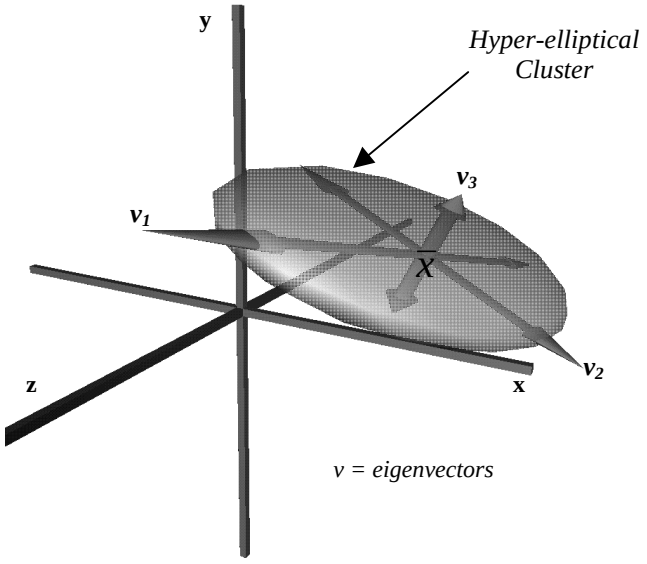


Figure 3.2.2 - Hyper-elipsoid in n Dimensional Space

Unfortunately, for all but the most simple of PDMs this hyper-elliptical assumption does not hold true and the linear mathematics behind the process generates a weak/un-robust model. This will be discussed in more detail in Chapter 5.

The construction of a Point Distribution Model can be summarised with the following algorithm,

1. Assemble a training set of shapes that represent an object class and its indicative deformation.
2. Resample each example to provide a consistent dimensionality throughout the training set.
3. Minimise the difference between examples by aligning each training example using rotation, scaling and translation.
4. Normalise the training set to provide numerical stability
5. Learn the shape space by performing Principal Component Analysis (PCA)

N.B. Steps 2 and 3 can be reversed depending upon the schemes used.

The remainder of this section will consider each of these steps in turn.

3.2.2 Obtaining Training Examples

In order to learn the natural deformation of an object class, a training set is first assembled. This training set must be indicative of the object deformation that is to be learnt.

Typically, training examples are extracted by hand (as in [Cootes 95][Ferryman 95][Heap 95]) to ensure that a uniform and well-labelled training set is obtained. However, for all but the most simple of objects this is an unfeasible approach.

Other approaches to the automatic and semiautomatic generation of training examples are the use of snakes [Kass 88] to segment simple deformable objects

from image sequences. In a temporal image sequence the pose of a converged snake can be used as an initial estimate for the next frame reducing the susceptibility of snakes to their initial location. Cootes et al have also proposed using the PDM itself to locate new objects by bootstrapping the procedure and using a partially 'learnt' PDM to constrain segmentation of future models.

Other researchers have shown how incremental eigen models can be used to recalculate the deformation of a model in light of new training examples without the need for a full decomposition on the co-variance matrix [Hall 98]. Although it has not been demonstrated that this could be used in the construction of examples, it is evident that this type of procedure could be invaluable in the automated construction of deformable models. An initial PDM could be used to locate and extract further examples which could then be added to the model, without the need for a full recomputation of the model.

A simple but effective approach can be achieved by tracing by hand a 2D contour representing features from an image and recording the path taken as the shape is traced. Although this aids in the assembly of a model, producing a chain code representation of the contour, it must be correctly labelled and resampled to put training examples within a mathematical framework on which PCA can be performed.

Automated methods produce similar results and can easily be achieved where only external boundaries are required. Throughout this work a common technique used to automatically extract contours is a simple boundary-tracing algorithm on binary blobs to extract the external contour of objects. This is facilitated through the use of a blue screen techniques to aid binary segmentation and will be seen in later chapters.

3.2.3 Landmark Point Assignment

In order to perform statistical analysis on a training set the procedure assumes a single cluster is formed in shape space by the training set. This assumption works on the principle that common points along the contour boundary do not change

between examples. Similar shapes therefore produce similar vectors which occupy a tight cluster in shape space. However, in order for this assumption to hold true, consistent points along the contour must be located.

The acquisition method for training data, as previously discussed, depicts the extent of this problem. Where a simple chain code representation is generated, there is no guarantee of consistency between examples. In fact, examples will generally differ in length due the size, shape and orientation of the object and how it projects onto the image plane. As the shape deforms, the number of pixels constituting the contour varies. As PCA relies on learning a hyper-ellipsoid in *n* dimensions, all examples must be *n* dimensional.

A simple form of resampling can be performed by equally spacing the new *n* dimensional vector along the original point contour using linear interpolation. However, this simple resampling scheme leads to a break down of the single cluster assumption (see Chapter 6.5.5). To provide a better sampling scheme landmark points are identified which correspond to specific features of the contour and resampling performed between them. These landmarks could be high curvature areas, corners or the physical features of an object. Whether extracted manually or automatically, the number of successfully located landmark points will increase the correspondence between training examples.

Techniques such as snakes and the bootstrap PDM methods mentioned in the previous section help alleviate this problem as they produce examples which are naturally within the PCA co-ordinate frame.

Other labelling techniques have been proposed such as Genetic Algorithms (see Chapter 2).

3.2.4 Training Set Alignment

Cootes *et al* suggested aligning training examples by calculating the scaling, translation and rotation of each model to minimise the sum of the squares of distances between equivalent points for all examples. This exhaustive process

although suitable for simple 2D contours of low dimensionality does not provide a suitable approach for more complex high dimensional objects.

In order to reduce the computational complexity of the approach it is possible to locate specific features of the object such as high points in curvature, or the moments of the object, and minimise according to these features. This can be done by analysing the constituent points of the contour and extracting specific features. Figure 3.2.3 demonstrates an approach to alignment by calculating the primary axis of the 2D contour: (a) The contour is first translated so the centroid of the object is at the origin; (b) By performing PCA on the contour points, the principal axis of the shape can be determined; (c) Finally the contour is rotated so the moments of the shape are aligned with the axis of the co-ordinate system.

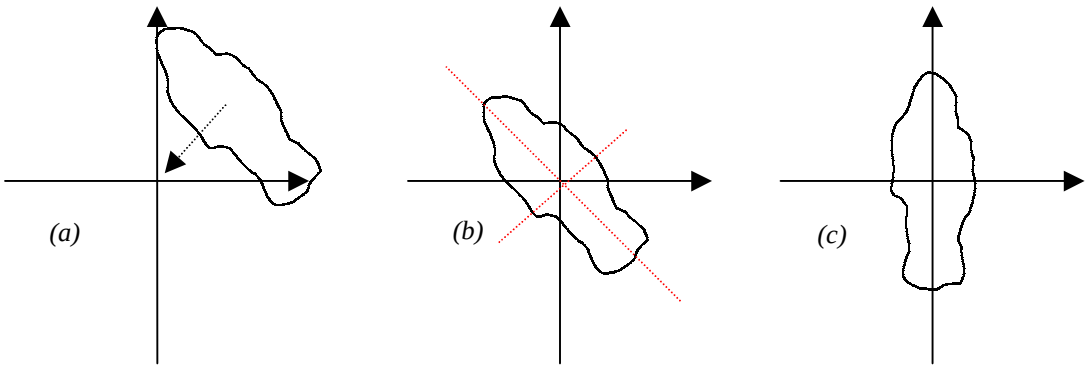


Figure 3.2.3 - Aligning the training set

(a) Move centeroid to origin, (b) Find Principal axis of shape
(c) Rotate to align object

It is necessary to rescale the training set to provide numerical stability during the learning process. However, if each shape is simply normalised, important information about the relative size of examples is lost. A suitable scaling for the contour can be extracted by calculating the mean distance of contour points from the origin (centroid) over the entire training set and scaling each accordingly, where

Equation 3.2-1 $$\left|\mathbf{x}_i\right|' = \frac{\mathbf{x}_i}{\left|\bar{\mathbf{x}}\right|} \text{ and } \left|\bar{\mathbf{x}}\right| = \frac{1}{N}\sum_{j=1}^{N}\left|\mathbf{x}_j\right|$$

This gives a pseudo normalisation where all training examples are approximately unit length while retaining the subtle size variation between examples. As this procedure uses the moments of the contour as features, this alignment can be performed prior to resampling and used to aid landmark point assignment.

3.2.5 Learning Shape Space

Once a resampled training set **E** of *N* examples, $\mathbf{x_i}$ *(i=1, ..., N),* is assembled. The training set **E** is aligned (using translation, rotation and scaling) and the mean shape calculated by finding the average vector. To represent the deviation within the shape of the training set Principal Component Analysis is performed on the deviation of the example vectors from the mean using eigenvector decomposition on the covariance matrix **S** of **E** where,

Equation 3.2-2
$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

The *t* unit eigenvectors of **S** (corresponding to the *t* largest eigenvalues) supply the variation modes; *t* will generally be much smaller than *2n*, thus giving a very compact model. A deformed shape **x** is generated by adding weighted combinations of $\mathbf{v_j}$ to the mean shape:

Equation 3.2-3
$$\mathbf{x} = \bar{\mathbf{x}} + \sum_{j=1}^{t} b_j \mathbf{v}_j$$

where b_j is the weighting for the j^{th} variation vector.

The formulation of the PDM can also be expressed in matrix form [Cootes 95]

Equation 3.2-4
$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}$$

where $\mathbf{P} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ is a matrix of the first *t* eigenvectors where $\mathbf{v}_i \in \Re^{2n}$ and $\mathbf{b} = (b_1, b_2, \ldots, b_t)^T$ is a vector of weights.

Chebyshev's theorem [Walpole 98] links the probability of the occurrence of data lying within the area of a normal distribution from the mean. This theorem is summarised by Table 3.2-1 [Elsayed 96] and demonstrates that there is a probability of .998 that the data will lie within three standard deviations of the mean. Principal Component Analysis makes the assumption that the training set is a multivariate Gaussian. As $\sqrt{\lambda_j} \approx \sigma_j$ (the standard deviation of the variance along $\mathbf{v_j}$), suitable limits for b_j are between $\pm 2.5\sqrt{\lambda_j}$ and $\pm 3\sqrt{\lambda_j}$, where λ_j is the j^{th} largest eigenvalue of **S**. Hence the multivariate Gaussian is bounded such that it encompass in excess of 98% of the deformation.

u	$P_{\bar{x},\bar{x}+u}$
0	0
0.5σ	0.192
σ	0.341
1.5σ	0.433
1.645σ	0.450
1.96σ	0.475
2σ	0.477
2.5σ	0.494
2.575σ	0.495
3σ	0.499

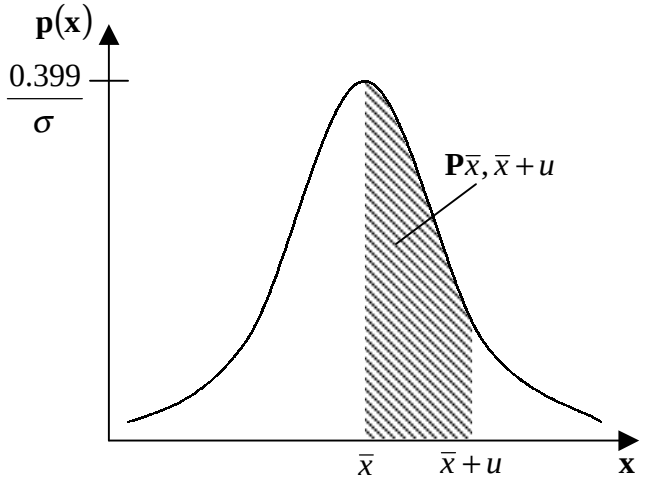


Table 3.2-1 - The area probability under a normalised Gaussian distribution

When high dimensional data sets are considered, eigenvector decomposition becomes a time consuming process, as the co-variance matrix is a square $2n \times 2n$ matrix for a *2n* dimensional data set. The memory requirements needed to store this matrix also become prohibitive as the size of the matrix approaches the size of a computer's physical memory. However, it is not always necessary to solve a matrix for all eigenvectors. If the number of training examples, *N*, is less than the dimensionality *2n*, the number of eigenvectors that can be extracted from the co-variance matrix cannot exceed the number training examples (*N-1*). For high dimensional problems, this is often the case and significant computational

benefits can be gained by solving for a smaller $N \times N$ matrix derived from the same data. If the covariance matrix,

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

is rewritten as

$$\mathbf{S} = \frac{1}{N}\mathbf{D}\mathbf{D}^T$$

where **D** is a $2n \times N$ matrix with the examples as columns.

Cootes *et al* demonstrated that if a new matrix T is a smaller $N \times N$ matrix

$$\mathbf{T} = \frac{1}{N}\mathbf{D}^T\mathbf{D}$$

and $\mathbf{e}_i$ *(i=1, ..., N)* are the unit, orthogonal eigenvectors of **T** with the corresponding eigenvalues γ_i:

$$\mathbf{T}\mathbf{e}_i = \gamma_i\mathbf{e}_i \quad \textit{(i=1, ..., N)}$$

then

$$\frac{1}{N}\mathbf{D}^T\mathbf{D}\mathbf{e}_i = \gamma_i\mathbf{e}_i$$

premultiplying by **D yields**

$$\frac{1}{N}\mathbf{D}\mathbf{D}^T\mathbf{D}\mathbf{e}_i = \gamma_i\mathbf{D}\mathbf{e}_i$$

and therefore

$$\mathbf{S}(\mathbf{D}\mathbf{e}_i) = \gamma_i(\mathbf{D}\mathbf{e}_i)$$

Thus if $\mathbf{e}_i$ is an eigenvector of **T**, then $\mathbf{D}\mathbf{e}_i$ is an eigenvector of **S** and has the same eigenvalue. The *N* unit orthogonal eigenvectors of **S** are then $\mathbf{v}_i$ *(i=1, ..., N)*, where

Equation 3.2-5 $$\mathbf{v}_i = \frac{1}{\sqrt{\gamma_i N}}\mathbf{D}\mathbf{e}_i$$

with corresponding eigenvalues $\lambda_i = \gamma_i$.

3.2.6 Human Head Example

To demonstrate the construction of a 2D PDM a model of a human head was constructed. Figure 3.2.4 shows the training set used to generate the model along with the source image from which the contour was extracted. The contour is selected such that it follows the high intensity edges of the face.

Figure 3.2.4 - Training Examples for 2D Head PDM

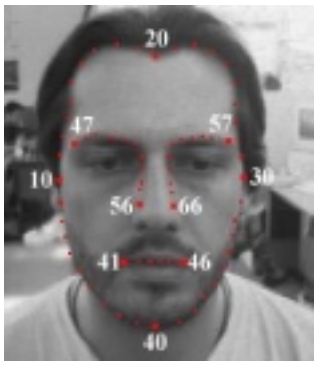


Figure 3.2.5 - Landmark points of the 2D Head PDM

Each 2D contour consists of 66 points (i.e. n=66), 40 for the external contour of the face, 6 for the mouth and 10 for each eyebrow. As each point is a 2D point in the image frame this generates an example $\mathbf{x} \in \Re^{2n} \Rightarrow \Re^{132}$. After the training set has been aligned, PCA is performed to extract the primary modes of deformation i.e. the eigenvectors. The eigenvalues provide bounds for the deformation along any mode or eigenvector as previously discussed, but by analysing the eigenvalues further the true dimensionality of the model can be determined.

Figure 3.2.6 shows the normalised eigenvalues sorted into descending order. As there are 9 training examples, this results in 8 eigenvectors (i.e. *N-1* modes, where *N=9*). The larger the eigenvalue the more significant the corresponding eigenvector or mode of variation. As the number of the mode increases, so the significance of the mode decreases. By analysing these eigenvalues, the linear subspace on which the data lies can be determined and the information loss estimated. The use of this technique is discussed further in section 5.3.

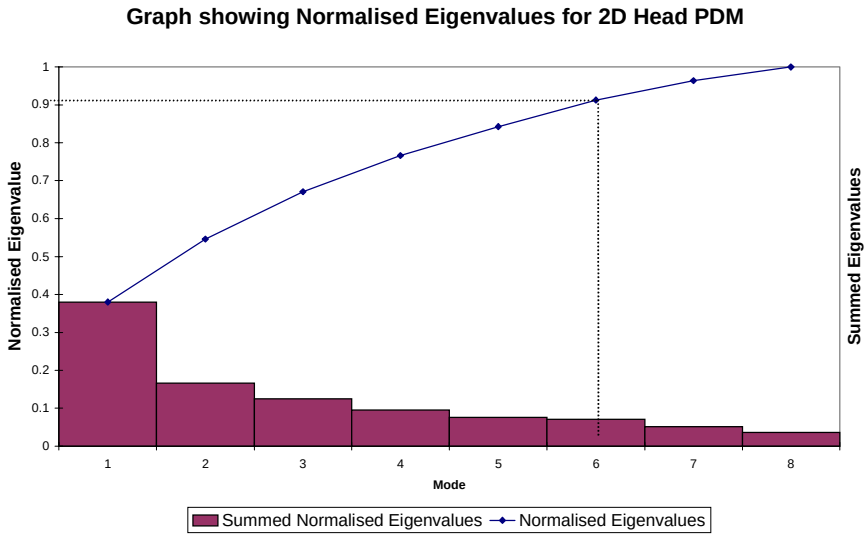


Figure 3.2.6 - Graph showing Normalised Eigenvalues for the 2D Head PDM

Figure 3.2.6 also shows the sum of the normalised eigenvalues. As the number of modes increase this sum of the normalised eigenvalues approaches 1. If this is converted into a percentile, it provides an indication of the amount of deformation contained within the accumulated modes. The combination of all 8 modes results in a sum of 1 or 100%. Therefore using all 8 modes of deformation, the model is capable of representing 100% of the deformation in the training set. It can be seen that the primary mode alone accounts for 40% of the deformation represented within the training set. It can further be seen that the 90% of the deformation is contained within the first 6 modes. If the loss of 10% of deformation is tolerable then the data can be said to lie upon a six dimensional space and not 122 as originally formulated. This provides a dimensional reduction of 122 to 6 and will be discussed further in section 5.3.

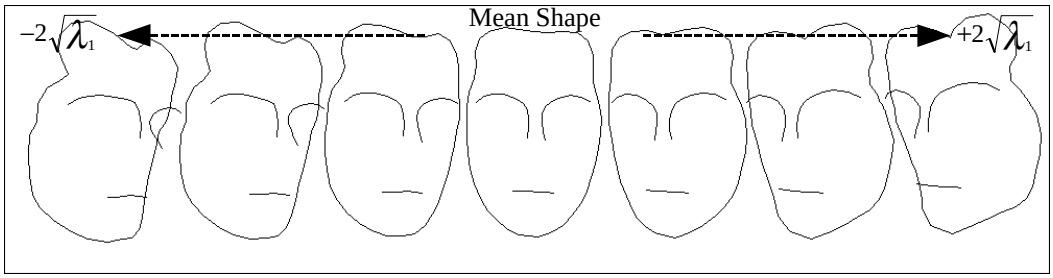


Figure 3.2.7 - Primary mode of the 2D Head PDM

Figure 3.2.7 shows the primary mode of variation drawn at intervals along the primary axis from $\pm 2\sqrt{\lambda_1}$ from the mean. This primary mode has clearly picked out the turning motion of the head. The model has generalised the training set and learnt what is typical deformation for the object. By applying different weighting combinations of b_j to Equation 3.2-3 new examples of the face under deformation can be generated.

3.3 Active Shape Models

3.3.1 Overview

The Point Distribution Model contains the constraints on deformation for a model class that has been learnt from a training set of examples. Cootes *et al* describe Active Shape Models (ASMs) as the application of this deformable model (PDM) to tracking objects within the image frame. In order to facilitate this, the object must be able to 'move' in addition to deform within the image. For a 2D contour, this movement consists of a translation, scale and rotation. Assuming a constant scaling in *x* and *y* this generates four parameters which position and orient the model within the image frame, where an instance **X** of the model is given by

$$\mathbf{X} = M(s,\theta)[\mathbf{x}] + \mathbf{X}_c \text{, where}$$

$$\mathbf{X}_c = (x_c, y_c, x_c, y_c, \ldots, x_c, y_c)^T$$

$M(s,\theta)$ is a rotation by θ and a scaling by *s*, and (x_c,y_c) is the position of the centre of the model in the image frame.

The ASM assumes that the next pose of the model **X',** will be a small variation on **X** (the initial pose) and requires that **X** be close to the desired feature. The model is then iteratively refined by calculating a new pose for the model **X'** by adjusting s, θ, x_c, y_c and the deformation parameters **b** in order to find the closest pose to the desired model in a least squares sense.

Throughout the course of this text the term least squares gradient descent tracking will be used to describe the common ASM tracking algorithm.

The ASM tracking algorithm can be summarised as

1. Initialise a model **X,** close to a desired feature in the image frame.
2. While still tracking,
    3. Using a local feature detection scheme assesses the next best movement of the model **X'**.
    4. Update the parameters s, θ, x_c, y_c to minimise the distance between **X** and **X'**.
    5. Update the shape parameter weightings **b** to mimise the distance within the constraints of the model.

Each of these steps will now be considered in turn.

3.3.2 ASM Initialisation

Due to the local search method used when deforming the contour (see next section) and the least squares parameter approximation, it is important that the initial contour is placed close to the desired feature. Hill et al described how a Genetic Algorithm (GA) search can be used to facilitate this [Hill 92a][Hill 92b]. Cootes et al have also demonstrated how multi-scale approaches to image searching can be used to reduce this susceptibility to model initialisation and providing more robust tracking [Cootes 98]. However, given an object of a specific class, other indicative features can be used to initialise the model. As these features are only required for initialisation or re-initialisation when the contour is lost, the computational complexity of such strategies is less important.

In chapter 9 it will be demonstrated how colour features, such as those discussed in chapter 4, can be used to initialise a model within the image frame.

3.3.3 Feature Detection

A PDM which consist of a 2D contour, typically represents the edges of an object within an image. An edge is a high rate of change in pixel intensity and edge detection algorithms are commonplace in image processing [Ballard 92; Russ 94]. However, as only a local search of the image is necessary and edges must be perpendicular to the contour, hence normal convolution methods are not necessary.

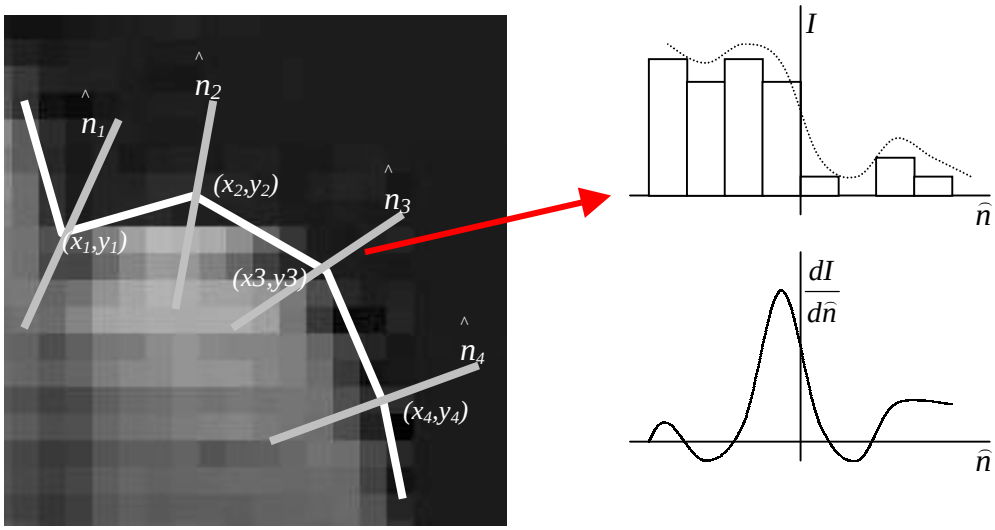


Figure 3.3.1 - Local edge detection along boundary normals

Figure 3.3.1 demonstrates a contour within a grey scale image with four key points along the boundary. The intensities along the normal $\hat{n}_3$ are shown in the histogram along with the continuous approximation to this data and the first derivative. The peak of this first derivative provides a position along the normal for the best fit edge. Once found, the control point can be moved to this new location.

If point along the contour P_m is denoted by $\vec{P}_m(x_m, y_m)$, where (x_m,y_m) is the pixel in the image frame, the normal of the contour can be estimated as

$\vec{n}_m\left(\frac{(y_{m+1}-y_m)+(y_m-y_{m-1})}{2},-\frac{(x_{m+1}-x_m)+(x_m-x_{m-1})}{2}\right)$ which can be rewritten as $\vec{n}_m\left(\frac{(y_{m+1}-y_{m-1})}{2},-\frac{(x_{m+1}-x_{m-1})}{2}\right)$

The unit vector normal $\hat{n}_m = \frac{\vec{n}_m}{\|\vec{n}_m\|}$ is therefore a one pixel length vector perpendicular to the contour at point *m*. Using this locally estimated normal, the intensity of pixels either side of the contour can be examined and any high intensity gradients (edges) located.

As the contour is designed to lie tangential to the high intensity edges within the image a 2D convolution is not necessary. Therefore, only the contour normal need be searched. This localised search provides a large computational saving over other convolution based methods such as the original formulation of the snake where an entire gradient image is pre-computed [Kass 87]. This also demonstrates the applicability of the colour enhancement approaches described in chapter 4, as they can be used without a significant computational overhead.

A pixel's intensity gradient along a 1D line can be estimated using a number of schemes. The simplest is possibly the local difference in intensity $dI_i = I_i - I_{i-1}$, where I is the intensity of a pixel. A 2nd derivative 1D Laplacian function $d^2I_i = dI_{i+1} - dI_i = I_{i+1} - 2I_i + I_{i-1}$ (which has a zero crossing value) provides an indication of a strong edge when $d^2I_i = 0$, or more realistically when $\min_i\left(\left[d^2I_i\right]^2\right)$. However, these methods are susceptible to noise and best results have been achieved using a 1D Gaussian derivative kernal which both smooths (blurs) in addition to detecting edges where

$$Gaussian_i = I_{i-1} + 4I_{i-2} + 5I_{i-1} - 5I_{i+1} - 4I_{i+2} - I_{i+3}$$

The *best* edge along a normal, and hence the movement for a point P_m upon a contour can therefore be estimated as

$$P'_m = P_m + \hat{n}_m \times w, \text{ where } w = \arg\max_{i=-l}^{l}\left(Gaussian_{P_m+\hat{n}_m\times i}\right)$$

Once a new position P'_m has been located for each point m along the contour, a new vector representing the model **X'** is constructed by concatenating the points into a vector as done earlier. This provides a new (noisy) shape vector where each contour point has been moved to its best match edge location where

$$\mathbf{X}' = \mathbf{X} + d\mathbf{X}$$

In order to calculate the constraints on the shape of the object, the contour must be transformed into the PCA co-ordinate space. In doing this the parameters (s, θ, x_c, $y_{c)}$ which provide the mapping from the model space to image space are derived.

3.3.4 Iterative Refinement

Once a model has been initialised in the image frame, the model need only make small iterative refinements to its shape and position between frames. Providing a high frame rate can be achieved (and hence this assumption holding true), local search techniques can be used to reduce the computational complexity of model tracking.

The parameters x_c, y_c are first calculated by finding the centeroid of the new contour **X'**,

$$x_c = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x'_i$$

$$y_c = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y'_i \text{, where } \mathbf{X}' = (x'_1, y'_1, x'_2, y'_2, \ldots, x'_n, y'_n)$$

therefore the mean point of the contour is equivalent to the contour position in the image frame where

Equation 3.3-1 $$\mathbf{X}_c = (x_c, y_c, x_c, y_c, \ldots, x_c, y_c)^T$$

The rotational parameter $d\theta$ is calculated by taking the average dot product of contour points $P_i'(x_i, y_i)$ with the model contour points $P_j(x_j, y_j)$.

Using $\|v_1\|\|v_2\|\cos\theta = v_1 \bullet v_2$

Equation 3.3-2 $$d\theta = \cos^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{(P_i - P_c)}{\|P_i - P_c\|}\cdot\frac{(P_i' - P_c')}{\|P_i' - P_c'\|}\right)$$

The scaling parameter ds is calculated by taking the average difference of the length of the contour from the centeroid between iterations.

Equation 3.3-3 $$ds = \frac{1}{n}\sum_{i=1}^{n}\left(\|P_i - P_c\| - \|P_i' - P_c'\|\right)$$

This can be performed in both x and y separately to allow shearing of the contour.

This 'noisy' contour is then transformed into the PCA space and the residual movements of the contour points, $d\mathbf{x}$, calculated where

Equation 3.3-4

$$d\mathbf{x} = M((s(1+ds))^{-1}, -(\theta + d\theta))[M(s,\theta)[\mathbf{x}] + d\mathbf{X} - d\mathbf{X}_c] - \mathbf{x}$$

As all rotation, scaling and translation has now been removed, the residual movements, $d\mathbf{x}$, can only be resolved by deforming the model. This is done by projecting the residuals into the PDM and finding the set of weightings which provide the closest 'allowable' point in space to $d\mathbf{x}$.

From Equation 3.2-4

$$\mathbf{x} + d\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{P}(\mathbf{b} + d\mathbf{b})$$

therefore

$$d\mathbf{b} = \mathbf{P}^{-1} d\mathbf{x}$$

or $d\mathbf{b} = \mathbf{P}^T d\mathbf{x}$ since $\mathbf{P}^T \equiv \mathbf{P}^{-1}$, as the columns of $\mathbf{P}$ are mutually orthogonal and of unit length [Cootes 95].

The weighting vector is then adjusted to ensure that each parameter lies within the range learnt during PCA where

$$\mathbf{b}' = \mathbf{b} + d\mathbf{b}\text{, and } -3\sqrt{\lambda_i} \le b_i \le 3\sqrt{\lambda_i}$$

The procedure then repeats using these new parameters for the next iteration.

3.4 Reconstructive Ability

The PDM *learns* shape space and in doing so generalises what is valid deformation, allowing valid unseen data to be reproduced in addition to the original training examples. Figure 3.4.1 shows a PDM of the hand tracking a real hand within the image. In this figure the first finger has been bent, however, the model remains with the finger extended. This is due to the fact that during construction no examples were provided in the training set that represented this type of deformation of the model. As no deformation is learnt the model is constrained to the extended pose. These constraints on shape provide a robust model for tracking where occlusion or clutter is present. If part of the hand is obscured the model will fill in the missing contour as the deformation of all points are statistically linked together.

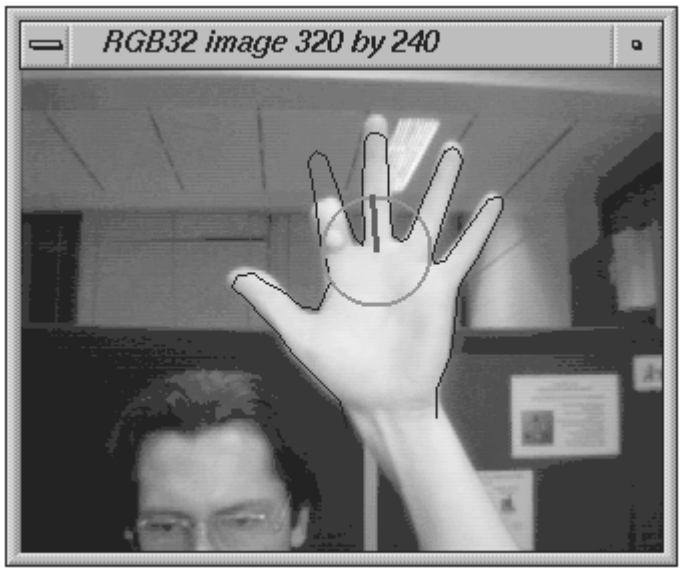


Figure 3.4.1 - Constrained PDM tracking hand

To illustrate the reconstructive ability of the PDM a sample training set was constructed which consisted of examples of leaf. Each leaf was segmented from images using a colour threshold and boundary-tracing algorithm. The contour was aligned as described in section 3.2.4 and four landmark points identified at the horizontal and vertical extremities of the boundary. Further points were then introduced at regular intervals between the landmarks. Before PCA is performed all shape vectors are normalised to provide numerical stability. The resulting PDM is shown in Figure 3.4.2. After PCA, 99.9% of the deformation contained in the training set is encompassed by the 44 eigenvectors corresponding to the 44 largest eigenvalues. Figure 3.4.2 show the primary 5 modes of variation, which corresponds to the 5 largest eigenvalues after PCA. The centre shape shows the mean, and the deformation from left to right shows the effect of each mode of variation.

It can be seen that the 1st mode of deformation encompasses the horizontal size of the shape, i.e. how elongated the leaf is. The 2nd mode is partly responsible for the curvature and size of the sample at its extremities, through their combination all training leaf samples can be reconstructed.

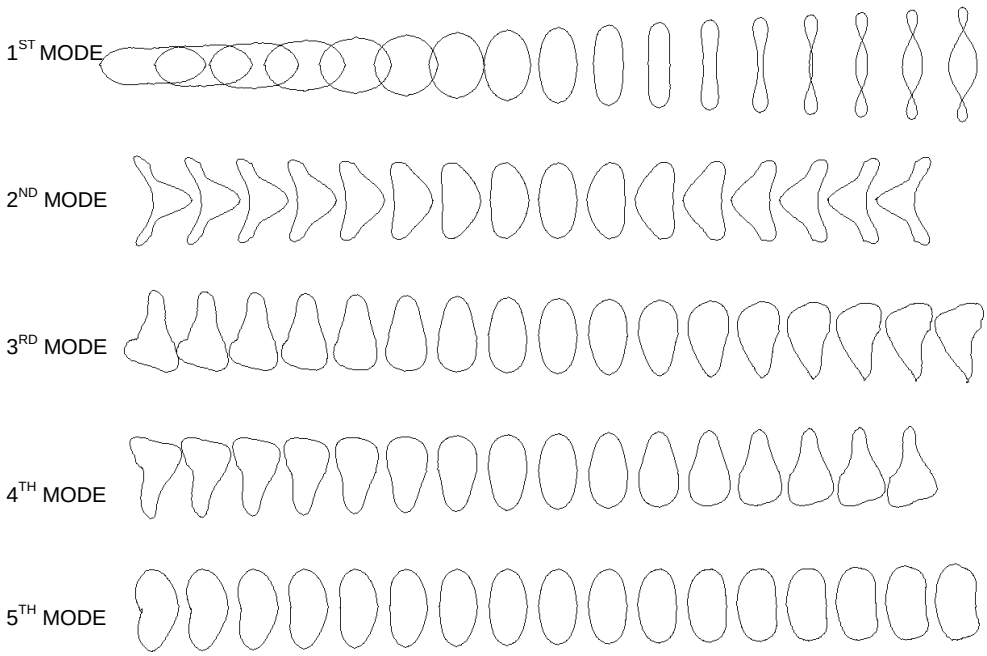


Figure 3.4.2 - First Five Modes of variation of the leaf PDM

Using the primary 44 modes of variation the accurate reconstruction of shape is possible. However, this is more information than is required for the purposes of the investigation. By reducing the number of modes further, two objectives are achieved. Firstly, the size of the model is reduced. Secondly, only the major deformations of shape are modelled and the finer deformation disregarded, i.e. the shape is smoothed while retaining the important information.

Figure 3.4.3 shows the results of using only the first nine modes of variation to reconstruct the shape. Notice that although the overall shape of the leaf is preserved the model is considerably smoothed.

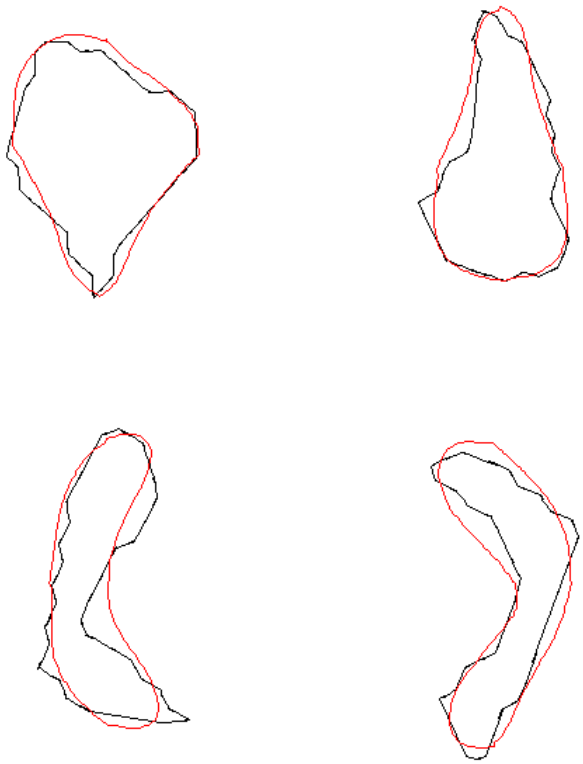
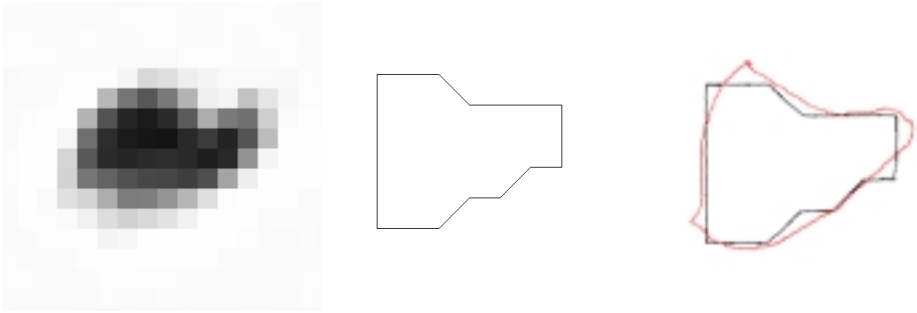


Figure 3.4.3 - Training examples and the reconstructed shape using 9 modes of variation

Although this smoothing is a lossy compression technique, the information that is discarded is of little use. This is due to small leaf samples where their extraction resulted in blobs of the order of tens of pixels rather than hundreds. The resulting boundary is heavily ‘step-like’ due to the pixelisation of the shape. During re-sampling, bilinear interpolation results in the boundary being

smoothed into unrepresentative shapes which are indicative of the modality used, and not the actual leaf sample. By using the minimum number of modes to reconstruct the shape, the errors introduced into the shape by the image size are discarded and a better estimation of shape provided. Figure 3.4.4 shows a small leaf sample, with the interpolated/resampled boundary and the resulting smoothing which comes from PDM reconstruction. It should be noted that the smoothed boundary produced by the PDM goes some way to reconstructing the information lost during acquisition. This is due to the statistical nature of the PDM and its knowledge of what a leaf *'should look like'*.



(a) *(b)* *(c)*

Figure 3.4.4 - Training examples and the reconstructed shape using 9 modes
(a) Original Image of leaf (b) resampled boundary of leaf (c) reconstructed boundary of leaf

3.5 Conclusions

The statistical constraints of the PDM provide several benefits over other model-based approaches. Firstly, the model is taught to fit known objects and deformations even when slightly different from those present within the training set. However, it does not allow deformation for unseen/unfamiliar objects i.e. it generalises shape. Secondly, the mean distance of constrained contour points to detected/desired edges can be used as a valuable error metric for model fitting. The constraints provide robustness to noisy, partially occluded object boundaries as well as background clutter and lastly the constraints allow the contour to statistically infer contour shape in the absence of local information from other available information.

Chapter 4

4 Enhancing Tracking Using Colour

4.1 Introduction

The colour content of an image is an important attribute, which is often discarded. Common practice in the processing of PDMs and snakes is to merely assess the intensity of pixels, processing as if grey scale i.e. calculating the mean intensity of the red, green and blue colour channels.

This chapter will discuss how colour can be used to enhance the appearance of objects in tracking algorithms. It will also be demonstrated how colour alone can provide a reliable feature for locating and tracking moving objects. Section 4.2 will demonstrate how the simple weighting of colour channels can be used to enhance specific features within an image. Section 4.3 will discuss the use of perceptual colour representations (alternative colour spaces to red-green-blue, RGB). Section 4.4 will discuss the advantage of colour in delineating regions. Section 4.5 shows how more complex colour models can be constructed and used to locate and track a humans. Section 4.6 demonstrates how these ideas can be extended to provide a reliable, computationally inexpensive solution to head and hand tracking, although these techniques extend to any colour object. Finally conclusions are presented.

4.2 Weighted Greyscale Images

In the previous chapter it was shown how high intensity edges could be located locally along a boundary. These high rates of change in pixel intensity were located by assessing the first or second derivative of the intensity along a normal to a boundary. This calculation is normally performed upon the grey scale values of pixels. However, as has already been mentioned, the ready availability of colour provides a far more distinguishable difference between foreground and background objects within an image. By performing processing upon a grey scale representation, calculated from the colour channels (typically the average intensity of the three colour channels) a considerable amount of information about object boundaries is lost.

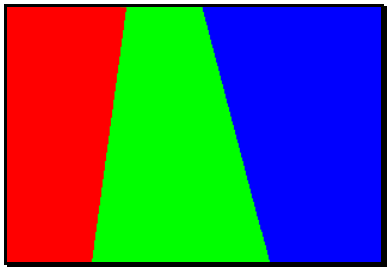


Figure 4.2.1- RGB image of iso-intensity

Figure 4.2.1 shows an image consisting of three colour regions. Each region has the same intensity in its colour channel: the red area has r=255, g=0, and b=0; the green area has r=0, g=255 and b=0; etc. By taking the average of the three colour channels at each pixel, the resulting image would have a constant intensity of 85 and no distinction would be possible between the various areas. However, in the colour image, it is visually apparent that such a distinction does exist and very clear boundaries are defined.

It is clear that reducing the colour information to one channel literally 'throws' information away, information which may be invaluable to the application at hand. One solution to this would be to process each colour channel individually. This can be done by assessing normals for each colour in turn, calculating three second order derivatives, and taking the average, where

$$d^2 I_i = \frac{R_{i+1} - 2R_i + R_{i-1} + G_{i+1} - 2G_i + G_{i-1} + B_{i+1} - 2B_i + B_{i-1}}{3}$$

However, this is still an averaging approach and as such will smooth edges. In addition, the approach effectively requires each normal to be assessed three times and hence results in a significant decrease in speed.

If an object of interest is sufficiently prominent within one of the colour channels, then the intensity of that channel can be used instead of the mean intensity.

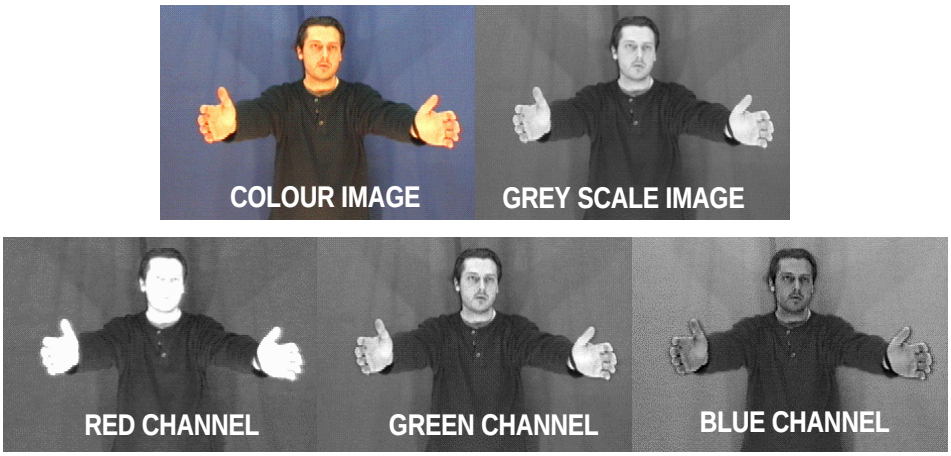


Figure 4.2.2 - The Separate Channels of a Colour Image

Figure 4.2.2 shows a colour image of a person in front of a blue backdrop, along with the grey scale version of the image and the three separate colour channels shown as grey scale intensity images. The grey scale image retains much of the distinctions between regions seen in the colour image due to the small number of highly distinct regions and the uniform background. The individual colour channels, however, each emphasise certain aspects of the image. The blue channel has a lighter background than red or green with a lower contrast figure. This is to be expected, as the blue background will generate high intensities in the blue channel. The red channel emphasises the skin regions of the subject, due to the high red component in skin tones. If the object to be located or tracked within the image were hands or head then using the red channel for image processing would produce far superior results than tracking on the mean intensity

(as the mean intensity effectively smoothes out this distinction). However, simply processing upon the red channel may disregard other important features. In addition, other channels could potentially be used to subdue features that are not desirable, i.e. the background. As it is known that the background is depicted best in the blue channel, subtracting this from the red channel will further increase the distinction between regions.

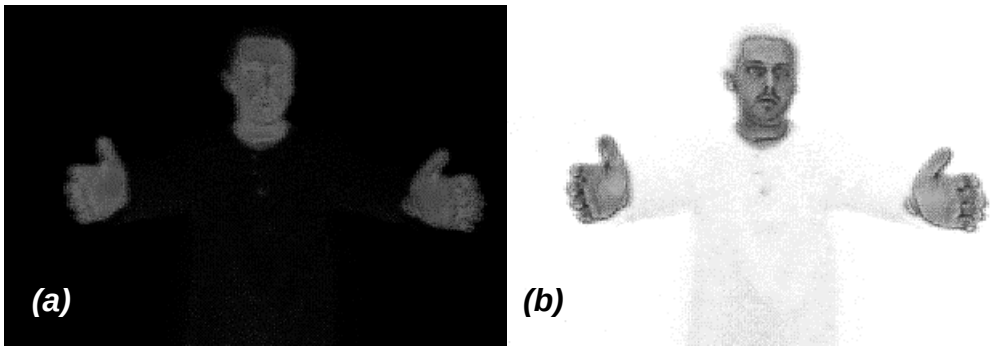


Figure 4.2.3 - Enhancing features Using Colour Channels

(a) Blue channel subtracted from red (b) Inverse of (a)

Figure 4.2.3(a) demonstrates the results of subtracting the blue channel from the red channel. Figure 4.2.3(b) shows the inverse of (a), which improves the visualisation of the distinction between regions. Although the overall contrast between skin and the surrounding area appears less, the segmentation of the skin from the overall image is greatly enhanced. The background and body have almost completely been removed.

If the simple conversion to grey scale is formulated as the average pixel intensity of the three colour channels, this can be expressed as

$$I_{x,y} = \frac{r_{x,y} + g_{x,y} + b_{x,y}}{3}$$

then subtracting the blue channel from red can be expressed as the weighted average of the pixels,

$$I_{x,y} = \frac{\alpha r_{x,y} + \beta g_{x,y} + \chi b_{x,y}}{\max\left(1, \left|\alpha + \beta + \chi\right|\right)}$$

where

$$\alpha = 1, \beta = 0, \chi = -1$$

by tailoring these colour coefficients for specific applications, features can be enhanced or subdued as required. Figure 4.2.4 shows the results of further enhancing the skin regions by applying the coefficients $\alpha = -2, \beta = 0, \chi = 2$.

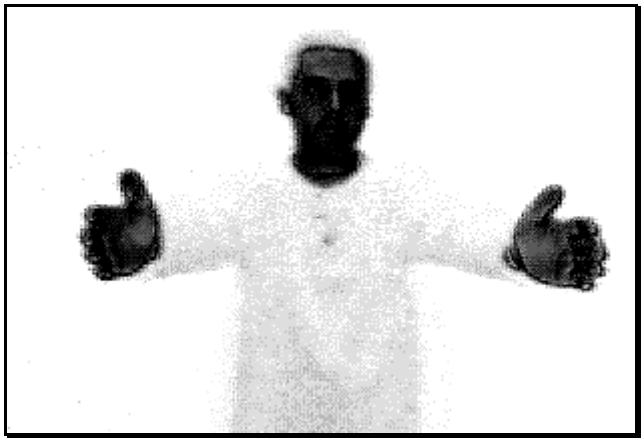


Figure 4.2.4 - Enhancing features Using Colour Channels

4.3 Perceptual Colour Spaces

The RGB-colour space (typically used in computer applications) allows three primary colour channels to be used to specify up to 16.7 million colours by representing the colour space as a 3D-colour cube (each channel having 256 discrete intervals). This provides a simple mechanism for constructing and representing a broad spectrum of colours. However, this is not an intuitive representation in terms of human perception, where similar colours (as judged by the eye) may occupy completely different areas of rgb-space. This is confirmed by the initial observations made from Figure 4.2.1. It has already been noted that the intensity of each colour region has the same value, even through the distinction between the areas is visually apparent. Furthermore, the central green region looks brighter to the human eye than either the red or blue regions. The notion of a perceptual colour space is to model the colour volume so to better correspond with how the human eye perceives colour and relative intensities.

Discussions of colour perception usually involve three quantities, known as *hue, saturation* and *lightness*. *Hue* distinguishes among colours such as red, green and purple. *Saturation* refers to how far colour is from a grey of equal intensity, i.e. red is highly saturated, pink is not, although both have similar hue/red-component. *Lightness* embodies the achromatic notion of the perceived intensity of an object. These perceptual colour spaces include Hue, Saturation, Value (HSV) (or HSB for Brightness); Hue, Lightness and Saturation, (HLS) (or *HSL* for Luminosity); and Hue, Value, Chroma, HVC [Foley 1990].

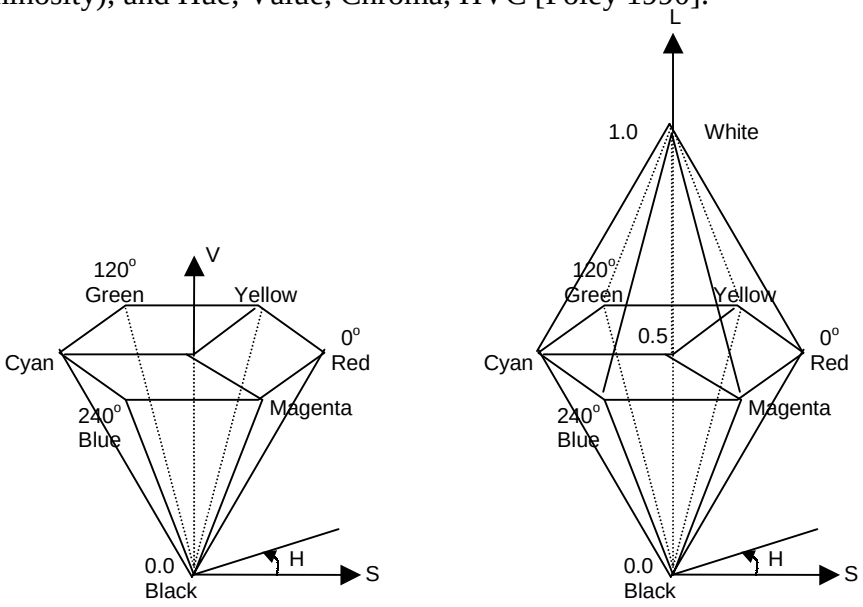


Figure 4.3.1 - HSV and HLS Colour Spaces

Hue **S**aturation **V**alue (HSV or HSB) colour space is a hexcone or six sided pyramid where Hue is the angle around the vertical axis, S is the distance from the central axis and V is the distance along the vertical axis. Colours along the vertical axis have zero saturation and are therefore grey scale values. **H**ue, **L**ightness **S**aturation (HLS or HSL) colour space is a double hexcone and can be thought of as a deformation of the HSV space.

The notion of separating colour from intensity provides a more robust method for colour feature extraction. Where colours change from shading or lighting differences, it would be expected that this would result in changes in intensity but not in colour.

Figure 4.3.2 – Separate Channels of HSL Image

Figure 4.3.2 shows the same colour image from section 4.2 converted in *hue, saturation* and *luminosity* with each channel shown as an intensity image. It can clearly be seen that the difference between the areas of the image is far more distinct in both hue and saturation than in any of the rgb colour channels (Figure 4.2.2). The saturation image provides excellent segmentation between the skin and other areas of the image frame, producing a distinct boundary between the skin and background elements.

Some devices provide colour space conversions in hardware. However, for the most part this must be implemented in software. For real-time systems where each pixel must be transformed independently, this overhead can become a significant speed-limiting factor. However, with contour based approaches this conversion does not produce a significant overhead, as only pixels along normals to the contour are assessed and hence need conversion.

A similar coefficient weighted expression to that demonstrated for rgb space can be used in HSL space, where

$$I_{x,y} = \frac{\alpha h_{x,y} + \beta s_{x,y} + \chi l_{x,y}}{\max\left(1, \left|h+s+l\right|\right)}$$

Provided hsl values are normalised to the range $0 \rightarrow 1$.

Further extensions can be made by combining both RGB and HSL weighted techniques. However, coefficient selection becomes a complex task. Instead, a more generic, automated method of enhancing/extracting features is required.

4.4 Colour Thresholding

As was demonstrated in the previous section, areas of skin produce high values in the saturation channel of the HSL colour image (Figure 4.3.2). These high areas can be used to threshold the areas of skin from the image in a similar manner to grey level thresholding. This technique is not dissimilar to chroma/luma keying.

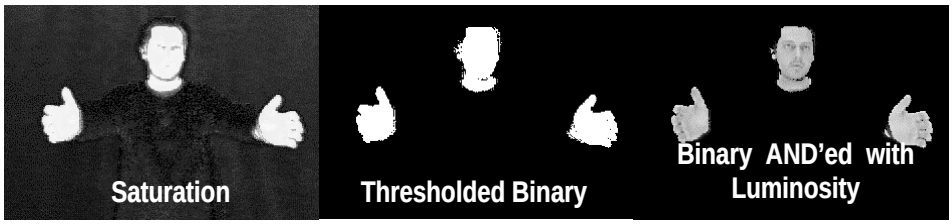


Figure 4.4.1 – Thresholded HSL Image

Figure 4.4.1 shows the saturation channel of the colour image. As the areas of skin produce high values of saturation, these areas can be extracted simply by thresholding the colour saturation channel into a binary image mask. The white-segmented areas correspond to the location of skin within the mask. Figure 4.4.1 shows the results of taking the logical AND of the binary image with the luminosity channel and demonstrates how the head and hands can be extracted using colour saturation instead of intensity to delineate colour regions of the image while retaining the internal features of objects or regions.

It should be noted that although the head and hands consist of various colour changes due to the features such as eyes, nose and the effects of non-diffused lighting, few of these features are apparent (to the eye) in hue or saturation. This is due to the separation of the *colour* information from the *brightness or luminosity*. The *luminosity* contains the information of how bright a pixel is and the hue-saturation *h-s* pair provides the information about colour. Rather than performing thresholding in $\Re^3$ of *rgb*, it can be performed in $\Re^2$ of *h-s* space. This provides a slight computational saving but has the added advantage that with the intensity component removed, much of the lighting/shading differences are absent. This provides a more uniform colour space in which to work.

Discarding the *luminosity* component of the colour effectively compresses the *hsl* colour space down onto a two-dimensional hexagon. In this space, consistent colours of varying *luminance* will produce clusters on the *h-s* hexagon. By discarding the luminosity for HLS and the value component of HSV spaces, both spaces become compressed onto the two-dimensional hexagon and the distinction between the two spaces is lost.

4.5 Gaussian Colour Models

For a number of years, research at the School of Computer Science, Carnegie Mellon University has used normalised rgb colour spaces to probabilistically label and segment regions of skin from image sequences for the location and tracking of the human face [Waibel 94] [Hunke 94] [Yang 98]. They have demonstrated that human skin clusters in a small region of colour space: Human skin colours differ more in intensity than actual colour, and under certain lighting conditions, a skin colour distribution can be characterised by a multivariate normal distribution in a normalised colour space [Yang 95]. Rainer, Stiefelhagen and Yang use this colour labelling to provide a rough estimate of the location of a head within the image frame to initialise a model based gaze tracking system [Stiefelhagen 97] [Stiefelhagen 98]. The normalisation of the colour space removes much of the variability in skin colour between individuals and lighting inconsistencies such as shadows [Yang 98]. Ivins and Porril used a normalised rgb colour space to label and track, in real-time, various colour regions of an industrial robot arm [Ivins 98].

McKenna, Gong and Raja have extended this work on colour labelling into the HSV colour space [McKenna 97]. Using a Gaussian mixture model to represent the colour space, they have shown how multiple models for individuals can be used to probabilistically label an image and determine the most likely person present. Azarbayejani and Pentland have used similar methods in HSV colour space to automatically segment both the hands and head from stereo image pairs, and using this, calculate the position and trajectory in 3D space [Azarbayejani 96].

Work by these authors has shown that human skin naturally clusters in a small region in colour space. Hunke and Waibel show that in a normalised rgb colour space, statistical bounds can be approximated for colour clusters and used to segment the human head from an image [Hunke 94]. Using colour as a feature for tracking has several problems: firstly, the colour representation of a face obtained by a camera is influenced by many factors such as ambient light, object movement, and the effect of diffused and specular reflections of an object moving relative to a light source. Secondly, different cameras produce significantly different intensity responses for the same wavelength of light. Thirdly, video signal encoding standards, such as PAL or NTSC, do not respond to the full colour space and effectively flatten the resulting colour spectrums of objects. Finally, human skin colours differ in rgb space from person to person [Yang 98]. McKenna *et al* demonstrated how these problems could be partially overcome by performing probabilistic classification in HS space, where variations in intensity have been removed [McKenna 97].

Human skin actually occupies a small cluster in HS space regardless of race or skin pigmentation. Differences in skin tone are primarily expressed by variation in the intensity of the colour: once the intensity has been removed the *h-s* colour space that they occupy is remarkably similar.

In order to verify this fact, four subjects were taken from different ethnic origins. For each subject, pixels were sampled in *rgb* from the skin tones on the palm of the hand. The results can be seen in the two graphs shown in Figure 4.5.1 and Figure 4.5.2. These two graphs allow the visualisation of the volume of the rgb colour cube in which the samples lie. It is clear that a fairly distinct single cluster is generated by the samples. However, this sample occupies a relatively large sub-volume of the total colour space. This is due to the difference in intensity of the samples along its major axis i.e. the variation in intensity of the pixels across any one sample.

Each sample pixel was then converted into HSL space, the luminosity discarded and the results shown in Figure 4.5.3. The Hue-Saturation space shows a far 'tighter' cluster with little variation in either hue or saturation. It is also important

to note that this colour 'fingerprint' of human skin is now 2 dimensional rather than the original 3D-rgb space.

The large number of sampled pixels and similarity in each of the four ethnic skin types makes the comparison of each difficult. To simplify, the mean and standard deviations in each colour channel can be calculated by

$$\bar{r} = \frac{1}{n}\sum_{i=0}^{n} r_i \quad \text{and} \quad \sigma_r = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \bar{r})^2}$$

Figure 4.5.4 demonstrates the colours generated for the skin of four subjects with varying racial origin and pigmentation.

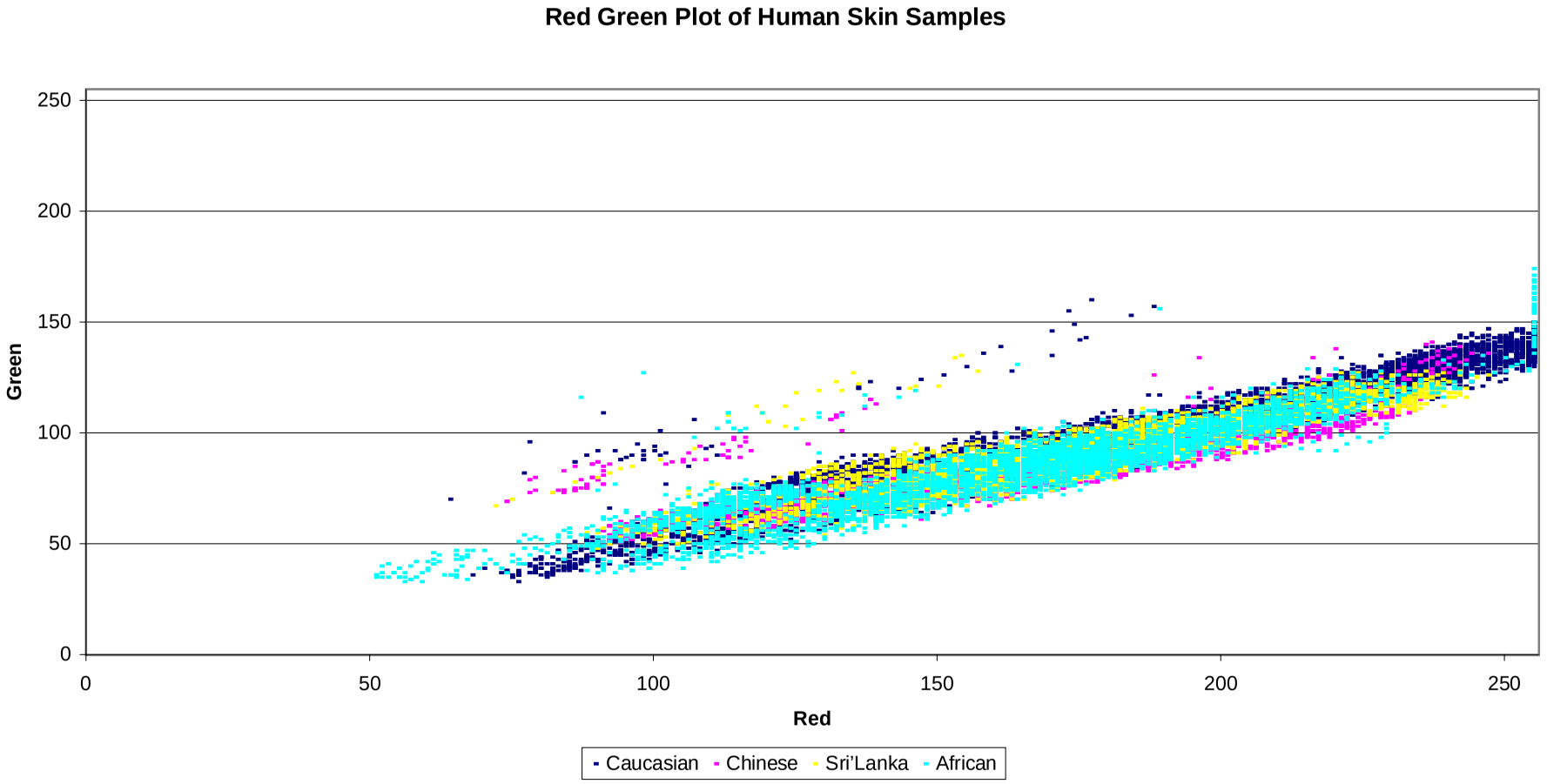


Figure 4.5.1 - Human Skin Samples Plotted in Red Green Space

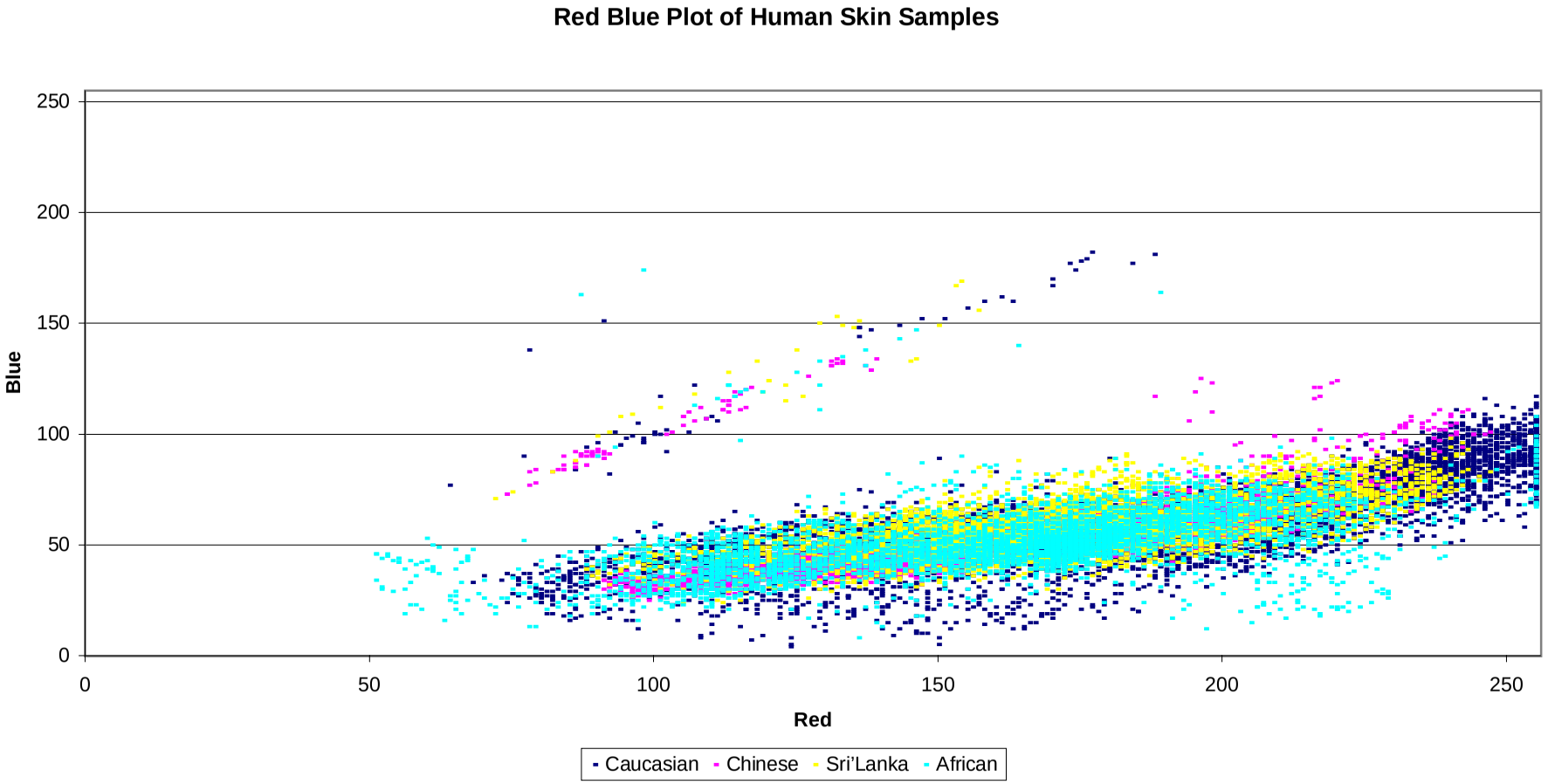


Figure 4.5.2 - Human Skin Samples Plotted in Red Blue Space

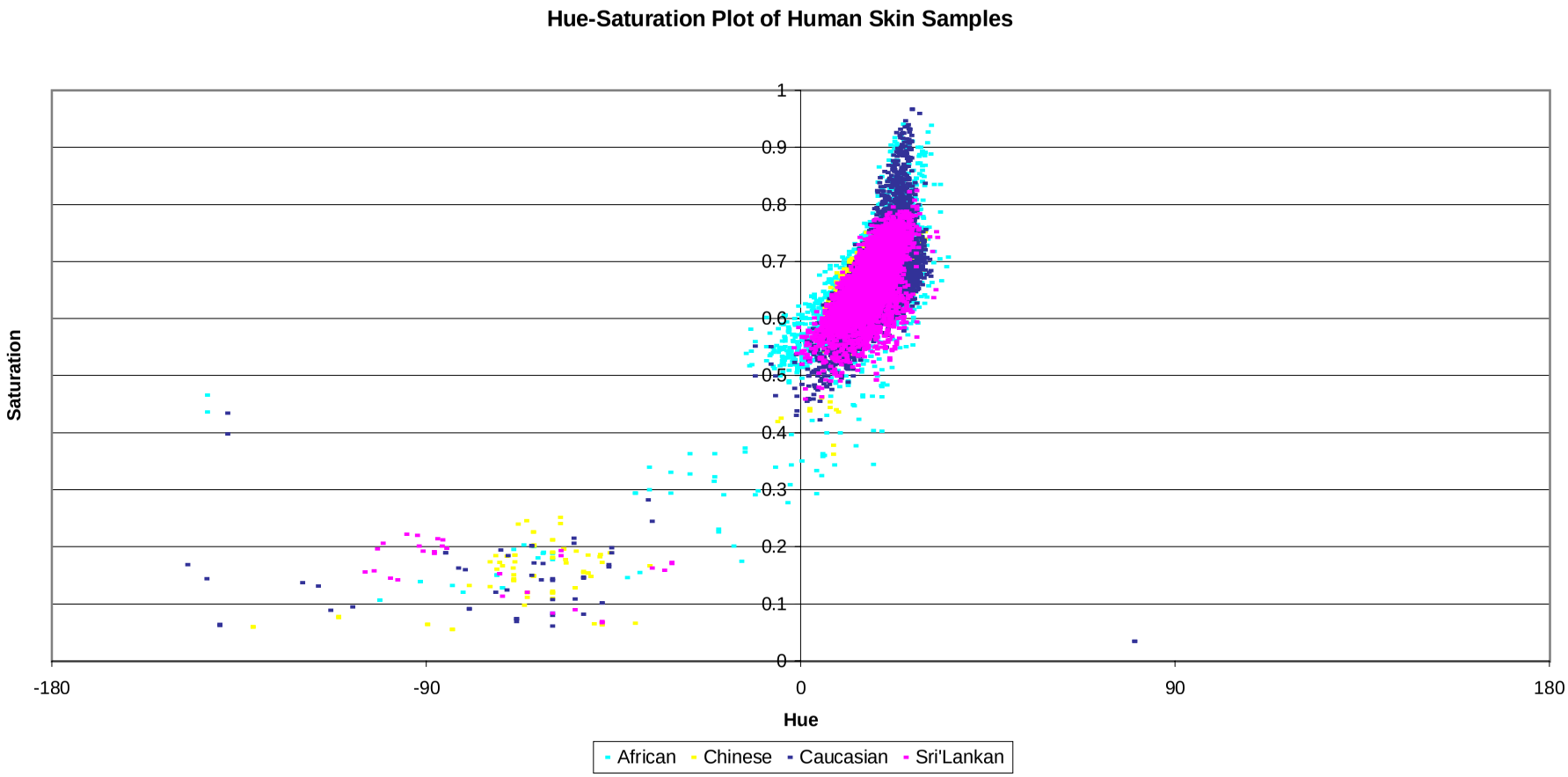
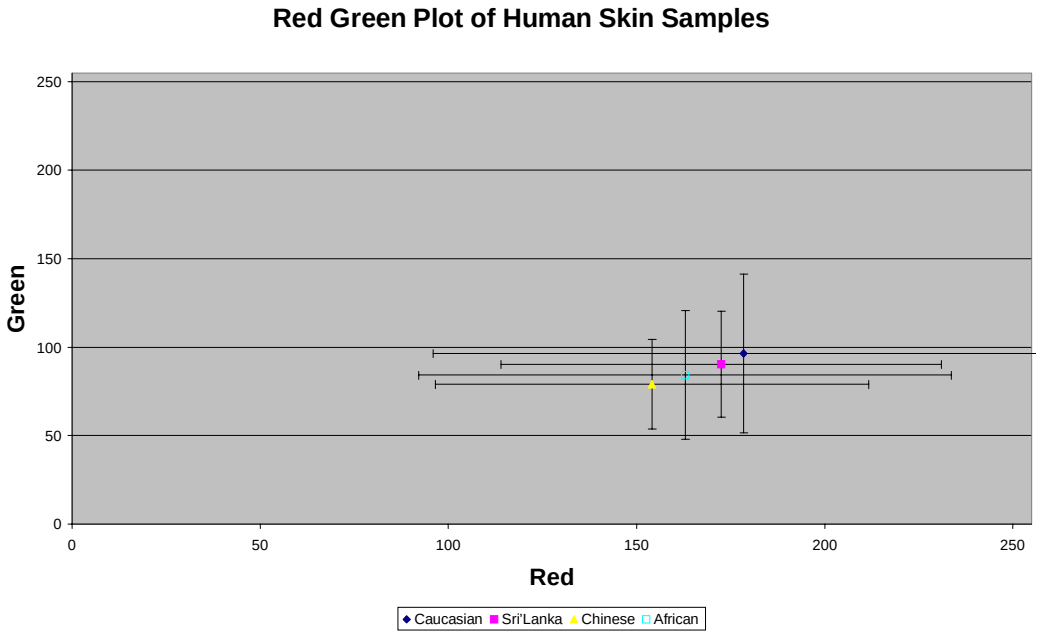


Figure 4.5.3 - Human Skin Plotted in Hue Saturation Space



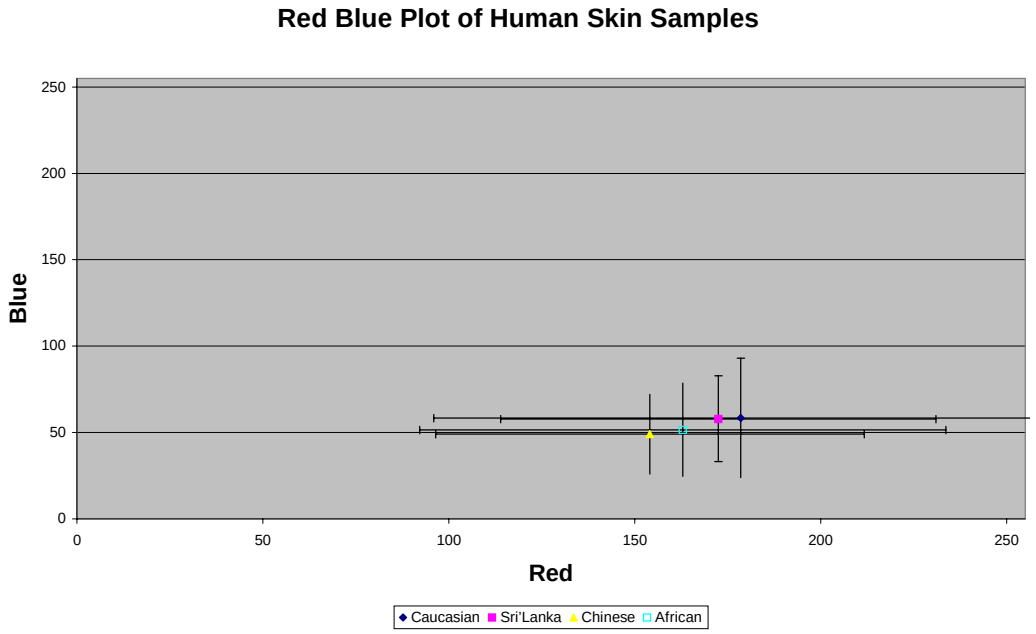


Figure 4.5.4 - Colour distributions of four skin types in r-g and r-b colour spaces

Figure 4.5.4 shows the mean value for each subject plotted with the error bars representing $\pm 2\sigma$. It can be seen in the Red/Green and Red/Blue plots that the various skin tones represent relatively small, overlapping clusters in *RGB* space, with subtle differences between subjects as would be expected. The darkest mean intensities are produced by the Chinese sample which would seem to contradict

any stereotypical observations about skin type. However, this is attributable to the distance of the hand from the camera during sampling. The Chinese sample was taken at a much closer distance than the other skin samples and hence produced darker results. However, this variation in lighting makes little difference to the results of the Hue Saturation plot.

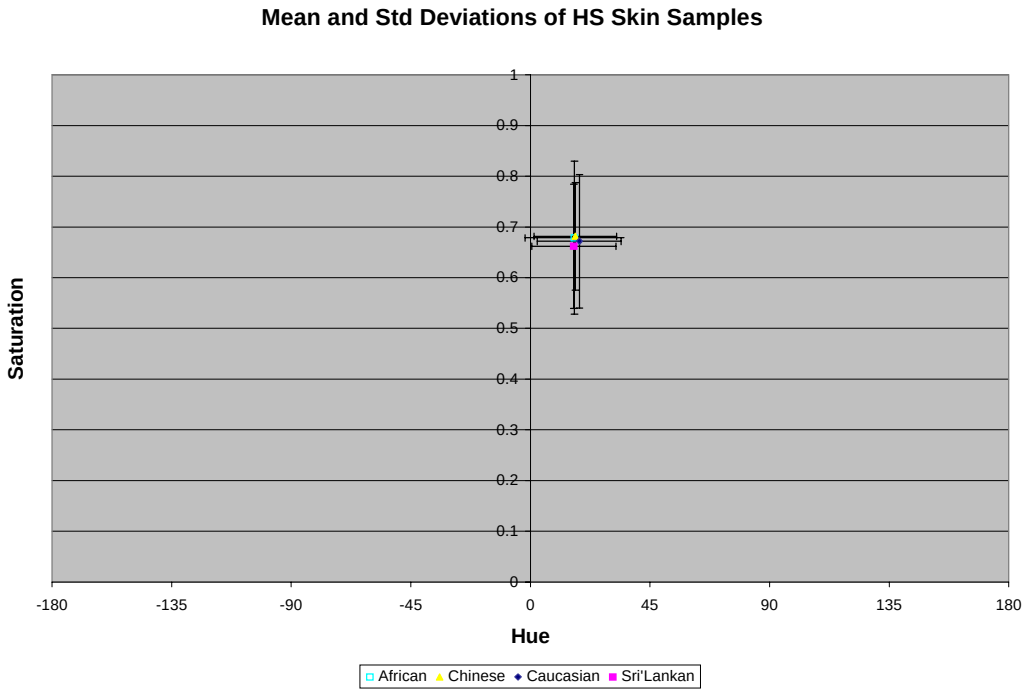


Figure 4.5.5 – Colour distributions of four skin types in HS space

The Hue Saturation plot shows the same statistical representation of the various skin types in *h-s* space. It can clearly be seen that this results in a far tighter colour cluster, which seems to vary little between skin types. Even the Chinese sample that produces dark results due to lighting is indistinguishable in the HS plot.

By using this single extracted cluster in HS space and fitting a multivariate Gaussian to it, a probabilistic measure that any pixel is human skin can be determined. A more accurate Gaussian PDF can be constructed by performing PCA on the colour cluster, and approximating its primary axis in addition to its bounds, or using the sum of Gaussians as used in chapter 5. If a sample pixel

from a new image is within the Hue-Saturation bounds of the Gaussian cluster then that pixel is marked as a probable location. Selecting a threshold for which probabilities of lower values are set to FALSE, and higher TRUE produces a binary image. By performing erosion then dilation, noisy points are removed and clusters of probable skin location consolidated into blobs. A simple blobbing algorithm can then be used to calculate approximate locations of skin artefacts within the image.

Figure 4.5.6 – Extracting Blobs of Skin

Figure 4.5.6 shows a sample image frame after processing. The results from the blobbing algorithm are used to calculate the centre of objects by finding the mean pixel of the blob and the approximate size by assuming circular blobs and calculating the radius of a blob from the area (i.e. the number of points in the blob). This is used to place a cross over the segmented features for demonstration purposes. In this instance the three largest blobs found within the image are deemed to constitute the head and the hands. The largest connected blob extracted from the colour labelled image can be used as a rough initial estimate for the position of the head.

4.6 Tracking Colour Features

Using a single Gaussian cluster to probabilistically segment skin tones from an image leads to noisy segmentations for two reasons:

1. The assumption that a single bivariate Gaussian is a good representation of the colour cluster is not completely valid.
2. Background clutter can be misclassified.

Specular reflections are particularly vulnerable to misclassification. Another draw back with the technique is that all the pixels of the image must be transformed into HSL space and colour classification applied. This process quickly becomes a computational overhead and when real-time applications are considered (25Hz or more) the approach becomes unfeasible.

One alternative is to locally search for skin using a Region of Interest (ROI) or window. Only pixels that fall within the ROI need to be converted and classified which significantly speeds up the procedure. In addition, background clutter, outside the ROI, cannot be misclassified. This produces a much cleaner segmentation without the need for erosion/dilation as previously described.

In order to limit processing to within the window (ROI), a mechanism for moving the window must be devised. This is itself a colour tracker, as the window must track the object in order to successfully segment the skin tones.

If the assumption is made that the binary-segmented object has a central white mass surrounded by black background, then the centre of gravity of the blob should be at the centre of the window.

Using a binary image window of size s_x, s_y where, $I_{x,y}$ is zero for the background and one for segmented skin, the centre of gravity for the segmented feature can be calculated by

$$CG_x = \frac{1}{\sum_x \sum_y I_{x,y}} \sum_{x=-s_x/2}^{s_x/2} \sum_{y=-s_y/2}^{s_y/2} x I_{x,y} \text{ , and } CG_y = \frac{1}{\sum_x \sum_y I_{x,y}} \sum_{x=-s_x/2}^{s_x/2} \sum_{y=-s_y/2}^{s_y/2} y I_{x,y}$$

A simple translation can then be calculated to position the centre of the window at the centre of gravity for the next iteration of the algorithm.

This assumption about the shape of an object within the window can also be used to calculate a new window size for the next iteration. Figure 4.6.1 shows a window of size 45x77 pixels with a binary segmentation of a hand achieved using the Gaussian probabilistic threshold described earlier. The figure also shows the horizontal and vertical histograms of the image. If the earlier assumption about the location of an object within the window holds true, then it can be assumed that these histograms will be approximately Gaussian, with their peaks at the centre of gravity previously calculated. By making this Gaussian assumption, the standard deviation in both x and y can be calculated and the bounds of the window for the next iteration estimated. Figure 4.6.1 also shows this fitted Gaussian curve superimposed upon both the x and y histograms. The Gaussian curve is estimated by calculating the standard deviation of the histogram in both x and y. Once done it is known that one standard deviation from the mean (σ) represents 34.1% of the information, 2σ represent 47.7% of the information and 3σ represents 49.9% (See Chebyshev's theorem, Section 3.2). It is therefore known that $\pm 2\sigma$ from the mean encompasses 95.4%. This simple calculation can be used to resize the window ensuring that over 95% of the information is encompassed by the ROI. In the Figure 4.6.1 the window is resized to $\pm 2.2\sigma$ where,

$$new\ size_x' = 4.4\sigma = 4.4 \sqrt{\frac{1}{\sum_x \sum_y I_{x,y}} \sum_{x=-s_x/2}^{s_x/2} \sum_{y=-s_y/2}^{s_y/2} I_{x,y} (x - CG_x)^2}$$

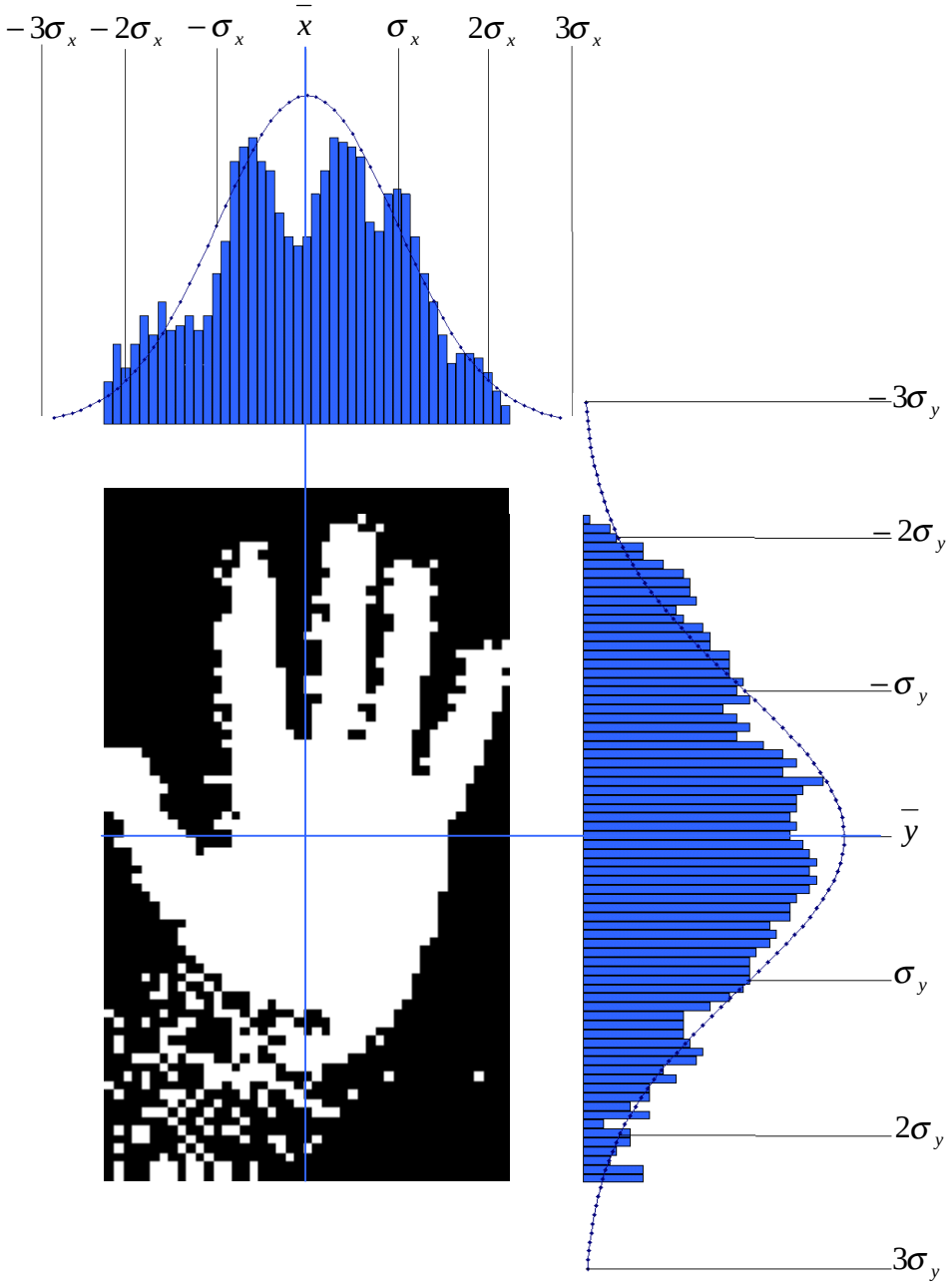


Figure 4.6.1 – Approximating the bounds on an object using a Gaussian Assumption

This simple procedure is iterated for each new image frame of a real-time image sequence. It relies upon a good initial location of the window. However, this can be achieved by performing the full image segmentation as described in section 4.5.

An Algorithmic overview is:

1. Construct PDF for colour thresholding model
2. Assign probability to each colour pixel from PDF
3. If probability is greater than some threshold mark pixel as TRUE else FALSE
4. Search image for largest blob
5. Calculate centre of blob and initialise window to this position
6. Calculate the approximate size of the blob and use to initialise window size

   $s_x = s_y = 2\sqrt{\frac{blob_{area}}{\pi}}$
7. While window size is greater than some threshold,
   8. Capture new image
   9. Segment window using PDF and threshold
   10. Calculate mean white pixel in x and y
   11. Move window to x,y
   12. Calculate the standard deviation in x and y, σ_x, σ_y
   13. Resize window to $2.2\sigma_x, 2.2\sigma_y$
14. Return to 1

If the object is much larger than the window, then the Gaussian that is fitted will be far larger and hence the window will grow in size until equilibrium is achieved. Conversely, if the window is too large, the resulting Gaussian will be far smaller than the window and hence the window will reduce in size until equilibrium has been achieved. This approach allows colour objects to be segmented and tracked quickly as the minimum amount of processing is necessary on each frame.

Figure 4.6.2 (a) and (b) shows the progress of applying this active sampling window to a live image sequence. As the hand is moved and rotated in the image frame, the window dynamically recalculates its parameters to retain the hand within its ROI. Figure 4.6.2 (c) shows the same procedure applied to the head with no change in parameters. Although the model is trained upon a single human, it has proved a generic skin tracker for all subjects regardless of skin type and without the need for relearning the colour space of skin. If however, the

lighting is changed, this requires that a new skin model be learnt due to the large variations in frequency for different sources of light (i.e. fluorescent tube or daylight). This provides a generic tracking approach for applications with consistent illumination.

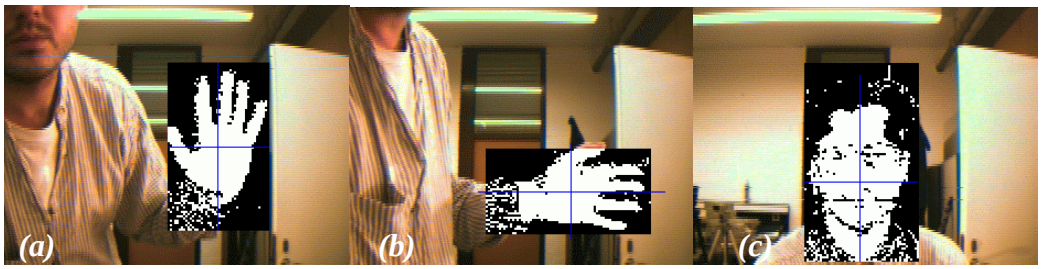


Figure 4.6.2 – Tracking head and hand in the image frame using colour

4.7 Conclusion

This chapter has demonstrated how colour can be used without high computational cost to enhance vision algorithms. Several colour spaces have been discussed and the benefits of 'perceptual' colour spaces demonstrated. It has been shown that object colour is a powerful feature capable of facilitating the robust tracking of objects in its own right. It has also been shown that with simple techniques, colour features can provide a fast, robust approach to tracking any generic colour object.

Throughout the remainder of this work, many of the simple techniques presented here will be used to enhance techniques in general. Chapter 10 will actively use the colour tracker approach presented in Section 4.6 but throughout the remainder of this work the use of colour in PDM tracking and boundary segmentation is implicit.

Chapter 5

5 Cluster Based Non Linear Point Distribution Models

5.1 Introduction

As was already mentioned in chapter 2, the major drawback with models which rely upon principal component analysis to model deformation is the non-linearity which is introduced either as natural curvature, inherent to the model, or introduced during the alignment and construction process of the PDM. This non-linearity within shape space (or PCA space) results in poor performance due to the linear nature of the underlying mathematics.

Bregler and Omohundro proposed estimating non-linearity by breaking PCA space down into piecewise linear clusters which could then be modelled with multiple hyperplanes [Bregler 94]. More details on this technique are discussed section 5.4. However, these *Constraint Surfaces* do not place any limits upon the local linear patches within the model and hence the surface extends to infinity producing un-specific models. The work of Bregler also concentrates on extremely low dimensional shape spaces with minimum non-linearity, where little concern is given to the application of computationally expensive techniques. In practice, the technique does not perform well in high dimensional spaces (as will be shown) due to both the computational complexity of cluster

analysis and PCA, in addition to the problems associated with discontinuous shape spaces[1].

The remainder of this chapter will propose an alternative approach, which, although similar in nature, produces a more specific model. The construction of such models along with the parameter selection will also be discussed. Section 5.3 will present the use of dimensional reduction techniques to disregard redundancy in high dimensional data, allowing analysis to be performed in lower dimensional spaces. Section 5.4 will discuss the method behind piecewise linear approximations. Section 5.5 will then demonstrate the use of the technique with example data sets. Section 5.6 will discuss the application of the model. Finally the technique will be evaluated and compared to other approaches in section 5.7 and conclusions drawn.

5.2 An Example of non-linearity

One of the classic examples within the field of neural networks is that of a helical data set. Helical datasets are often used to assess a neural network's ability at creating a non-linear mapping. Figure 5.2.1 shows a helix in three dimensions from a front and plan view. Although the helix exists in 3D, it is actually a one-dimensional data set, and can be smoothly paramertised by a single value if the primary non-linear axis, which follows the path of the helix, can be extracted.

[1] see Figure 5.4.5 and associated text for details

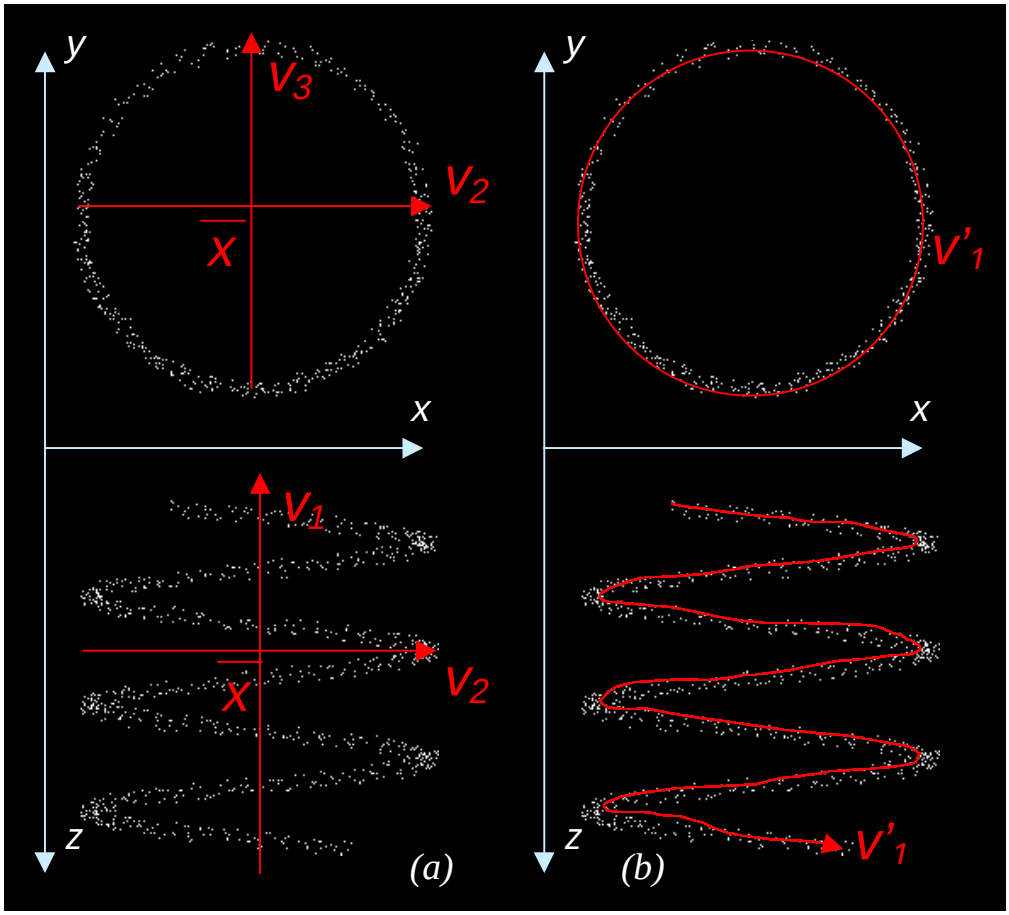


Figure 5.2.1 - Linear PCA, three-dimensional helical data set

(a) a helix using three orthogonal axis, (b) a single non-linear axis

Figure 5.2.1(a) shows the helix represented using three primary axes as determined by linear principal component analysis: $\bar{x}$ is the mean value and exists outside the bounds of the helix, the vectors $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ are the three orthogonal axes as extracted through PCA. The helix does not lie on any single axis and all three must be used in order to reproduce the path of the helix.

In terms of shape space, where the primary concern is to encompass the bounds of a training set in the most compact and constrained way possible, this is an extremely inaccurate representation as both the mean shape and primary modes are not indicative of the training set shape (ie the helix). Using this linear approach would not only allow paths to be produced which are indicative of the helix but many other non-representative paths within the volume bounded by the vectors $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$.

Figure 5.2.1(b) shows the helix parameterised by a single non-linear axis which closely follows the path of the helix. Any point on the helix can be represented by a single parameter which indicates the distance along this primary axis from some origin. In order to accurately represent the non-linear data set, a means of extracting the non-linenear axis is required. Unfortunately the data set is seldom parameterised by a single axis and the problem of extraction is compounded by the high dimensional nature of computer vision applications.

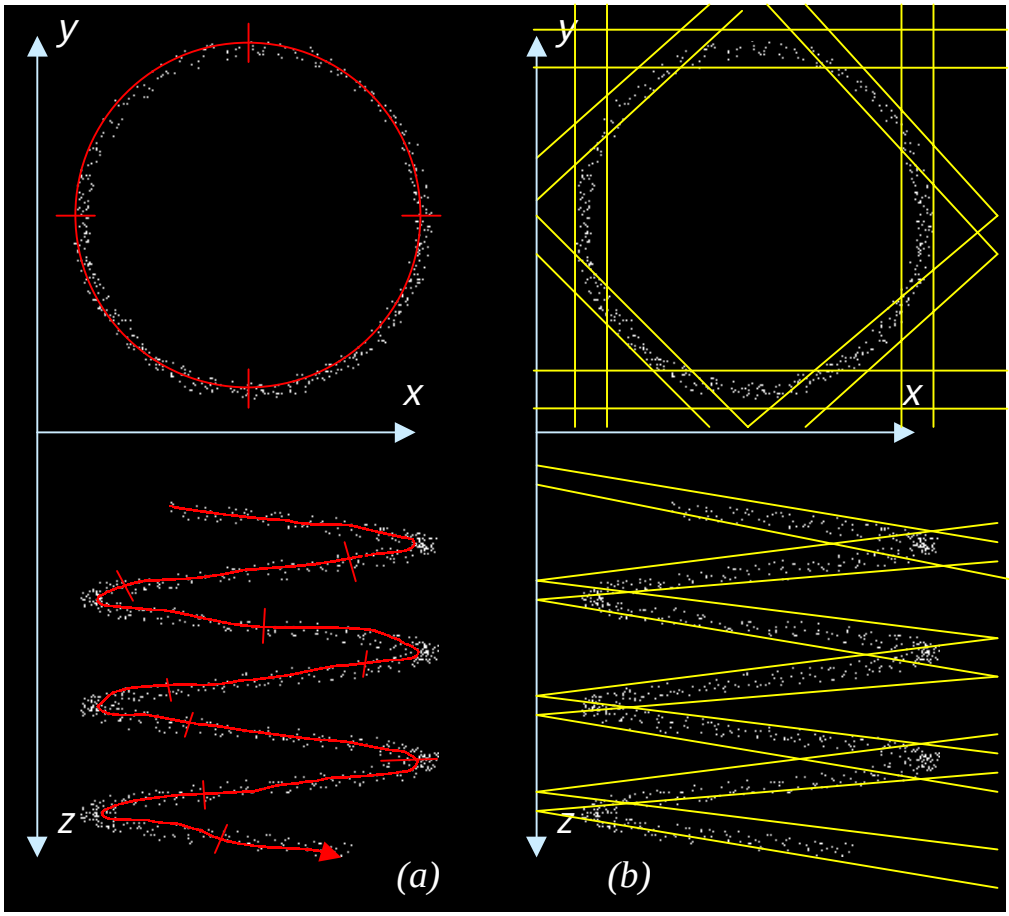


Figure 5.2.2– Non-linear PCA, three dimensional helical dataset

(a) non-linear modes of variation (b) segmenting shape space with multiple planes

Figure 5.2.2(a) shows a secondary axis fitted to the data set. Here, the secondary mode changes dependent upon the position along the primary axis. The fitting, therefore, becomes a computationally expensive process in even the lowest of

dimensional spaces. Figure 5.2.2(b) shows how the space can be segregated through the use of multiple hyper-planes. This is akin to the procedure used by a neural network when fitting to a data set. Although faster than attempting to fit true curved axis to the data, it is essentially **estimating** the curvature to a specified degree and hence has a loss in accuracy. This procedure also becomes an infeasible approach as the dimensionality of the space increases. In order to find a suitable technique for performing non-linear PCA, two considerations must be addressed: the dimensionality of the data set must be reduced to a managable level; a means of estimating the non-linearity (while retaining a low computational complexity in both analysis and run time implementation) of the final model is required.

5.3 Reducing Dimensionality

It is often important to decide what is the actual dimensionality of a data set, as the true dimensionality is often lower than the dimensionality of the space in which the data lies. This statement is more accurate when large dimensional spaces are considered. For example a data set may exist in two dimensions, but if it lies along a straight line then the true dimensionality is 1D. If, in general, the position $\underline{x} \in \Re^N$ of a point in N-dimensional space were representable by a relationship of the form $\underline{x} = \underline{x}(\underline{u})$, where $\underline{u}$ is a point in $\Re^M$, then the data is said to be M-dimensional. The transformation $\underline{x} : \Re^M \rightarrow \Re^N$ provides the mapping between the two spaces and allows any point $\underline{x} \in \Re^N$ to be dimensionally reduced to $\Re^M$ [Waite, 1992]

Using PCA, the value of *M* can be determined and the information loss estimated. This procedure also provides the transformation matrix that facilitates the projection $\Re^M \rightarrow \Re^N$.

The process of principal component analysis realigns the axis to fit the major deviation of the data set. These *extracted* axes can be used to describe the data in a new co-ordinate frame, which is the principle behind the PDM. As is typically the case, training data can be represented using fewer eigenvectors than the

original dimensionality (see Chapter 2). This is itself a *lossy* dimensional reduction technique and relies on transforming the shape space into a lower dimensional space. In this reduced dimensional space the original data and its deformation from the mean can be expressed using the fewest number of parameters possible as determined from the eigenvectors of the covariance matrix.

By transforming the eigenvectors into percentiles it can be quickly seen how the dimensionality of the reduced space relates to the information loss of the reduction technique. By using the same analysis of this information as is used in the construction of the PDM (see section 3.2) a suitable mapping can be determined which provides minimal loss of information, typically less than 1%.

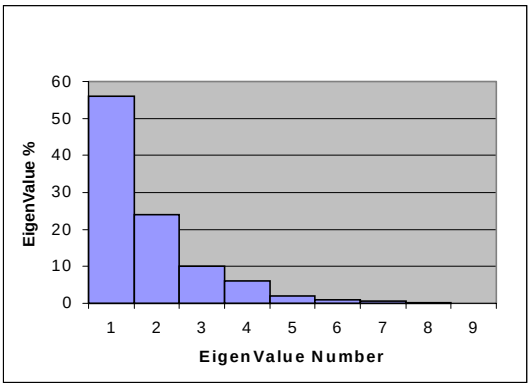


Figure 5.3.1- Table showing eigenvalues of co-variance matrix extracted via PCA

Figure 5.3.1 shows an example bar chart of eigenvalues extracted from a co-variance matrix, converted into percentiles and sorted into order. It can be seen that the 1st mode contains the majority of the deformation within the data set with the subsequent eigenvectors contributing in diminishing amounts. By summing the percentage contribution of each of the eigenvectors, a suitable dimensionality for the reduction can be determined (see section 3.2). For this example 99% of the deformation is encompassed within the first 6 eigenvalues with the last three contributing little to the information. These smaller 3 modes can therefore safely be discarded without adversely affecting the information content of the data set. It is also useful to note that these smaller modes are often

largely attributable to noise within the data set and hence discarding this information can have benefits in smoothing the data.

Once the dimensionality, *M*, of the reduced space $\Re^M$ has been determined, the *M* primary eigenvectors can be used to project the original data set into this lower dimensionality. This is achieved by projecting the training examples onto each of the eigenvectors in turn, and recording the distance from the mean. The resulting transformed training set will therefore be represented in the lower dimensional space (using the co-ordinate frame of the eigenvectors), while the important information about the shape and size of the data remains preserved.

The dimensionally reduced vector is calculated as $\mathbf{x_r} \in \Re^M = (d_1, d_2, ..., d_M)$, where the j^{th} component,

$$d_j = v_j \bullet \left(x - \bar{x}\right) \qquad \textbf{Equation 5-1}$$

or alternatively in matrix form where $\mathbf{P} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)^T$ is a matrix of the first *t* eigenvectors

$$\mathbf{x}_r = \mathbf{P}^T \left(\mathbf{x} - \bar{\mathbf{x}}\right) \qquad \textbf{Equation 5-2}$$

To reconstruct the original vector x, from the d_j component of the reduced vector x_r,

$$x = \bar{x} + \sum_{j=1}^{nr} d_j v_j \qquad \textbf{Equation 5-3}$$

Note that equation 5-2 is the formulation for the linear PDM, where each component of the reduced vector is effectively the weighting parameter of the final shape.

This does not provide a true dimensional reduction, as *M* eigenvectors $v_{1 \to M}$ must be stored for use in the transformation between the reduced and original dimensional spaces. However, the primary concern, which is perfectly satisfied by this technique, is to reduce the dimensionality of the training set for non-linear analysis.

5.4 Estimating Non-linearity

It has already been shown how non-linearity can be estimated by breaking the shape space down through the use of multiple planes (Figure 5.2.2). A similar procedure can be performed by breaking the curvature of the space up into piecewise linear patches which estimate any curvature present. This is similar to the polygonal representation of a parameterised surface. As the number of polygons increase, so the visual accuracy of the resultant surface increases. However, as in most graphical (polygonal) representations there is a trade-off between the number of polygons (and hence render speed) and the accuracy of the representation. This optimum number of polygons is easily selected for graphical representation dependent upon simple visual criteria. For high dimensional data sets this number is more difficult to determine.

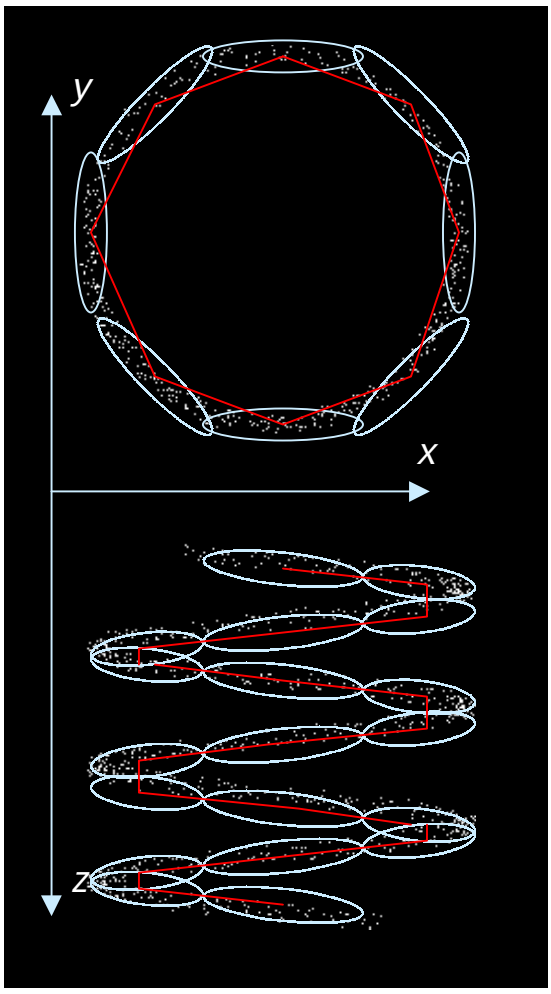


Figure 5.4.1 - Cluster Based Approximation

Figure 5.4.1 shows the helical data set broken down into smaller clusters which themselves can be treated as linear patches. The centres of each of these clusters when connected allow the estimation of the primary mode of the helix. Each cluster contains local information on how the data set varies, and must be analysed further in order to provide an accurate representation of the space. However, providing the space is segregated into a sufficient number of clusters, each can be treated as piecewise linear patches which encompass the major curvature of the space. The assumption that each cluster is approximately linear allows a local linear mathematical model to be used, such as principal component analysis. To provide a smooth transition between these linear patches it is important that there is a good overlap between them. This is important where a gradient descent approach is to be used in tracking, as a single iteration of the model may not be sufficient to allow the model to make the transition between two adjacent, non-connecting clusters.

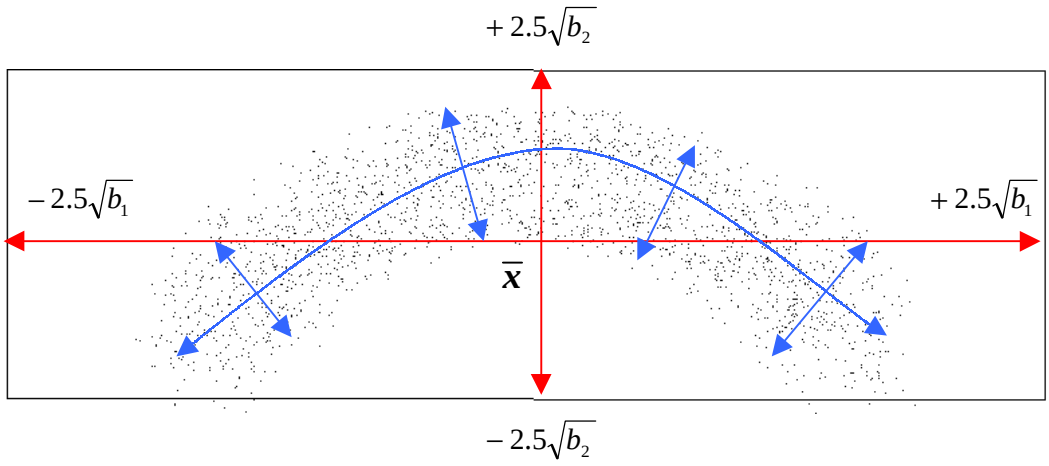
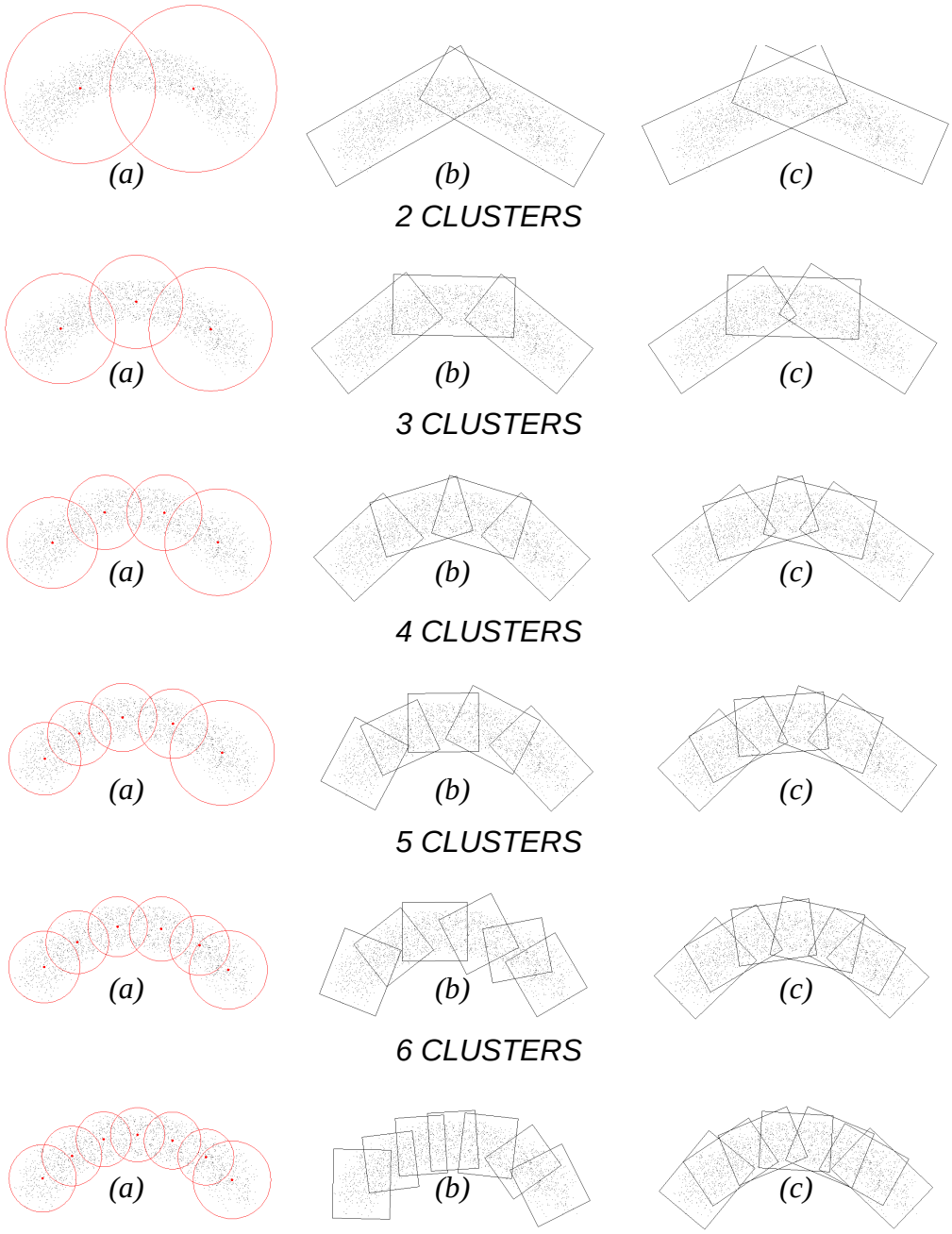


Figure 5.4.2 - Linear principal components of a curved data set

Figure 5.4.2 shows a synthetic data set with 2000 members in a two-dimensional curve. Performing standard linear PCA on this data set gives two primary modes, which are represented by the red arrows. Using suitable limits to bound these modes (2.5 times the square root of the corresponding eigenvalue from the mean shape) gives the bounding box shown in the diagram. It can be clearly seen that the mean shape is only just within the training set and the boundaries encompass

far more of the space than is inhabited by the data points. The blue lines show the ***ideal*** primary and secondary non-linear axis of the data set.

Using this piecewise linear approximation to model the non-linear data set results in a more constrained model which better represents the original shape space. Figure 5.4.3 demonstrates the use of (a) cluster analysis to break down the original space into linear patches, and (b) the resulting bounds of these patches after linear PCA have been performed upon them for increasing number of clusters. (c) shows the results of the fuzzy k-means algorithm.



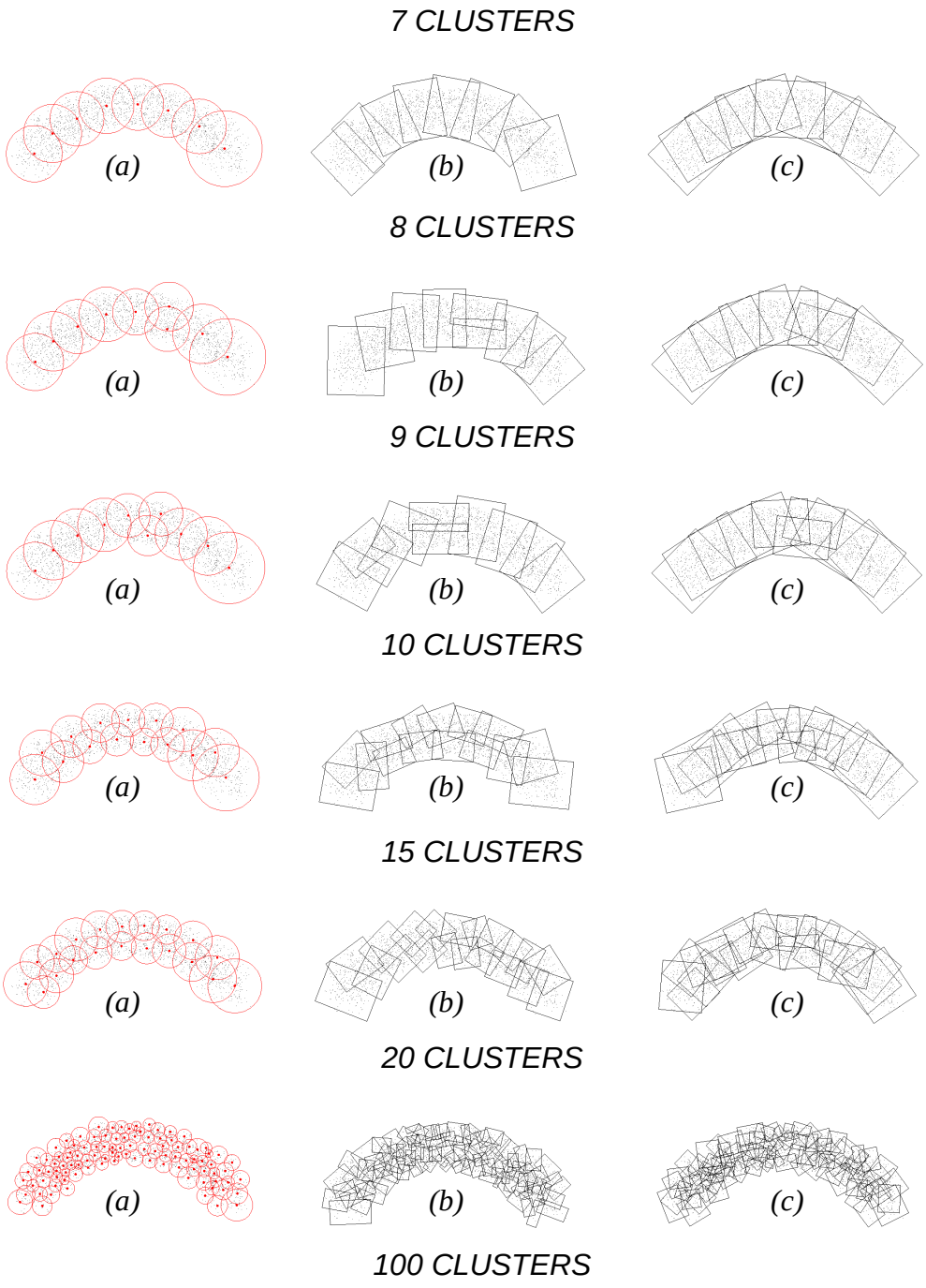


Figure 5.4.3 - Cluster analysis on shape space

(a) Cluster centres and bounds, (b) k-means (c) Fuzzy k-means

Figure 5.4.3(a) shows the results of running a k-means clustering (see Appendix 1) algorithm on the synthetic data set with curvature. The red points depict the centres of the final extracted clusters and the circles show the approximate bounds of these clusters. Using cluster analysis to segregate the space, PCA is then performed upon each cluster and the results are shown in Figure 5.4.3(b).

Each bounding box shows the extent of each linear patch, modelled as $\pm 2.5\sqrt{\lambda_i}$ (as described earlier). It should be noted that as the number of clusters is increased the resulting model better encompasses the curvature, although the rate of increase in accuracy diminishes as more patches are used.

It is clear from the 2-cluster example that it performs significantly better than the single linear PCA model and greatly reduces the redundant space, which is incorporated into the final model. When the number is increased to 3 or 4 clusters there remains a visible benefit in the accuracy of the model. However, as the number of clusters is increased further it becomes increasingly hard to determine if the benefits in model specificity can be justified against the increase in computational complexity. In the analysis of *true* data, where it becomes impossible to visualise the high dimensionality of the space, such visual assessment is not possible. An alternative method of assessment for choosing the number of clusters can be provided through normal cluster analysis as described in Appendix 1. From Figure 5.4.4 the natural number of clusters can be estimated to be 5 which ties in with the visual observations discussed earlier.

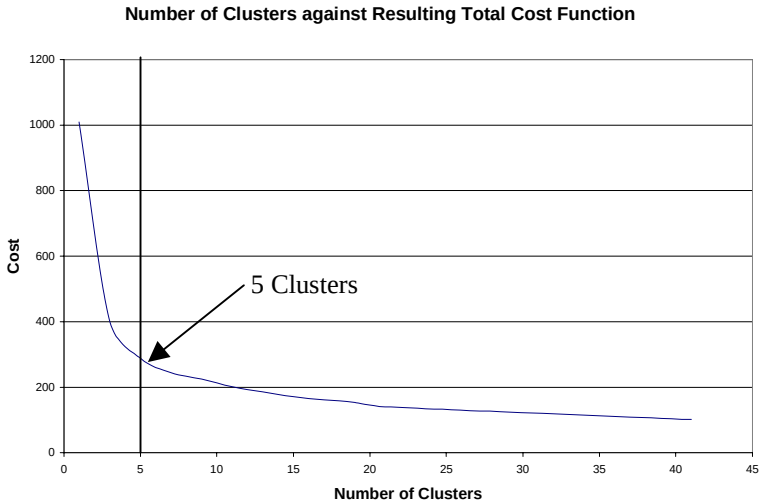


Figure 5.4.4 - Cost graph for synthetic curved data set

Figure 5.4.3(c) shows the results of using a fuzzy k-means clustering algorithm (see Appendix 1) on the same data set. It can be clearly seen that using the fuzzy algorithm significantly increases the overlap between adjacent clusters and provides a smoother composite model for estimating non-linearity. This is

important during tracking, especially when using a gradient descent approach (iterative refinement approach). This ensures there exists a smooth path between the composite elements of the model.

Bregler and Omohundro [Bregler 94] made no provision for this problem when separating the shape space into sub-clusters and hence this adds to the observed model error which will be shown during comparison in section 5.7.

This technique also allows discontinuous surfaces to be modelled accurately, which is an important consideration when attempting to model non-linearities for computer vision applications. If a test example were to be considered in which a break exists in the training set (see Figure 5.4.5), then existing techniques would attempt to model this discontinuity by a single model. The resulting linear PDM would be similar in nature to that shown in Figure 5.4.2(a).

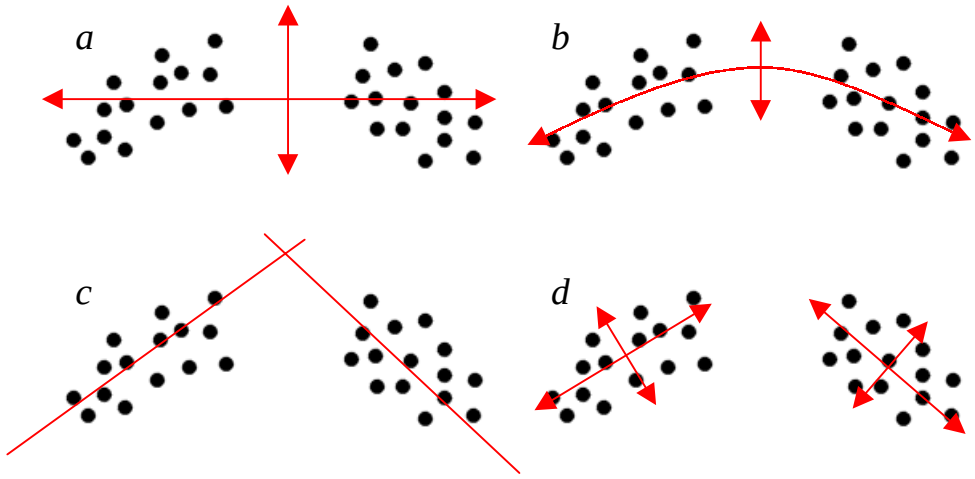


Figure 5.4.5 - Modelling Discontinues Data Sets - Types of Model

(a) Linear PDM, (b) Polynomial Regression PDM,
(c) Constraint Surface (d) Composite NLPDM

Figure 5.4.5 shows an example discontinuous data set with various forms of PDM model fitted: *(a)* shows the linear PDM which models the entire space as a single rectangle, the mean within the central ***null*** space; *(b)* shows the non-linear axis of a polynomial model smoothly parameterising the curvature, still with a mean shape within the ***null*** space; *(c)* shows the constraint surface approach of

Bregler which models the space as two finite thickness infinite hyperplanes; and (d) shows the composite NLPDM technique proposed here.

If new points are considered and the closest valid shape found within the model, the performance of each approach can be assessed.

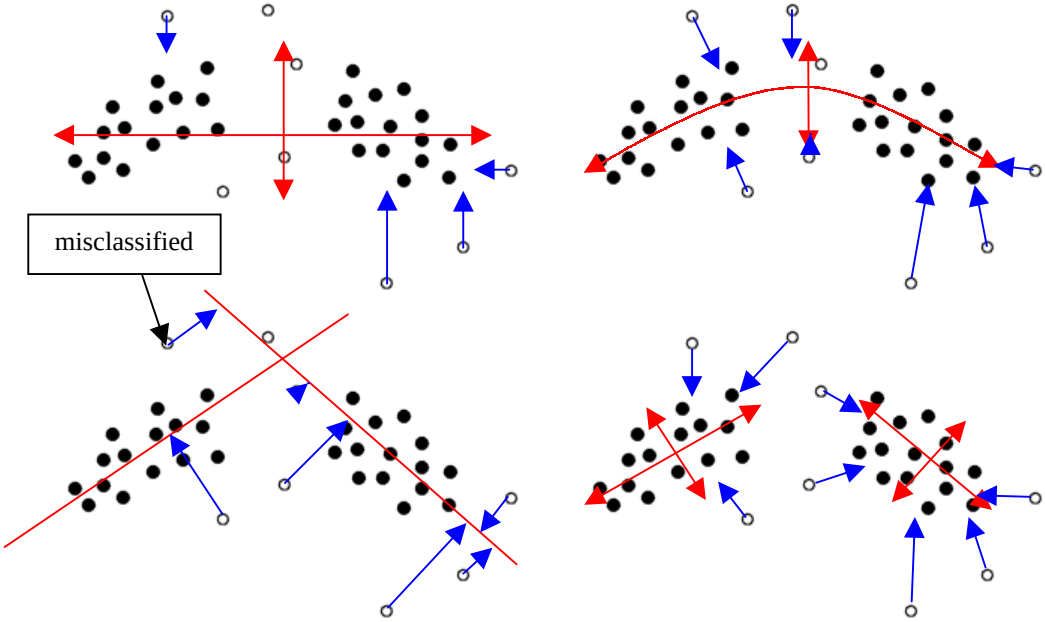


Figure 5.4.6 - Modelling Discontinues Data Sets - Nearest Valid Shape

(a) Linear PDM, (b) Polynomial Regression PDM,
(c) Constraint Surface (d) Composite NLPDM

It can be seen from Figure 5.4.6 that the linear PDM performs poorly for both the modelling of curvature and the discontinuity of the data set: many points remain unconstrained within the central *null* area. The polynomial model works well at modelling curvature; however, it performs poorly at modelling discontinuity. Although points on the extremities are drawn closer to the original training set shape, points within the *null* area remain unchanged. The constraint surface models curvature to an extent, but draws all model points to lie along the hyperplanes and does not work well for the discontinuity. In addition, the unlimited extent of the hyperplanes introduces further errors at boundaries, allowing points to be misclassified to the wrong hyperplane. The composite NLPDM seems to be able to model both types of non-linearity correctly, and

only introduces boundary errors due to the rectangular assumption of linear patches.

An example of complex discontinuous surfaces can be found in Section 7.3.

5.5 Composite NLPDM

This section presents two test cases to demonstrate the validity of the approach at modelling non-linear data sets. The examples were chosen to represent both high non-linearity and high dimensionality. The construction of the composite non-linear PDM is outlined below.

An algorithmic overview is given below.

1. Perform PCA on training set

2. For each training example do

Project training example onto eigenvectors, recording distance from mean.

Concatenate these distances into a reduced dimensional vector.

3. Perform cluster analysis on dimensionally reduced data set to determine natural number of clusters

4. Use this natural number to segregate the data set into multiple clusters using fuzzy k-means

5. Perform PCA on each cluster of training set

5.5.1 Robot Arm

The first example that will be considered is of a relatively low dimensionality, but with high non-linearity present. The robot arm example meets these criteria as the nature of its hierarchical, pivotal construction guarantees a non-linear data set. The training data for the robot arm example was constructed automatically from a synthetic model used to generate examples that encompassed the total possible movement of the arm. Figure 5.5.1 shows the construction of the arm model. The 2D representation of a robot arm consists of four rectangles, each rectangle described by four key points at its corners. This gives a total of 16 2D key points which, when concatenated together, provide a 32 dimensional vector

that describes the shape of the arm at any time. The model also incorporates 3 pivotal joints, which allow the constituent sections of the arm to rotate about each other. Examples were generated for the arm in all its various positions by taking examples of the model as the joints were rotated from $\pm 45°$ in $10°$ intervals. This resulted in a 32 dimensional training set containing 918 examples.

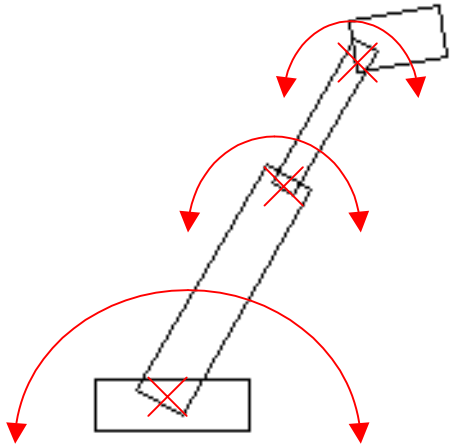


Figure 5.5.1 - The construction of a non linear robot arm data set

Figure 5.5.2 shows examples taken from the synthetic training set.

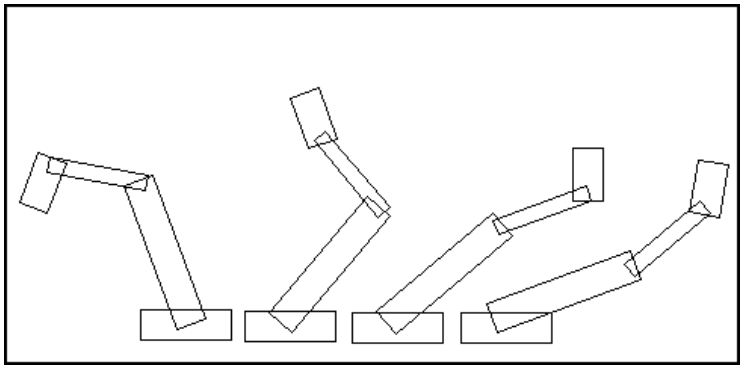


Figure 5.5.2 - A selection of training examples from the robot arm data set

As the dimensionality of the model is already low (i.e. 32D) it is not necessary to perform dimensional reduction on the model and therefore k-means analysis can be carried out on the raw data set. Performing standard cluster analysis (see Appendix 1) the graph in Figure 5.5.3 is produced and indicates the natural number of clusters to be approximately 20. Using this number of fixed clusters

the fuzzy k-means algorithm is applied in order to segregate the data set into its constituent linear patches.

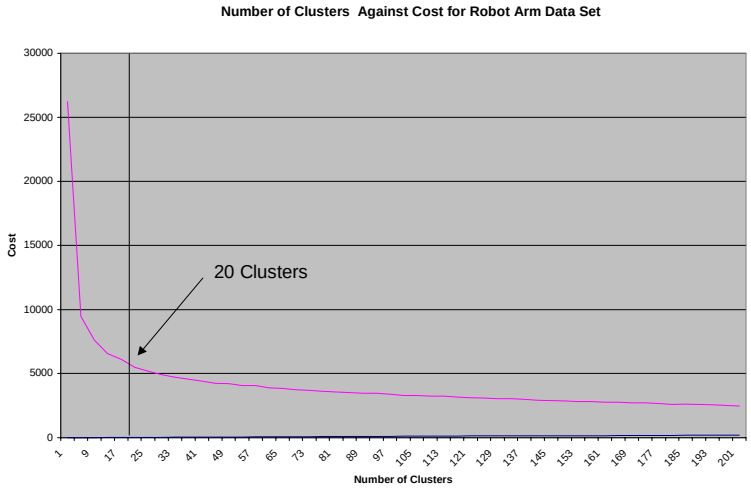


Figure 5.5.3 - Cluster analysis on raw robot arm data set

Figure 5.5.4 shows the resulting boundaries on the data set after PCA has been performed on the extracted clusters projected into 2-dimensions. Note that rectangles are skewed due to the projection of each model ($m_{0\text{-}31}$) down from 32 to 2 dimensions. This figure clearly shows the non-linearity of the model and how the linear patches estimate this curvature.

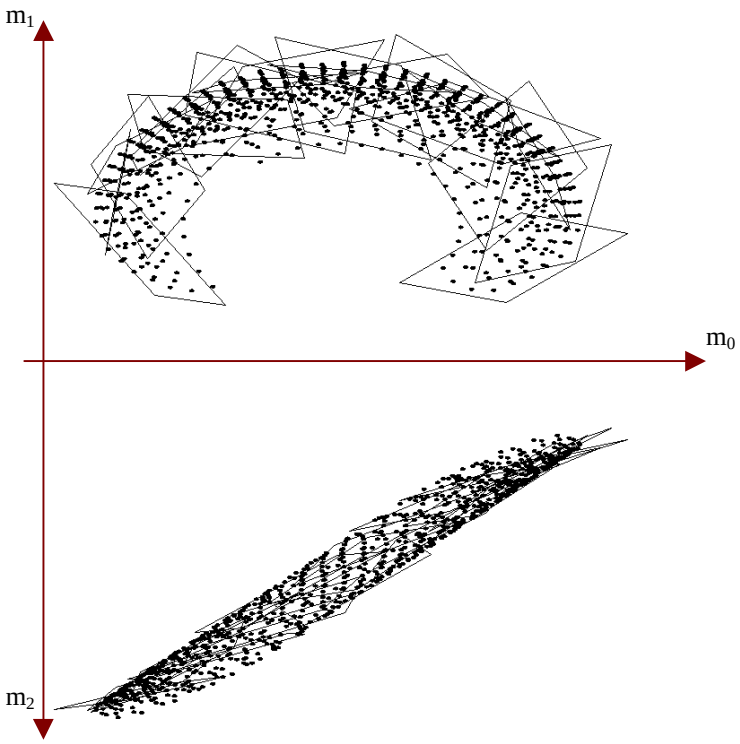


Figure 5.5.4 - Linear patches of the robot arm data set

In order to validate the hypothesis that reducing the dimensionality of the data set before analysis does not affect the information content of the resulting model, the procedure was repeated upon the data set after dimensional reduction.

PCA was first performed upon the raw data set and from the eigenvalues a suitable reduction was determined. 99% of the deformation is contained within the first 4 eigenvectors, corresponding to the four largest eigenvalues. The data set was then projected down into this 4 dimensional space using *equation 5-1* (page 65). Cluster analysis was then performed to extract the natural number of clusters and the fuzzy k-means algorithm performed to extract the membership of each cluster. The results of cluster analysis can be seen in Figure 5.5.5. Apart from the difference in the scale of cost, the graph is almost identical to that previously produced and, as in Figure 5.5.4, provides a natural number of clusters equal to approximately 20.

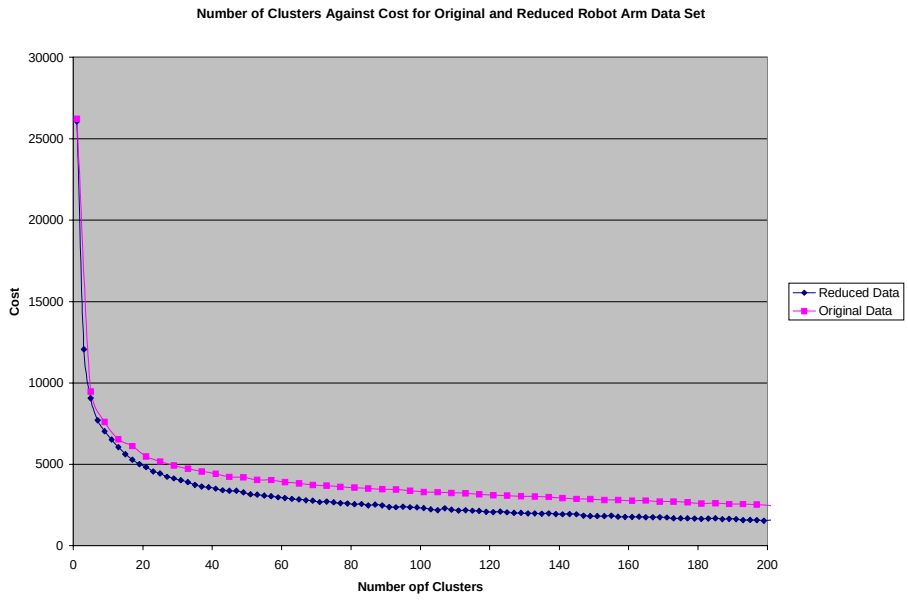


Figure 5.5.5 - Cluster analysis on the reduced robot arm data set

Once the cluster membership has been extracted, each element of the clusters is transformed back into the original space using the *equation 5-2* (page 65) before PCA is performed. This procedure leads to the loss of up to 1% information due to the lossy compression technique used. As an alternative, the reduced vectors

can be used merely as pointers to the original data set, since the 1st element of the reduced data corresponds to the 1st element of the original data. Once this reverse mapping has been completed, PCA is performed on each of the fuzzy clusters to produce the composite model as done previously.

The lower cost solutions for the reduced dimensional data results from the disregarded data no longer contributing to the overall cost of the k-means function. However, although this makes little difference to the selection of the *natural number*, it provides a huge computational saving as the analysis is performed in a 4 dimensional space rather than one of 32. In fact, if the assumption is made that the primary modes contain the largest contribution to the separation of shape space (which is known), then this cluster analysis could feasibly be performed with even higher dimensional reductions. However, it is not obvious how this number would be selected.

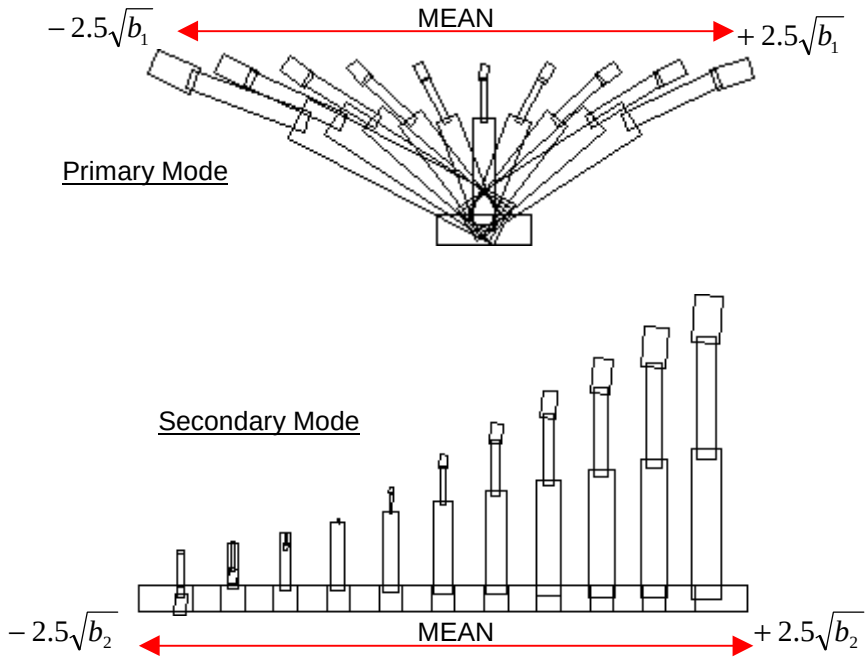


Figure 5.5.6 - Primary modes of the linear robot arm PDM

Figure 5.5.6 shows the primary and secondary modes of variation of the linear PDM. The non-linearity of the model is clear in the distortion of the dimensions of the robot arm. The primary mode encompasses movement along the horizontal, but also has distortion in the size of the arm, which must be rectified

by other higher modes of variation. The second mode encompasses movement in the vertical, with more extreme size distortions, especially at the head of the model. Below the mean on the second axis, the model takes on shapes which were not present within the training set by inverting the arm back upon its self.

Figure 5.5.7 shows examples from the final composite non-linear model. It demonstrates that much of the non-linearity has been removed except in the end of the model where small abnormal deformations can still be seen. By increasing the number of clusters this can be reduced further, but at a computational cost at run-time.

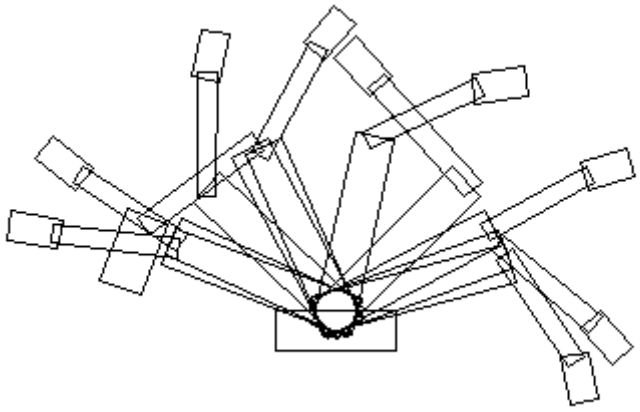


Figure 5.5.7 - Examples from the non-linear robot arm PDM

5.5.2 Image Space

An image training set was constructed from a sequence of 200 images of a head turning in the image frame. No alignment was performed so as to produce as non-linear a problem as possible. Each frame is 80 by 60 pixels in size, producing a 4800 dimensional training vector. PCA is first performed and the 33 eigenvectors corresponding to the 33 largest eigenvalues extracted. These vectors account for 99.9% of the deformation in the training set. Figure 5.5.8 shows the first and second modes of variation after linear PCA

Figure 5.5.8 - Primary modes of the image PDM

Each vector is then projected into this PCA space (using *equation 5-1 page 65)* giving a new dimensionally reduced training set on which cluster analysis can be performed. This generates a dimensional reduction of 4800 to 33.

Cluster analysis results in an estimate for the natural number of clusters, k=15. PCA is performed on each of the 15 clusters in turn to generate the composite non-linear model. Selected shapes reconstructed from the composite model are shown in Figure 5.5.9. Notice that each model has reduced blurring, due to the original data set being subdivided into smaller clusters. Each cluster now has less information to encode and hence linear PCA can better estimate the deformation.

Figure 5.5.9 - Examples from the composite non-linear image PDM

As mentioned earlier the technique also has the advantage that the hyper surface, or volume, on which the data lies need not be contiguous. For example, given an image sequence of two people, one with glasses and one with a beard, both linear PCA and the high order non-linear approaches will model the data set with a principal mode which interpolates between the two. However, there is no example in the training set where both glasses and a beard are present. The cluster-based technique will separate these two distinct clusters, allowing the

model to ‘jump’ between the two, better representing the training set. This issue and its implications will be discussed in the following chapters.

5.6 Application of the Model

To apply the model to an image, a similar procedure to the linear PDM (see section 2.3) can be used. After making an iterative refinement to the model within the image frame, the closest possible shape within the *learnt* bounds of the model is calculated. This constrained shape is then used as the model pose for the next iteration.

In the case of the linear PDM, this *constrained shape* is found by projecting the model into the PCA space and reconstructing the closest allowable model (point in shape space) that is within the bounds of the linear model. The same procedure can be used in the composite model. However, the closest allowable point may exist in any of the clusters which constitute the non-linear model. The centre of each cluster can be used to check for closest cluster in Euclidean distance from the model point. However, using a Euclidean distance metric makes the assumption that all clusters are of the same size. Figure 5.6.1 illustrates this problem. Assuming a point *p* in shape space, it should be apparent that the point belongs to the cluster *C1*. Using a Euclidean distance metric will result in the point being assigned to the cluster *C2* due to the size difference in the clusters. However, the point *p* is actually closer to the cluster *C1* even though in Euclidean space the point is further from the centre C1 due to the standard deviation of the clusters.

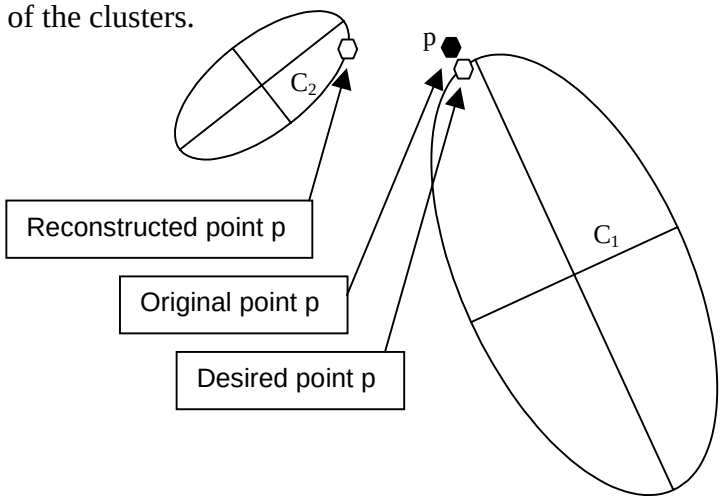


Figure 5.6.1 - Distance Metrics in Shape Space

To overcome this problem a Mahalnobis distance metric can be used. However, due to the simplicity of the k-means-clustering algorithm, it is a fair assumption that if the selected natural number is correct then clusters will be similar in size. It is important to bear this consideration in mind, especially when discontinuous surfaces are considered. In these situations, many clusters may be of different sizes and therefore the Mahalnobis approach should be used.

An algorithmic overview for model application is:

For a new shape **S**,

1. Transform **S** from image frame to PDM model basis eg. Normalise and align (as in alignment of training set)
2. Locate closest cluster centre and hence linear patch $\mathbf{P_i}$ using either Euclidean or Mahalnobis distance metric
3. Project **S** down onto linear patch $\mathbf{P_i}$
4. Project back up to reconstruct closest allowable shape **S'**
5. Transform **S'** back into image frame co-ordinates

5.7 Evaluation and Performance

To asses the performance of the approach to the modelling of non-linear data sets an error metric must be defined which provides a measure of the accuracy of an approach. As has already been demonstrated, a common problem with the linear representation of non-linear data is the tendency to over-generalise shape and to incorporate non-valid deformations into the model. These non-valid deformations often manifest themselves as the distortion in scaling of the model as observed in the robot arm example (section 5.5.1). In this example, the robot arm should remain constant in size and area as it rotates around its pivotal joints. Since this size is the major artefact of the linear representation, it provides a suitable error metric with which to assess non-linear performance.

Random points chosen from within the linear PCA space are selected and then projected into the composite model. The constraints of the model are applied and the resulting (supposedly valid shape) assessed by calculating the length of the model perimeter (projected onto the image plane). Since the ideal length of a

valid shape should remain constant (in this case 66 pixels), any deviation from this constant can be used as a measure of the model's inability to reproduce valid shapes.

A number of random shapes were generated and passed through the model, the absolute difference from the ideal length recorded and the mean calculated over the test set. This procedure was then repeated for the constraint surface, a nearest neighbour approach and the cluster based NLPDM proposed here for varying numbers of clusters between *1* to *n* (where n equals the number of training examples). The procedure is outlined thus,

1. Take *n* random shapes X_i^{rand}
2. Project each X_i^{rand} into non-linear model and find closest reconstructed point X_i^{recon}
3. Calculate length in image plane of projected model X_i^{recon}, L_i^{recon}
4. Calculate length in image plane of any valid model X, L^{valid}

Calculate deformation error metric $e = \frac{1}{n}\sum_{i=1}^{n}\left|L_i^{recon} - L^{valid}\right|$ **Equation 5-4**

This error metric provides a zero error if the resulting reconstructed model is valid in shape. Therefore, the higher the error, the worse the performance of the constraints and hence the worse the performance of the model. By repeating this procedure for varying number of clusters between 1 (which is effectively a linear PDM) and 912 clusters (the number of training examples and therefore nearest neighbour), we can assess the advantage on model specificity as the number of clusters increases. Figure 5.7.1 shows the resulting graph from this analysis.

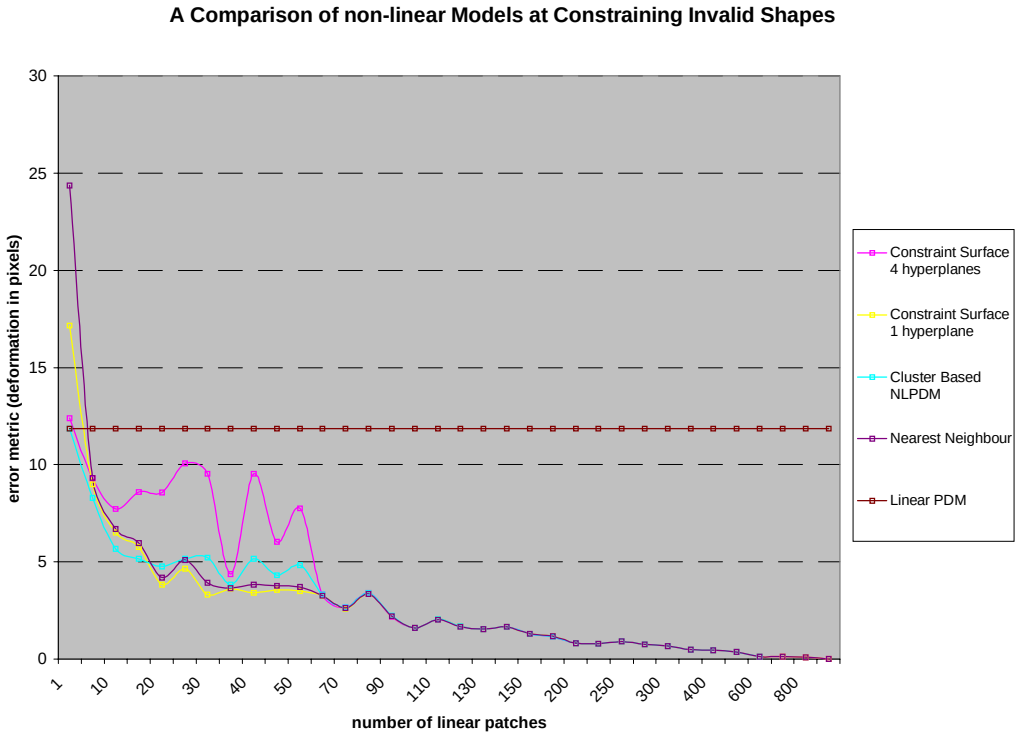


Figure 5.7.1 - Graph showing error rates of non-linear approximation techniques

The single hyper plane constraint surface, the nearest neighbour approach and the cluster based NLPDM all perform comparably and provide far lower error rates than either the multi-plane constraint surface or the linear PDM. However, the cluster based NLPDM (CB-NLPDM) provides lower errors until 5 patches are reached. With only a single linear patch the CB-NLPDM is effectively a linear PDM and as such does not produce errors that exceed the linear PDM. However, the other approaches produce significantly higher errors than even the linear model until sufficient patches have been introduced. As the number of clusters increases, so the error rate decreases, showing that the procedure does indeed increase the model's ability at representing non-linearity. The yellow trace on the graph shows the error results of the unconstrained surface approach of Bregler [Bregler 94] which, although performing slightly better between 25 and 70 patches, produces higher error rates at the pre-chosen patch number of 20 which was determined earlier from cluster analysis. It is important to note that this error graph confirms the results of the cluster analysis for the natural patch number, as further increases beyond 20 result in less significant results in the final model.

This confirms the conclusion that the approach for the selection of the natural number of clusters is valid, and hence the number of patches needed by the model is correct.

As the number of clusters increases to 912 (which is the number of examples within the training set) the error reaches zero. This is to be expected: when the number of clusters is equal to the number of training examples, each cluster contains only one member. The procedure then becomes a nearest neighbour approach. Since each nearest neighbour is in fact a valid training example, the validity of the shape is ensured, hence the zero error. This fact also explains the error results of the nearest neighbour approach which performs comparably to the other techniques. The question could be posed, why not use a nearest neighbour approach to perform the procedure simply and accurately? However, there are two issues, which have not as yet been considered.

1. The speed of the procedure increases as the number of linear patches (clusters) increases, as each patch is itself a linear PDM.
2. A nearest neighbour approach is only valid if every possible model pose is represented within the training set. This is often not the case and the power of the linear PDM is the ability to model shapes not present within the training set by linearly interpolating between examples.

It is therefore apparent that in order to consider the validity of any technique, two questions must be posed.

Does the model stop non-valid shapes from being produced?
(which has already been addressed in Figure 5.7.1)

Does the model allow valid shapes which were not present within the training set to be reproduced?

In order to answer this latter question a new set of experiments must be devised.

By constructing a new set of *n* examples that are all valid in shape and deformation not present within the training set (possible due to the synthetic nature of the test case), the ability of the CBNLPDM at reproducing unseen, valid shapes can be assessed. Using the same equation 5.4 (page 82) along with a Euclidean distance measure between the 'original valid but unseen data' and the 'reconstructed shape' this feature of the model can be assessed.

1. Take *n* valid shapes not present in the training set X_i^{new}
2. Project each X_i^{new} into non-linear model and find closest reconstructed point X_i^{recon}
3. Calculate the length in image plane of projected model X_i^{recon}, L_i^{recon}
4. Calculate the length in image plane of any valid model X, L^{valid}

Calculate deformation error metric $= \frac{1}{n}\sum_{i=1}^{n}\left|L_i^{recon} - L^{valid}\right|$

Euclidean distance error $= \frac{1}{n}\sum_{i=1}^{n} D\left(X_i^{recon} - X_i^{new}\right)$

Using these error metrics it would be expected that if the model were performing perfectly any valid shape projected into the model would have zero deformation error and zero Euclidean error. However, using the nearest neighbour approach would result in a zero deformation error but produce a high distance error. The result of performing this analysis on the data for both approaches is shown in Figure 5.7.2, Figure 5.7.3 and Figure 5.7.4. The test set consisted of examples generated from $\pm 38°$ angles and $17°$ intervals producing 135 valid, but unseen, examples with which to test the various models.

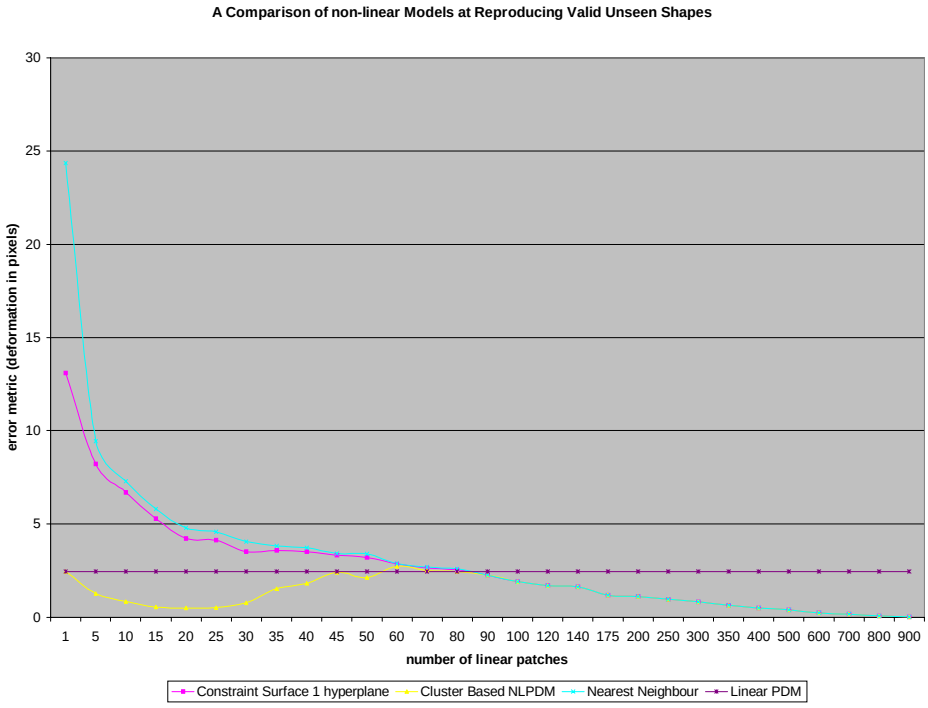


Figure 5.7.2 - Graph showing error rates of non-linear approximation techniques for Constraining Valid Unseen Data

Figure 5.7.2 shows the results generated via the deformation error metric for valid, but unseen, shapes applied to the various models. From this graph it can clearly be seen that the linear PDM produces a low baseline error of around 2.5 pixels deformation. This demonstrates the ability of the linear PDM to encapsulate the deformation of the training set, allowing valid shapes to be reproduced which were not present within the original data. It is not until in excess of 85 linear patches are used that either the nearest neighbour or constraint surface performs comparably to the linear PDM. The nearest neighbour approach generates the highest error rates as was suspected. The constraint surface with 4-hyperplanes produces the same results as the proposed NLPDM technique, both of which produce by far the lowest errors. Using 20 linear patches, both techniques produce their lowest error rates of approximately 0.5 pixels deformation, which again confirms the selection of the natural number of clusters for the data set.

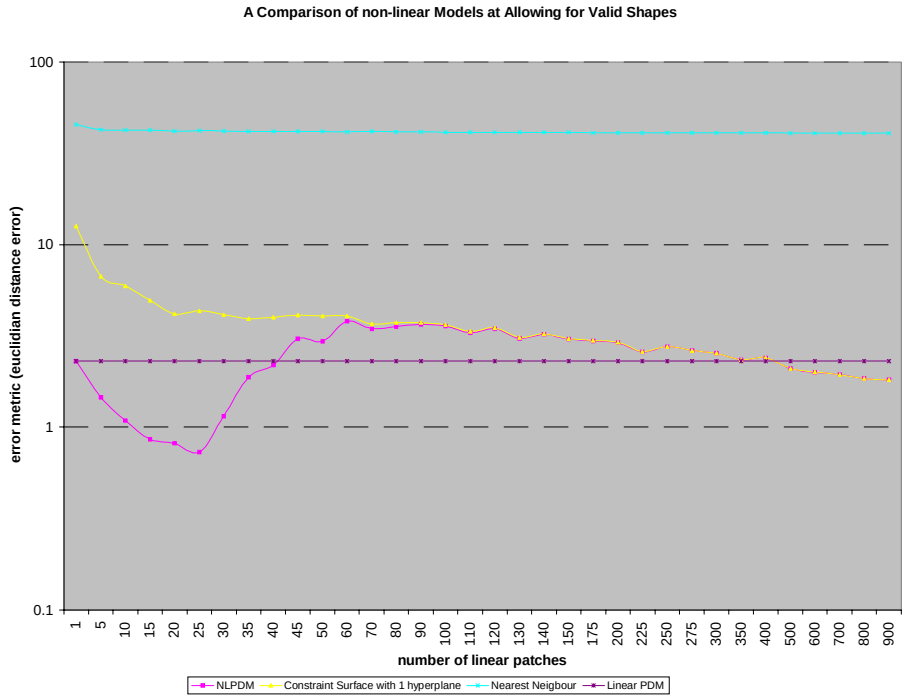


Figure 5.7.3 - Graph showing error rates of non-linear approximation techniques for Allowing Valid Unseen Data

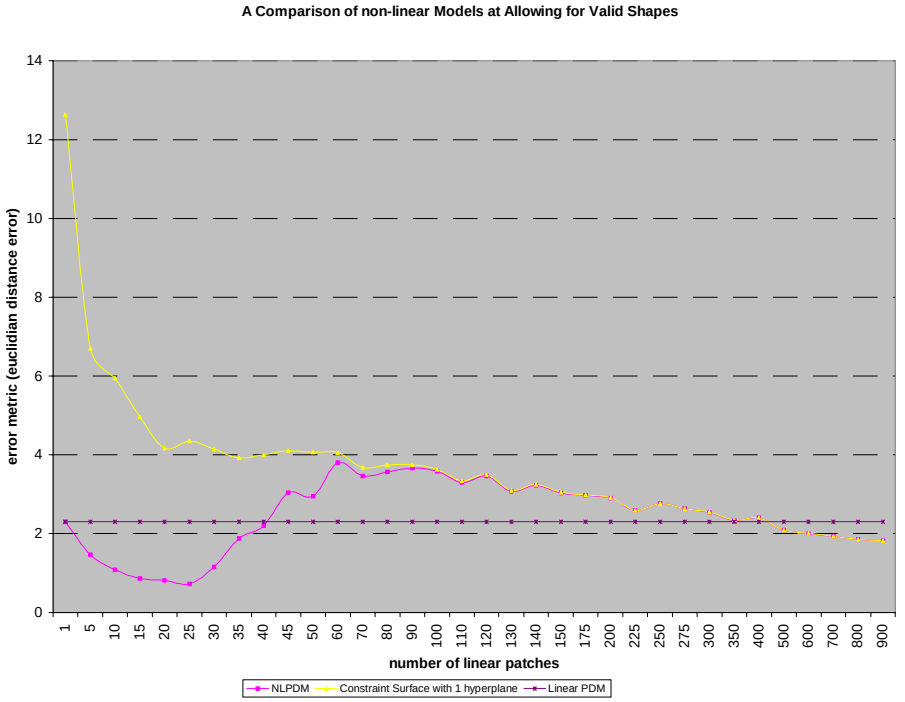


Figure 5.7.4 - Graph showing error rates of non-linear approximation techniques for Allowing Valid Unseen Data

Figure 5.7.3 shows the results generated via the Euclidean distance error metric for valid, but unseen, shapes applied to the various models. The figure uses a logarithmic scale due to the extremely high error rates produced by the nearest

neighbour approach. Figure 5.7.4 shows the same data (without the nearest neighbour approach present) on a linear scale. It can clearly be seen that the nearest neighbour approach produces error rates far in excess of any other approach. The linear PDM produces a low baseline error, which could be reduced further by increasing the number of modes of variation. The constraint surface with 1-hyperplane produces much higher error rates than the linear PDM and does not perform comparably with the linear PDM until around 450 linear patches, where each patch effectively has only two members. If a patch has only two members then it can have only one hyperplane, which means that more planes are required to model the data. This is confirmed by the 4-hyperplane approach which produces error rates identical to the NLPDM model, both of which produce errors of around 0.7-0.8 at the chosen number of clusters. If all these graphs are considered, the lowest errors are produced at 20-30 linear patches which suggests that the natural number may be slightly higher than was chosen. However, changing this number would result in little gain in accuracy.

Model Approach	Ability to Constrain Unseen Data	Ability to Constrain Valid Data	Ability to Allow Valid Data
Linear PDM	*BAD*	*POOR/GOOD*	*GOOD*
Nearest Neighbour	*GOOD*	*BAD*	*BAD*
Constraint Surface 1 hyperplane	*GOOD*	*BAD*	*POOR*
Constraint Surface 4 hyperPlanes	*POOR*	*GOOD*	*GOOD*
Cluster Based NLPDM	*GOOD*	*GOOD*	*GOOD*

Figure 5.7.5 - Table Showing Comparison of Techniques

If the performance of each technique is considered for each of the comparative studies performed, the conclusions can be summarised in a table, as shown in Figure 5.7.5. From this table it can be demonstrated that the proposed NLPDM approach produces superior performance in all aspects of modelling.

5.8 Conclusions

In conclusion, a NLPCA technique has been presented which models non-linearity by breaking the problem down into a set of linear models, which estimate high dimensional curvature. This has the advantages of the speed and simplicity of linear PCA, whilst providing a robust solution to object modelling. It has been shown how this technique performs in comparison to similar techniques and how the simple selection of model parameters can produce optimum solutions in the final model. These models have been shown to work on both low dimensional, high non-linear, and high dimensional, high non-linear problems where other procedures would fail.

Chapter 6

6 Cluster Constraints on Shape Space

6.1 Introduction

Thus far techniques have been discussed to project a non-linear data set into a lower dimensional space where further analysis is feasible. Once the shape space and its non-linearity have been estimated through cluster analysis, this segregation is modelled through multiple linear PDMs. The position and bounds of each linear patch is obtained by performing PCA on each extracted cluster and its members. The dimensional reduction allows the non-linear analysis (clustering) to be performed on high dimensional problems, but provides no added benefit to the final model. Each sub PCA cluster has the original dimensionality of the training set.

The inherent dimensional reduction of the linear PDM often provides a useful representation during classification. However, by breaking the original space up into linear patches this benefit of the model is lost. To provide static classification as demonstrated in [Bowden 96] a linear PDM formulation would still need to be maintained in addition to the composite model. This would not be the case if each patch of the composite model segregated the space in such a way

as to naturally aid classification. By retaining the dimensional reduction of the linear model throughout, and applying the constraints to the reduced data set, several advantages are achieved:

1. The dimensional reduction is retained throughout the model, providing a simplified model for classification.
2. For complex models where the number of clusters is high, the computational complexity of applying constraints is decreased.
3. Any noise within the model is filtered out by the linear PDM before constraints are applied.

The remainder of this chapter is concerned with the application of constraints to the dimensionally reduced data. Section 6.2 will discuss the application of these constraints. Section 6.3 will evaluate the approach and make comparisons with the previous chapter. Section 6.4 will demonstrate how this new model can be used in classification using sign language as an exemplar application. Section 6.5 will evaluate the performance of the proposed appraoch and lastly, conclusions will be drawn.

6.2 Constraining Shape Space

The basic procedure proposed in the previous chapter is outlined in Figure 6.2.1 where $\Re^N$ is the original dimensionality of the training set and $\Re^M$ is the reduced dimensionality of the training set after it has been projected down into the PCA space ($\underline{x}:\Re^M \rightarrow \Re^N$).

Previous work by the author and other researchers (see section 2.4) has shown how the reduced dimensionality of PCA space is invaluable in the classification of static poses of the model. Indeed, this is often used as an important tool in classification. It is therefore beneficial to combine these techniques in the modelling of a non-linear data set.

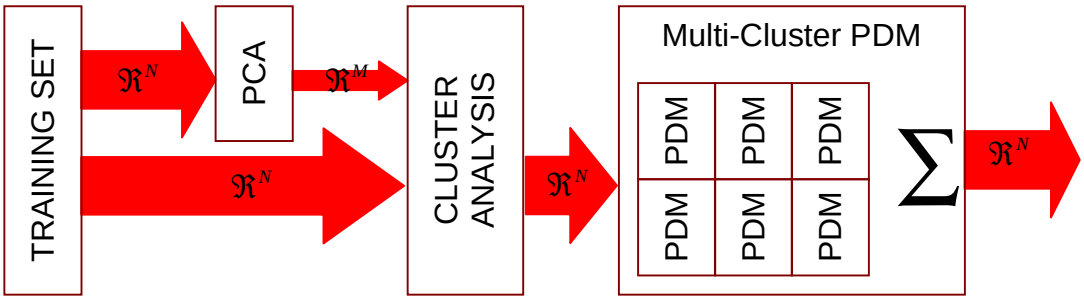


Figure 6.2.1 - Cluster Based non-linear PDM

Figure 6.2.2 gives an overview of this new approach, which will be referred to as '*Constraining Shape Space*' or **CSSPDM**. In this procedure the dimensional reduction of the PCA is retained throughout the entire model. In addition to the cluster analysis, PCA is performed on each cluster in the dimensionally reduced space, constraining the model in PCA space. The model must then be projected back up into the original dimensionality to extract the final shape.

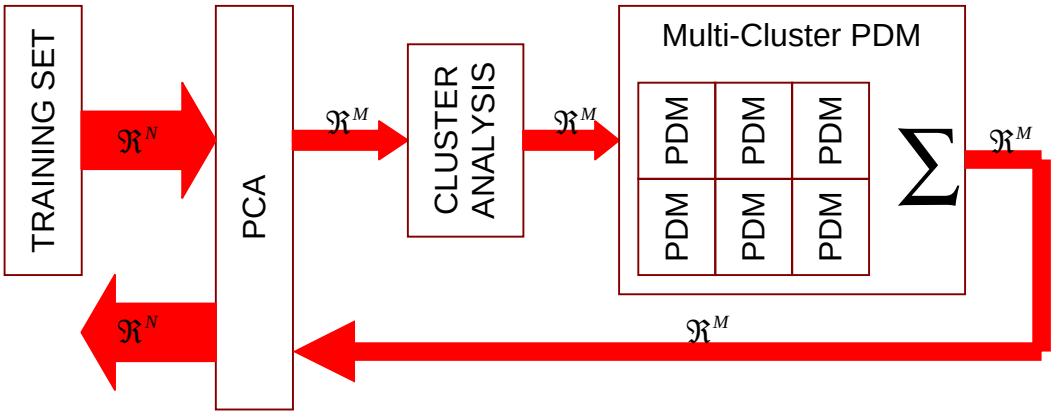


Figure 6.2.2 - Cluster Based non-linear Constraints on Shape Space

An algorithmic overview is given below.

1. Perform PCA on training set

2. For each training example, do:

Project training example onto eigenvectors, recording distance from mean.

Concatenate these distances into a reduced dimensional vector.

3. Perform cluster analysis on dimensionally reduced data set to determine natural number of clusters present

4. Use this natural number to segregate the data set into multiple clusters using fuzzy k-means

4. Perform PCA on each cluster of training set

> *PCA is performed on the reduced dimensionality cluster. Here models must be transformed to the reduced space at runtime, the closest allowable shape from the model reconstructed and transformed back to the original dimensionality.*

6.3 Evaluation

In principle, this procedure should produce identical results to that produced by applying the constraints to the original training set, with the added advantage of the computational saving of performing the constraints within the reduced space. However, in practice this approach performs better due to the data smoothing effect of the initial linear projection, which reduces the dimensionality. Each linear patch has a far lower dimensionality, hence the linear patch can be modelled to encompass all the deformation. The initial linear projection is where the data smoothing (lossy compression) occurs and as such the model's accuracy is limited by this single factor.

In order to assess the performance of the technique the experiments detailed in chapter 5.7 can be repeated and the error graphs produced. Returning to the robot arm example (chapter 5.5.1), after the initial dimensional reduction from $\Re^{32}$ to $\Re^{4}$ the reduced dimension training set is fuzzy-clustered in the same manner. From this data clustering PCA is performed on each linear patch in $\Re^{4}$ space. As the maximum number of eigenvectors for each cluster cannot exceed the dimensionality of the space, each cluster is constructed so as to encompass 100% of the deformation (i.e. all four modes are used). This means that no decisions need be made for the dimensionality of individual clusters and therefore simplifes the procedure of model construction.

The error metrics previously defined (section 5.7) are now used to assess the new model's ability at both reproducing valid shapes and constraining non-valid shapes.

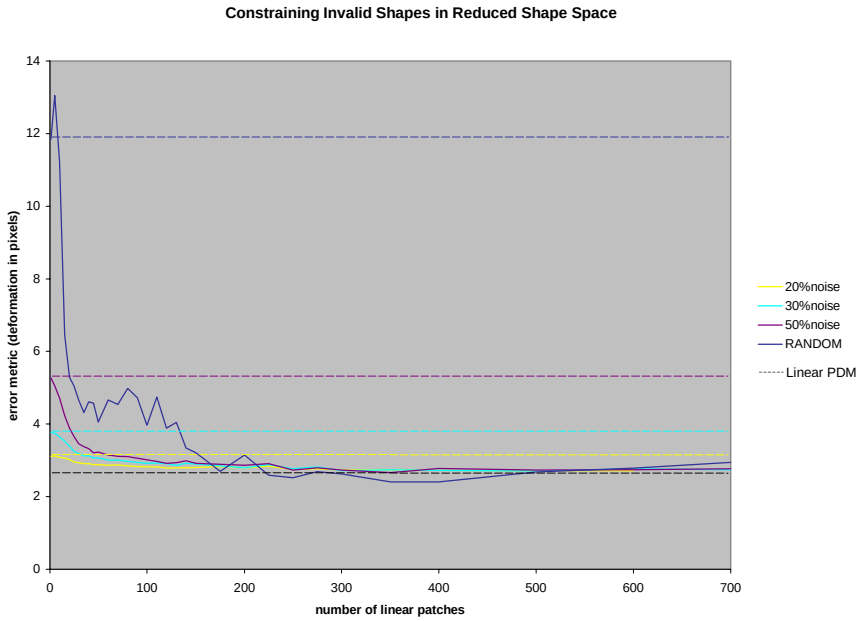


Figure 6.3.1 - Error graph showing ability to constrain non-valid shapes

Figure 6.3.1 demonstrates the result of measuring the performance of this new technique on the non-valid test set described in the previous chapter for increasing numbers of linear patches. This procedure was repeated for the original training set perturbed by noise and for a completely random training set. As would be expected, increasing the number of linear patches decreases the error rate and hence results in less invalid deformation being produced.

The dotted lines show the error produced by the single linear PDM in comparison to each of the data sets. The linear PDM produces identical results to that of using a single linear patch in the CSSPDM. The single cluster contains 100% of the deformation of the reduced data and therefore has no effect. Due to the dimensional reduction of the linear PDM being the same as that used in reducing the dimensionality of the data, the single cluster has no effect as they are essentially the same. However, because the linear PDM remains present throughout the technique the resulting loss of data ensures that no level of

constraints can perform better than the 2.7 error rate produced by the information loss of the data projection. Surprisingly however, for models with between 200 and 500 linear patches, the technique does produce higher accuracy rates but only in the order of fractions of a pixel. By altering the dimensional reduction to utilise more eigenvectors in the initial projection and hence retaining more information from the model, this baseline error can be reduced further. However, this poses the same question as the linear PDM and the trade-off between accuracy and compactness/robustness (see section 3.2.6).

The important features of Figure 6.3.1 are that the error rate is significantly reduced by increasing the number of linear patches initially, and the most benefit can be deemed to be at around 20 linear patches which correlates with the initial analysis of the data set.

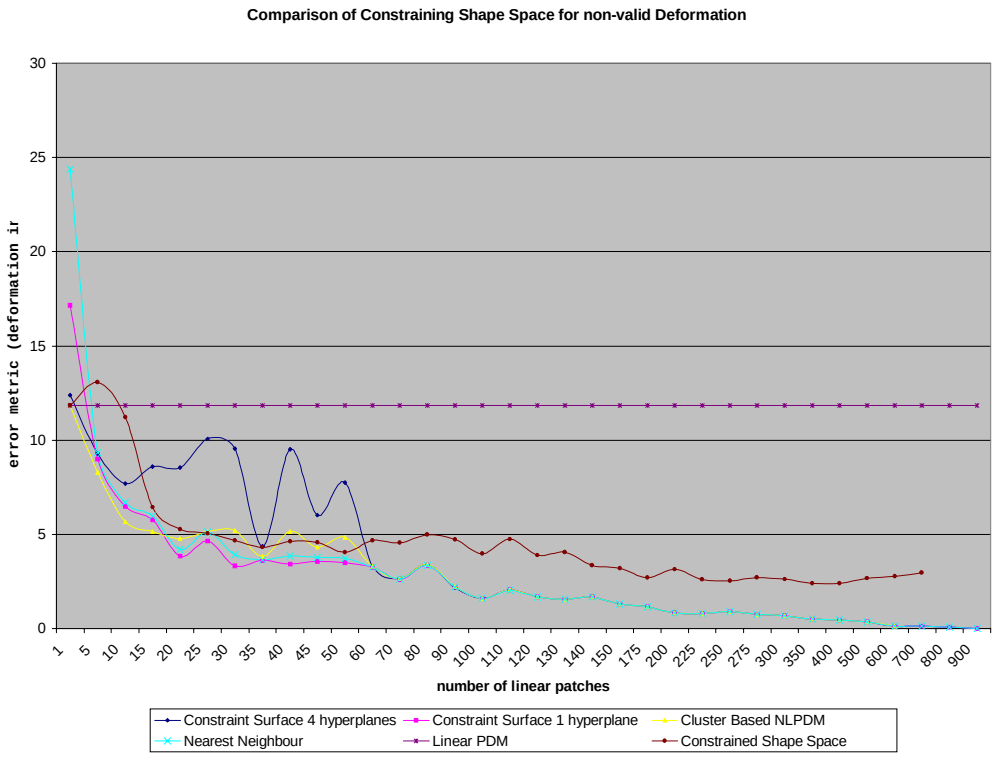


Figure 6.3.2 - Error graph showing comparison of Constraining Shape space against previously discussed Techniques

Figure 6.3.2 shows the error line produced from the random data set in Figure 6.3.1 superimposed upon the results of the previously discussed approaches from section 5.7. It can clearly be seen that although *Constraining Shape Space* does

not produce the lowest error rates, it does perform comparably with the lower of the error plots generated by other techniques. Since it has already been established in the previous chapter that the CBNLPDM produces the most desirable results, the comparative performance of this solution is of primary concern. The data smoothing of the dimensional reduction can be attributed to the smoothed error graph produced by this technique. Although the error rate does not reach zero, like many of the other approaches, it follows the same trend until more than 60 linear patches are used. Since the model only utilises 20, this artefact of the approach can be disregarded, as model complexity would never reach this level. It is also important to bear in mind that the minimum error of the *Constrained Shape Space* approach can be reduced further by reducing the information loss of the dimensional reduction (the initial linear PDM projection) and including more information in the model i.e. using more eigenvectors.

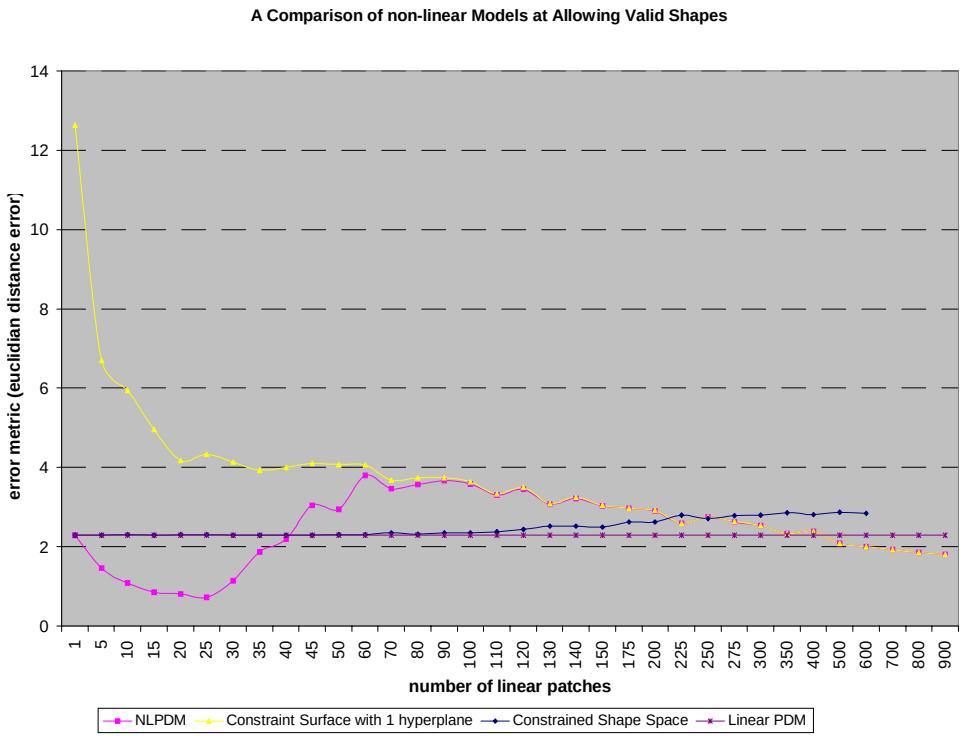


Figure 6.3.3 - Error graph showing ability to model valid shapes

Figure 6.3.3 shows the result of measuring the performance of this new technique upon the valid unseen test set (described in the previous chapter) for increasing numbers of linear patches. The performance would be expected to be comparable with the linear PDM model (as the initial projection is a linear PDM). Although

the performance is not as high as the CBNLPDM, it performs significantly better than the constraint surface or nearest neighbour approach which are not shown on this graph due to the extremely high error rates they produce (around 45). In a similar manor to the linear PDM, the CSSPDM error rate can be further reduced by reducing the dimensional reduction of the initial projection to include more deformation. However, this in turn will increase the dimensionality of the model and hence computational complexity in analysis and runtime application. When the huge dimensional reductions that can be achieved for analysis are considered, this slight degradation in performance can be justified. In this example the reduction from 32 to 4 may not be considered advantageous but when larger dimensional examples are considered (examples in next section and later chapters) the benefits of this approach can be seen.

To summaries these techniques,

An algorithmic overview is given below.

1. Perform PCA on training set

2. For each training example do

Project training example onto eigenvectors, recording distance from mean.

Concatenate these distances into a reduced dimensional vector.

3. Perform cluster analysis on dimensionally reduced data set

4. Perform PCA on each cluster of training set

When performing PCA on individual clusters two approaches can be taken.
(1) PCA can be performed on the reduced training set cluster. Here models must be transformed to the reduced space at runtime, the shape reconstructed and transformed back. This is slightly more computationally expensive, but has the advantage that the original encoding remains and therefore aids simple pose analysis/recognition.
(2) PCA can be performed on the original training set clusters after the clusters are transformed back into the original space. This technique is slower in analysis but faster at runtime and ensures that little high frequency information is lost.

6.4 Classification

6.4.1 Introduction

Due to the nature of constraining shape space, much of the segregation of the data set which is important to classification is contained within the model. In addition to this, the improved modelling capability of the non-linear estimation allows more complex problems to be tackled. If the assumption is made that similar poses of a model produce similar training vectors and each pose of the model corresponds to a point in shape space, it is therefore a fair assumption that similar poses of the model will produce tight clusters within this shape space. These clusters should automatically be modelled by the non-linear constraints that are placed on the model and facilitate more complex static pose recognition. The application of gesture recognition provides an ideal application for the proof of this assumption.

6.4.2 Sign Language & Gesture Recognition

American Sign Language or ASL has a finger spelt alphabet similar to other national sign languages. These simple gesture alphabets are used to spell names or words (letter by letter), for which there is no signing either known or present in the vocabulary. ASL provides a more suitable problem domain over British Sign Language as the BSL finger spelt alphabet is a two-handed system. Although this two handed system in reality provides a method of signing which is far easier to understand, it presents added difficulty for computer vision tasks due to the problems associated with occlusion.

Watson presented a review of work related to hand gesture interface techniques which consisted of glove sensor-based techniques, vision-based techniques and the analysis of drawing gestures [Watson93]. These were later summarised and techniques evaluated in by Handouyahia, Ziou and Wang [Handouyahia 99] and are discussed later in this chapter.

Figure 6.4.1 shows the ASL[2] alphabet with images taken from the training set.

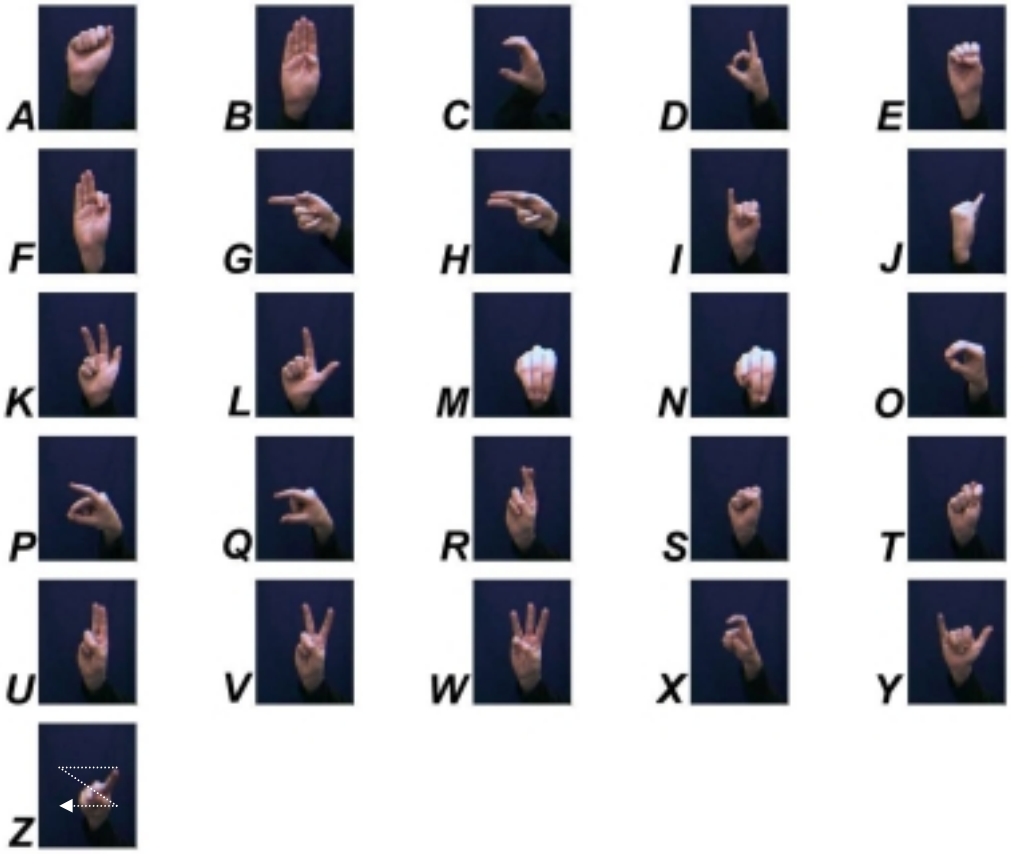


Figure 6.4.1 - The American Sign Language Finger Spelling Alphabet

It can clearly be seen from Figure 6.4.1 that each letter of the alphabet corresponds to a specific pose of the hand, with the exception of the letter 'z' which is a dynamic gesture and requires movement. This being the case, each gesture should occupy a distinct area in shape space.

6.4.3 Constructing the Non linear Hand Model

Several image sequences were recorded which encapsulated numerous occurrences of each of the letters of the alphabet. These sequences included three 'runs' through the alphabet, along with a small selection of simple sentences and words. These image sequences were recorded using a blue backdrop and sleeve to allow simple extraction using chroma key techniques.

[2] American Sign Language alphabet is almost identical to the alphabet of International Sign Language (ISL).

Once these sequences had been extracted, the hand was segmented to produce a binary image, and a contour-tracing algorithm initiated to extract the external contour of the hand for each image frame. Figure 6.4.2 shows: (a) a sample image frame of the hand; (b) the binary image produced from chroma keying; (c) and (d) the resulting extracted boundary. The procedure was then repeated for every image frame, providing training examples of the hand as it moves throughout the alphabet and the possible shapes it can take as it makes transitions between the letters.

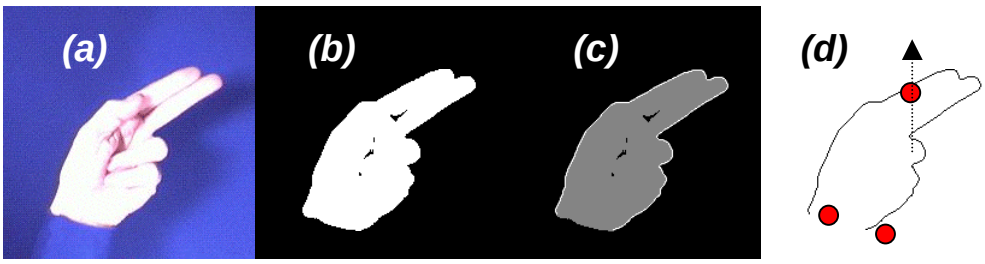


Figure 6.4.2 - Extracting Training Examples for ASL Data Set

(a) Hand image, (b) Segmented hand, (c) Extracted Contour (d) Resampled Contour

Before any statistical analysis can be performed, the training examples must first be resampled and aligned. The contour was automatically allocated 3 landmark points around the contour as shown in Figure 6.4.2(d). These landmark points were allocated at the start and finish of the contour and one at the vertical extremity within a 10° arc of the centeroid of the boundary. Once done, these landmarks were used to resample the boundary using linear interpolation to produce a contour consisting of 200 connected points. The low number of landmark points and the simple landmark identification used guarantees that non-linearity through non-optimum landmark point assignment will be present within the training set. However, this non-linearity will be modelled through the use of the *Constrained Shape Space* non-linear model discussed earlier. No rotational alignment was performed to preserve as much information about the pose of the model within the shape space. This again would introduce non-linearity into the model. The rotation non-linearity is necessary in the recognition of gestures.

Poses produced by the dynamic gesture ('z' for example) are similar to other gestures ('g') except for the rotation of the hand pose around the camera's z-axis. If this rotation were to be removed, then the distinction between these two poses would be lost. Again the non-linear constraints will model this non-linearity and allow simple distinctions to be made.

Finally any translation of the hand model in the xy image plane was removed by translating the origin of the contour to that of the wrist, located by taking the mean of the start and finish points of the contour. This approach removes any translation of the hand in the image plane, but assumes that the hand is kept at a consistent distance from the camera throughout the training set and hence has no need to be scaled.

Once the training set had been prepared, a total of 7441 example contours were produced and labelled with the actual letter the pose corresponded to. Poses that were deemed transitory poses between real gestures were labelled as ***null*** gestures.

Under the normal procedure for the construction of a PDM, the last phase before PCA is performed would be to normalise all contour boundaries, ensuring a consistent training set. However, for reasons that have already been mentioned with regard to rotation, it is important that this information is preserved. Theoretically the length of vectors on which PCA is performed should not affect the resulting model except for its overall size. However, due to the nature of floating-point arithmetic and the problems associated with overflow errors, it is still necessary to reduce the size of the computations. This is facilitated by dividing each training vector not by its own length (as in normalisation), but by the length of the mean vector of the training set. This effectively normalises the training set but retains any subtle size deviations between examples.

6.4.4 The Linear ASL Model

The Linear ASL model is now generated by performing linear PCA upon the training set. Figure 6.4.3 shows the primary modes of the linear ASL PDM and how these modes deform the model from the mean.

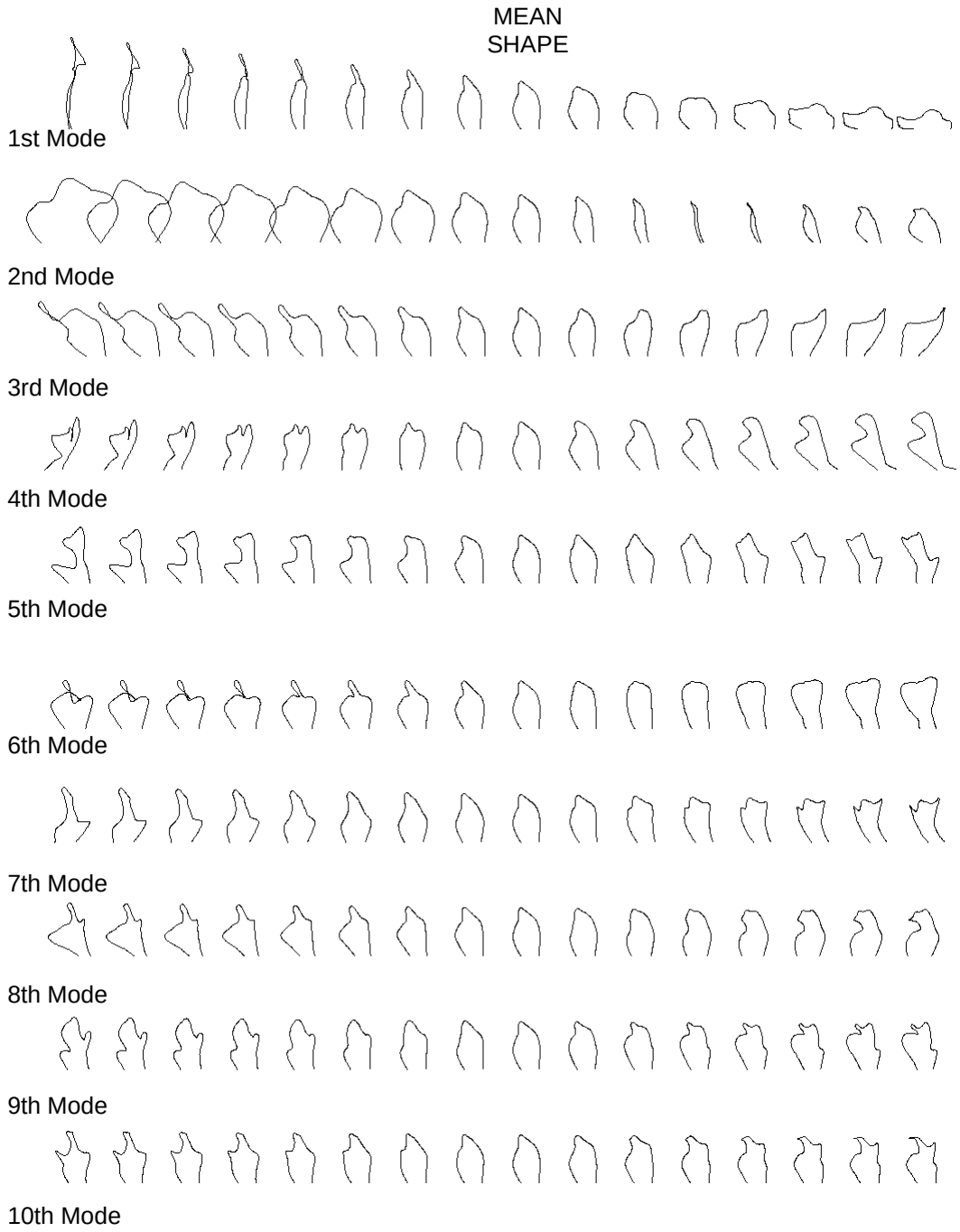


Figure 6.4.3 - The linear ASL PDM Model

It can clearly be seen that the major modes of variation include large amounts of deformation which, when put together, will produce an unreliable model capable of producing far too much deformation (see examples in Figure 6.4.4)

By analysing the eigenvalues of the covariance matrix it can be determined that the first 30 eigenvectors corresponding to the 30 largest eigenvalues encompass

99.6% of the deformation within the model. Unfortunately, due to the natural rotational non-linearity and high order non-linearity which has been introduced into the model during re-sampling (as discussed in the previous section), this linear model is unsuitable for tracking and classification. Figure 6.4.4 shows a selection of invalid shapes that can be constructed from the linear ASL PDM. These examples were produced by generating random vectors that were within the bounds of the linear model. It is the linear PDM's ability to allow invalid shapes which make the model unreliable for tracking and classification. These invalid deformations are due to the linear approximation of the non-linear data set.

Figure 6.4.4 - Example Invalid Shapes produced by the linear ASL PDM

6.4.5 Adding non-linear Constraints

Using the procedure previously outlined, non-linear constraints to the model are added by performing cluster analysis on the dimensionally reduced data set after it has been projected down into PCA space. From the linear model it has been determined that the 30 primary modes encompass 99.6% of the deformation, by projecting each of the training vectors down into this space (as previously described), a dimensional reduction of 400 to 30 is achieved. Cluster analysis is now performed upon the reduced data set.

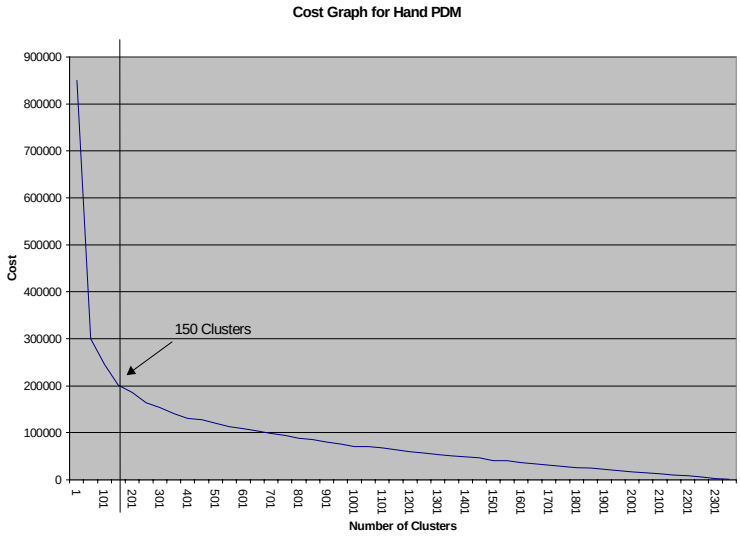


Figure 6.4.5 - Cluster Analysis on Dimensional Reduced ASL Training Set

Figure 6.4.5 shows the resulting cost graph from the cluster analysis of the reduced data set and the natural number of clusters estimated to be 150. The fuzzy k-means algorithm is then used to segregate the space into 150 clusters. These clusters are then learnt by performing PCA on their members.

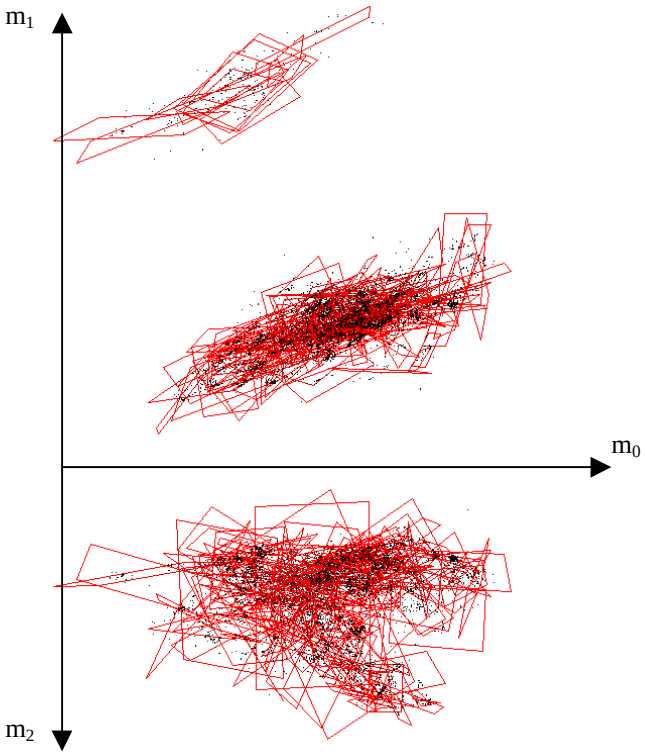


Figure 6.4.6 - Constrains on PCA space for the ASL Model

Figure 6.4.6 shows the PCA space for the model projected into 3 dimensions for visualisation purposes, with the constraints shown as the bounding boxes (first two primary modes) of the linear patches (clusters) extracted via PCA. Notice the two distinct clusters produced in the direction $m_{1,}$ meaning that the shape space is discontinuous and there is no smooth path between the two distinct areas of shape space. This is due to the simple landmark identification and the problems associated with it. Further discontinuities may exist in the model which are not apparent in the dimensions that are shown in Figure 6.4.6. These types of spaces and solutions to the problems they introduce will be discussed in the chapter on temporal dynamics (specifically sections 7.3-7.4 for the ASL shape space)

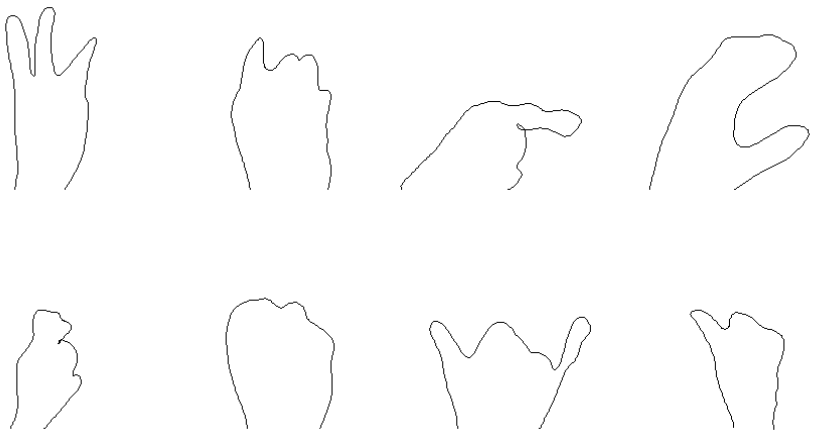


Figure 6.4.7 - Example Shapes Produced by the constrained non-linear ASL PDM

Figure 6.4.7 shows random shapes generated within the constrained model, If these are compared with those produced in Figure 6.4.4, it can be seen that the constrained model contains far less invalid deformation and therefore results in a more reliable model for tracking. Each random shape is also very close to a natural gesture in ASL and it is this correlation between cluster and gesture that can be used to perform gesture recognition.

6.4.6 Recognising Gestures

Ideally for an alphabet with 26 characters, the model would contain 26 clusters, where each cluster directly corresponds to a specific letter. However the non-linearity of the model requires far more clusters to encompass the deformation reliably. As a result, multiple clusters may correspond to a single letter. This is due to:

1. The presence of null (transitional) poses of the hand within the training set should not correspond directly to any specific letter. As these null poses will be distributed throughout the space it is incorrect to assume that it is possible to model them with a single cluster.
2. The landmark point assignment used may result in two very similar poses of the model occupying completely different areas of the PCA space (i.e. discontinuous shape space). Therefore, again, it is incorrect to assume that any single letter will produce a single tight cluster.
3. The presence of dynamic gestures like 'z' requires movement of the hand to complete the gesture. This movement results in a trajectory in PCA space that corresponds to a letter rather than a cluster. This trajectory may require multiple clusters in order to model the deformation.

Once these issues are considered it is apparent that in order to classify any specific gesture, multiple clusters must be assigned to each letter rather than single clusters as previously used in previous work by the author [Bowden 96; Bowden 97]. This can be achieved by analysing the training set and probabilistically assigning each cluster to a specific letter. This provides a conditional probability that the model represents a letter given that model is in any specific cluster. These conditional probabilities are constructed in a probability matrix as shown in Figure 6.4.8.

	$letterA$	$letterB$	$\cdots$	$letterZ$	$null$
$Cluster_1$	p_{C_1A}	p_{C_1B}	$\cdots$	p_{C_1Z}	p_{C_1null}
$Cluster_2$	p_{C_2A}	p_{C_2B}	$\cdots$	p_{C_2Z}	p_{C_2null}
$Cluster_3$	p_{C_3A}	p_{C_3B}	$\cdots$	p_{C_3Z}	p_{C_3null}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$Cluster_{149}$	$p_{C_{149}A}$	$p_{C_{149}B}$	$\cdots$	$p_{C_{149}Z}$	$p_{C_{149}null}$
$Cluster_{150}$	$p_{C_{150}A}$	$p_{C_{150}B}$	$\cdots$	$p_{C_{150}Z}$	$p_{C_{150}null}$

Figure 6.4.8 - Probability Matrix for ASL Classification

As each of the vectors from the training set has been pre-assigned a letter which provides a label for each shape of the training set, the matrix can be constructed by calculating which cluster a specific training example belongs to, and assigning that cluster to the labelled letter. Each training example is projected down into the PCA space and the closest cluster, α, located. The value along the row α, $P_{\alpha\beta}$ which corresponds to the letter β is then incremented. This procedure is carried out for the entire training set and each row normalised to calculate the conditional probability that any cluster belongs to a letter i.e. $\sum_i P(letter|Cluster_i) = 1$. Now by locating which cluster the model exists in there is a conditional probability that the model is representing a letter, with the highest probability for a cluster representing the most likely letter. By analysing this matrix information about how this correlation is achieved can be extracted. Table 6.4-1 shows how many clusters each letter uses in this mapping.

Letter	N' Clusters	Letter	N' Clusters
a	23	O	16
b	9	P	9
c	8	Q	10
d	9	R	15
e	26	S	15
f	13	T	20
g	14	U	15
h	13	V	5
i	12	W	2
j	11	X	7
k	7	Y	9
l	4	Z	26
m	11	NULL	130
n	8		

Table 6.4-1 - Correlation between ASL Gestures and Clusters in non-linear Model

Cluster N'	N' Letters	Cluster N'	N' Letters	Cluster N'	N' Letters	Cluster N'	N' Letters	Cluster N'	N' Letters	Cluster N'	N' Letters
1	4	27	2	53	4	79	3	105	4	131	2
2	2	28	3	54	3	80	3	106	2	132	5
3	2	29	3	55	1	81	6	107	2	133	4
4	3	30	4	56	2	82	3	108	2	134	2
5	4	31	1	57	2	83	2	109	4	135	1
6	2	32	2	58	2	84	2	110	5	136	7
7	5	33	2	59	7	85	3	111	3	137	2
8	4	34	3	60	3	86	2	112	2	138	2
9	5	35	1	61	1	87	2	113	2	139	3
10	5	36	3	62	2	88	5	114	3	140	3
11	2	37	2	63	2	89	2	115	4	141	2
12	4	38	3	64	1	90	5	116	3	142	2
13	2	39	4	65	3	91	3	117	4	143	2
14	3	40	3	66	4	92	5	118	5	144	0
15	1	41	4	67	6	93	1	119	4	145	5
16	4	42	3	68	2	94	1	120	3	146	3
17	3	43	2	69	4	95	2	121	2	147	3
18	6	44	4	70	2	96	2	122	2	148	4
19	6	45	2	71	4	97	4	123	2	149	5
20	4	46	4	72	6	98	4	124	3	150	4
21	3	47	2	73	3	99	4	125	2	**Average**	**2.98**
22	1	48	2	74	3	100	4	126	3		
23	1	49	7	75	1	101	1	127	3		
24	1	50	2	76	4	102	2	128	4		
25	2	51	5	77	3	103	3	129	1		
26	1	52	3	78	3	104	3	130	1		

Table 6.4-2 - Correlation between Clusters of non-linear model and ASL Gesture

Table 6.4-2 shows the number of ASL gestures that correspond to each cluster. It would be expected that each cluster would correspond to only one letter, however due to inconsistencies in labelling and the complexity of the model this is not the case. The average cluster corresponds to 2.98 letters, but the matrix gives us a probability that the cluster corresponds to a specific letter; The highest probability entry in the matrix gives the best estimate to the recognised letter.

	Highest Probabilistic Match	Second Highest Probabilistic Match
Minimum	0.285714	0
Maximum	1	0.454545
Mean	0.706031	0.210881

Table 6.4-3 - Analysing the Resulting Probabilities

Table 6.4-3 shows the range of probabilities that result for this procedure. Using an unseen test set of segmented hand shapes with (hand labelled) letter ground truth for comparison, the average probability for the best match of the matrix is around 0.7. The maximum value of 1 demonstrates that some clusters exclusively belong to specific gestures and this can be confirmed by the presence of clusters assigned to only one cluster in Table 6.4-2. The next highest probability from the matrix is also shown with the mean value being much lower than that of the best match, demonstrating that although there is some ambiguity between gestures there is significant distinction probabilistically as to the function of each cluster.

By comparing the resulting highest probability match with the original labelled letter for each of the training examples and converting this to a percentage, a measure of the classifications accuracy can be determined.

Out of a total of 4741 examples the highest probability match was correct in 3348 cases, with the second highest probability match being correct in 1000 cases. This gives a 70.62% accuracy for the most likely match, with 20.09% accuracy for the next most likely match. From this it can be said that there is a

91.71% chance that the correct letter for each pose will be recognised as one of the two highest probability matches from the matrix.

6.5 Evaluation

Initially these results may not seem overwhelming, however the complexity of performing such a task using computer vision is considerable due to the variability of the hand and the problems associated with accurately segmenting or extracting features which represent its shape. If other approaches are considered this becomes apparent. Table 6.5-1 [Handouyahia 99] summarises other authors approaches the problem.

Authors/ Properties	Size of Vocab	Type of Vocab	Capture	Representation	Recognition	Success Rate %
Gourley[3]	26	ASL[4]	Elect[5]	Templates	Perceptron Neural Network	95
Harling[3]	5	ASL[4]	Elect[5]	Templates	Perceptron Neural Network	96
Murkami[3]	42	JSL[6]	Elect[5]	Templates	Perceptron Neural Network	98
Takahashi[3]	46	JSL[6]	Elect[5]	Joint and orientation coding	Template Matching	65
Gao[3]	13	D.Set[7]	Camera	Convex/Concave coding	Backpropogation Network	80
Uras[3]	25	ISL[8]	Camera	First size functions family	K-Nearest Neighbour	85
Uras[3]	25	ISL[8]	Camera	Second size functions family	K-Nearest Neighbour	86
Freeman[3]	15	D.Set[7]	Camera	Orientation Histograms	K-Nearest Neighbour	75
Handouyahia[3]	25	ISL[8]	Camera	Moment Based Size Functions	Perceptron Neural Network	90
Our Method	26	ASL[4]	Camera	NL Point Distribution Model	Fuzzy Nearest Neighbour	71(92)

Table 6.5-1 - Table Showing a Summary of Gesture Recognition Methods

The highest accuracy rates are achieved using an electrical sensor based data glove as an input device. Those techniques that rely upon computer vision perform less well. The higher accuracy's are also generated for systems which use neural networks to provide the mapping between feature space and gesture space. If the simplicity of the CSSPDM augmented with the conditional probabilities which provide the gesture recognition is considered then the attraction of this approach becomes apparent.

[3] Details of the authors work are contained in and Handouyahia 99 and Watson 93
[4] American Sign Language
[5] Electronic sensor based glove
[6] Japanese Sign Language
[7] The type of the vocabulary is pre-defined
[8] International Sign Language

It is also important to note that the CSSPDM is assessing the model at every frame and attempting to recognise the gesture contained there in. This assessment of each frame is static. No temporal or contextual information is used. Further constraints could be applied from the English Language to increase accuracy (see Chapter 7). Since humans tend to pause slightly at each gesture, the accuracy could be further increased by accumulating probabilities over time, i.e. consecutive frames would 'vote' towards the current gesture, further reducing the effect of noise.

Selected Feature/Criteria	Scale Invariant	Translation Invariant	Rotation Invariant	Lighting Invariant	Robust to N' of Fingers	Computational Complexity
Basic Chain Code[3]	No	Yes	No	No	No	Low
Convex-Concave Coding[3]	Yes	Yes	Yes	No	Yes	Low
Fourier Desc.[3]	No	No	No	No	No	Low
Hu Invariant Moments[3]	Yes	Yes	Yes	No	No	High
Alt Invariant Moments[3]	Yes	Yes	No	No	No	High
Principal axes[3]	Yes	Yes	No	No	No	Low
Grey Level Histogram[3]	No	Yes	Yes	No	No	Low
Hist. Of Local Orientation[3]	Yes	Yes	No	No	No	Low
Size Functions[3]	Yes	Yes	No	Yes	Yes	High
Moment Based Size Funct[3]	Yes	Yes	No	Yes	Yes	Low
Authors Method[3]	Yes	Yes	Yes	Yes	Yes	Low

Table 6.5-2 - Table Showing the Evaluation of Features used in Various Gesture Recognition Methods

The CSSPDM naturaly lends itself to the probabilistic classification of pose, however if the CSSPDM is compared to other features used in Gesture Recognition, its benefits can clearly be seen. Table 6.5-2 [Handouyahia 99] summarises features used by other methods.

Unlike other approaches the CSSPDM is:

1. **Scale Invariant**: Gestures can be executed by different people with different hand sizes.
2. **Translation Invariant**: The location of the hand in the image plane can change.

3. **Rotation Invariant**: The hand can rotate around the cameras z-axis, other rotations of the hand can be incorporated into the deformation of the model.
4. **Lighting Invariant**: The illumination and background of the scene can change.
5. **Robustness to number of fingers**: Additional training data can be incorporated into the model to allow for individual changes in hand shape and gesture.
6. **Computation Complexity**: The simplicity of the linear mathematics and single layer of conditional probability means the method is fast to compute.

6.6 Conclusions

This chapter has demonstrated that by projecting the dataset through a linear PDM and hence reducing the overall dimensionality of the problem before further non-linear constraints are applied, several benefits are gained:

1. The data is smoothed before constraints are applied, producing better results in the final model.
2. The data reduction of the CSSPDM produces a significant computational saving over the CBNLPDM at the cost of accuracy. However this accuracy can easily be controlled to ensure model precision is maintained.
3. Construction is simplified as only one decision need be made as to the information loss of the model. In CBNLPDMs each cluster requires a different number of eigenvectors to achieve the required accuracy while compressing the data. However, CSSPDMs need not be concern with the local dimensionality of clusters as the initial projection allows each linear patch to model 100% of the deformation of that cluster.

Furthermore, it has been shown that, although the nature of the space is complex, simple classification techniques can be applied to perform static recognition of object shape and pose. These models allow deformable models to be constructed which, under the linear constraints of a simple PDM, would fail to be robust enough for “Real World” applications.

One important consideration is that as models become more complex, the simple gradient descent approach used on linear models begins to fail. These issues will be addressed in the next chapter.

Chapter 7

7 Adding Temporal Constraints

7.1 Introduction

The deformation that has been 'learnt' thus far is time independent deformation. Models have been constructed that know **what** is valid deformation but not **when** deformation is valid. This important temporal constraint is beneficial in disambiguating models. When such mathematical constraints have been placed upon the deformation of an object in order to increase robustness, the important consideration of how a model moves with time should also be considered.

The linear formulation of the PDM makes iterative movements within the image frame based upon the assumption that the model will not alter considerably between consecutive frames. Providing a simple model and a slow moving/deforming object this assumption holds true. However, as has been demonstrated with non-linear models, this smooth iterative movement through shape space does not provide a sufficient mechanism to 'jump' between discontinuities in shape space. It is therefore apparent that if complex models are to be successfully tracked within the image frame, additional constraints must be applied to both increase robustness and to improve the transition through shape space.

The remainder of this chapter is concerned with the construction and use of temporal dynamics, which can be learnt in addition to deformation. Section 7.2 takes a graphical simulation example to construct a 3D non-linear PDM from which temporal dynamics are learnt. These dynamics can then be used to reproduce the deformation and motion of the model. Section 7.3 will discuss the issues of tracking complex non-linear models and how these temporal dynamics can be used to increase robustness and support multiple hypotheses. Section 7.4 demonstrates how these temporal constraints can be used to enhance classification. Lastly conclusions are drawn.

7.2 Learning Temporal Model Dynamics

7.2.1 Introduction

The work thus far has discussed the computer vision applications of non-linear models of shape and deformation, where models have been used to locate and track objects in the image frame. The models produce graphical representations of objects, which can be mapped to the appearance of real world objects within the image. In the field of computer graphics, similar representations are required for animation. The main difference is that graphical models are required to be 'life-like' and three-dimensional for rendering. The models must therefore exist in 3D. The rendering procedure then projects these models into 2D for viewing. In computer vision applications this projection is often incorporated into the statistical model, representing how an object deforms on the image-plane rather than within its own 3D co-ordinate system. However, this is not always the case and deformable models have also been applied to 3D in computer vision in order to reduce some of the non-linearity introduced during the projection process. [Heap 96; Ferryman 95; Hogg 83] have tackled computer vision from this 3D perspective, which is basically the reverse mapping of the rendering procedure. In computer graphics, [Pentland 96; Parker 97] have used statistics and interpolated models to produce 'life-like' renderings and animations of human facial motion.

The use of computer vision techniques in motion capture is common placein acquiring trajectories for key points of objects that are used to produce life-like 3D animations. Figure 7.2.1 and Figure 7.2.2 show motion trajectory files for a running and walking human female[9]. These were captured using reflective IR markers on a real world human subject. The trajectories of these markers in space were recorded in multiple camera views and the trajectories of these points calculated using standard stereo reconstruction techniques. The model consists of 32 3D-marker points and their trajectories through space. By connecting these points with a simple stick model the human motion can be visualized. In computer animation, these key points would be used to animate the articulated sections of a 3D virtual character for computer games or virtual environments.

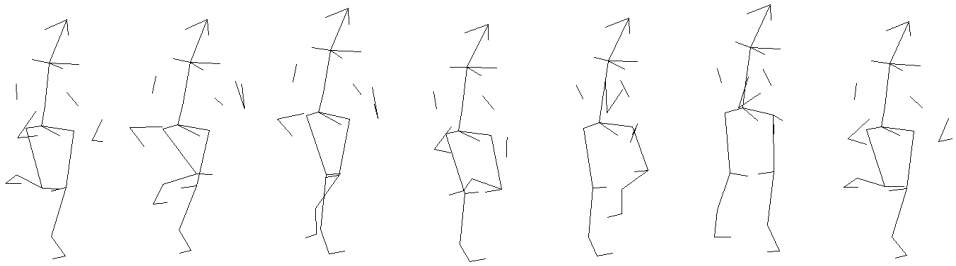


Figure 7.2.1 - Examples from a Key-frame animation of a Running Woman

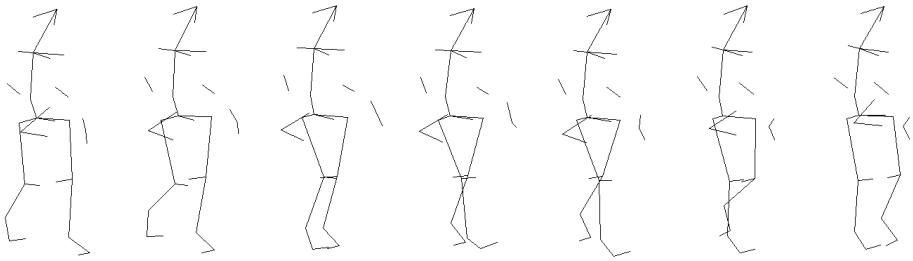


Figure 7.2.2 - Examples from a Key-frame animation of a Walking Woman

It is this notion of *key points* in the motion capture process that provides the link between statistical models and animation, where animation key points are akin to the landmark points used in statistical models. If statistical models of shape and deformation can be ***learnt*** from a training set, producing realistic constraints on the shape (or motion of landmark points), then similar ***learnt*** models of animation trajectories can also be achieved.

[9] The motion capture data for the female subject was provided by TeleVirtual Ltd.

7.2.2 The Linear Motion Model

The human motion capture data for both the running and walking woman consists of 32 key points for each frame of the animation; these points can be concatenated into a single 96 dimensional vector $V=(x_1,y_1,z_1, ..., x_{32}, y_{32}, z_{32})$. The running animation consists of 474 key frames recorded at 30Hz which produces a training set of 474, 96 dimensional vectors. The walking animation consists of 270 key frames, again captured at 30Hz using 32 key points producing a training set of 270, 96 dimensional vectors. Now the training sets are in a form that enables further statistical analysis: linear PCA can be performed upon them to produce a linear 3D PDM.

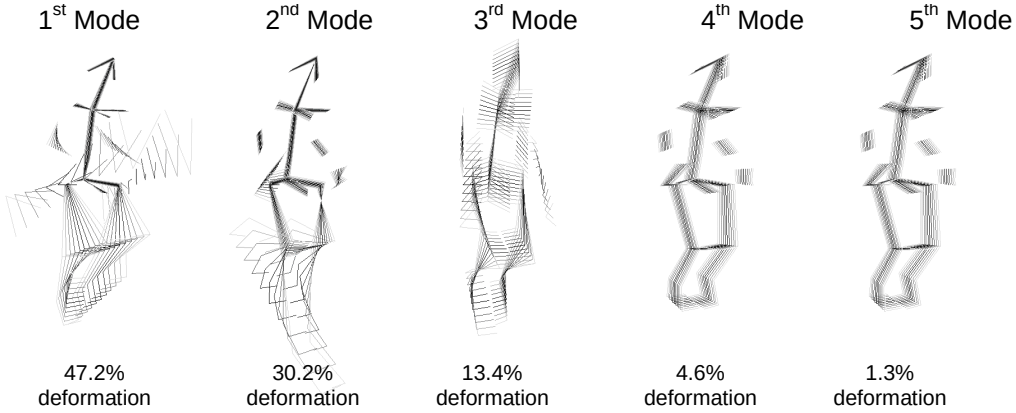


Figure 7.2.3- The Running Linear 3D PDM

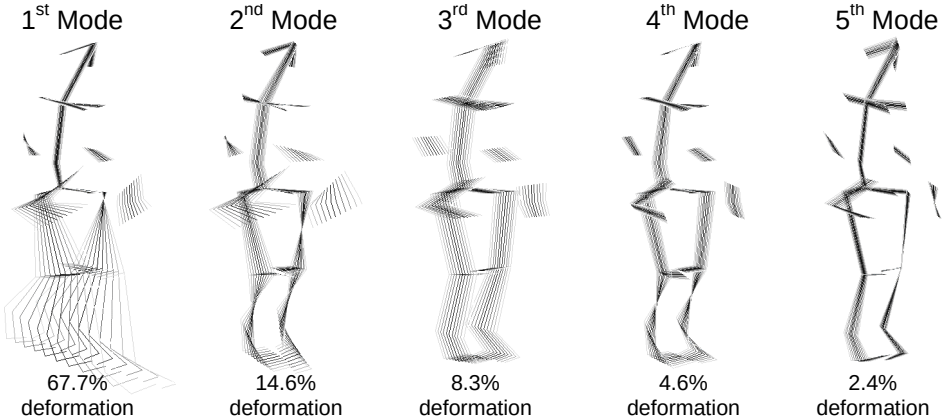


Figure 7.2.4 - The walking Linear 3D PDM

From the eigenvalue analysis, 98.8% of the deformation of the running model is contained within the first 10 eigenvectors, with 99.4% of the walking model being encompassed by the 10 eigenvectors.

It can be seen from Figure 7.2.3 and Figure 7.2.4 that the linear 3D PDM does not model the trajectories of key points (and associated body parts) well. The motion files contain perfect landmark point identification between examples. However, the data sets are still non-linear due to the circular motion of the body parts. This non-linearity can be seen in Figure 7.2.6 and will be discussed shortly. It should be noted that the 3rd mode of variation of the walking model encompasses mainly translation. This is due to the change in speed as the walker establishes a consistent gait, and remains a part of the model due to the absence of the alignment of the training examples. Had the normal alignment procedure been followed, then this translational information would have been reduced. The translation correlates to the shift in m_1 of the walking model seen in Figure 7.2.6b. However, this information is important to the realism of the animation and must therefore remain a component of the model. It will later be removed through the use of temporal dynamics.

7.2.3 Adding Non-linear Constraints

Using the methods previously discussed, the data sets are first dimensionally reduced by projecting each of the training examples down onto the eigenvectors of the linear PDM. Using the 10 primary modes of the linear model as determined in the previous section, both the running woman data and the walking model are projected down from 96 to 10 dimensions. These lower dimensional data sets are shown in Figure 7.2.6 as points drawn in 3D from two 2D views.

Cluster analysis was then performed on the reduced data sets. The resulting cost files are shown in Figure 7.2.5. The natural number of clusters for the run and walk trajectory files can be estimated to be 25 and 30 respectively. The larger number for the walking model is due to the model translation introduced as the subject establishes a consistent gait, as mentioned earlier.

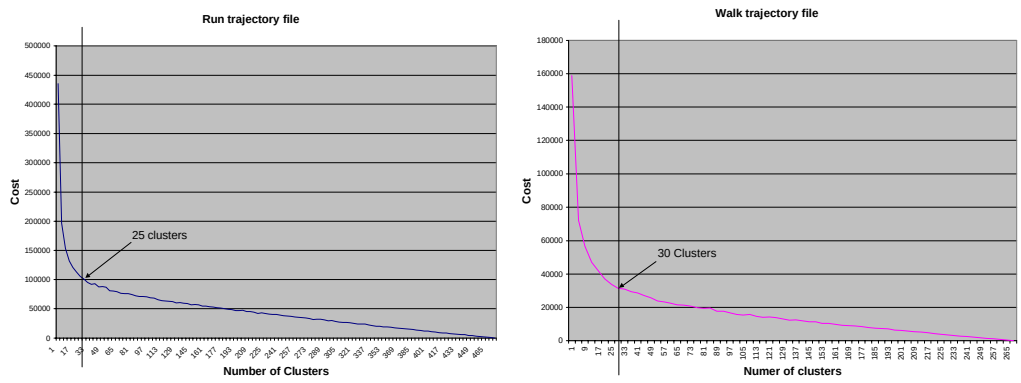
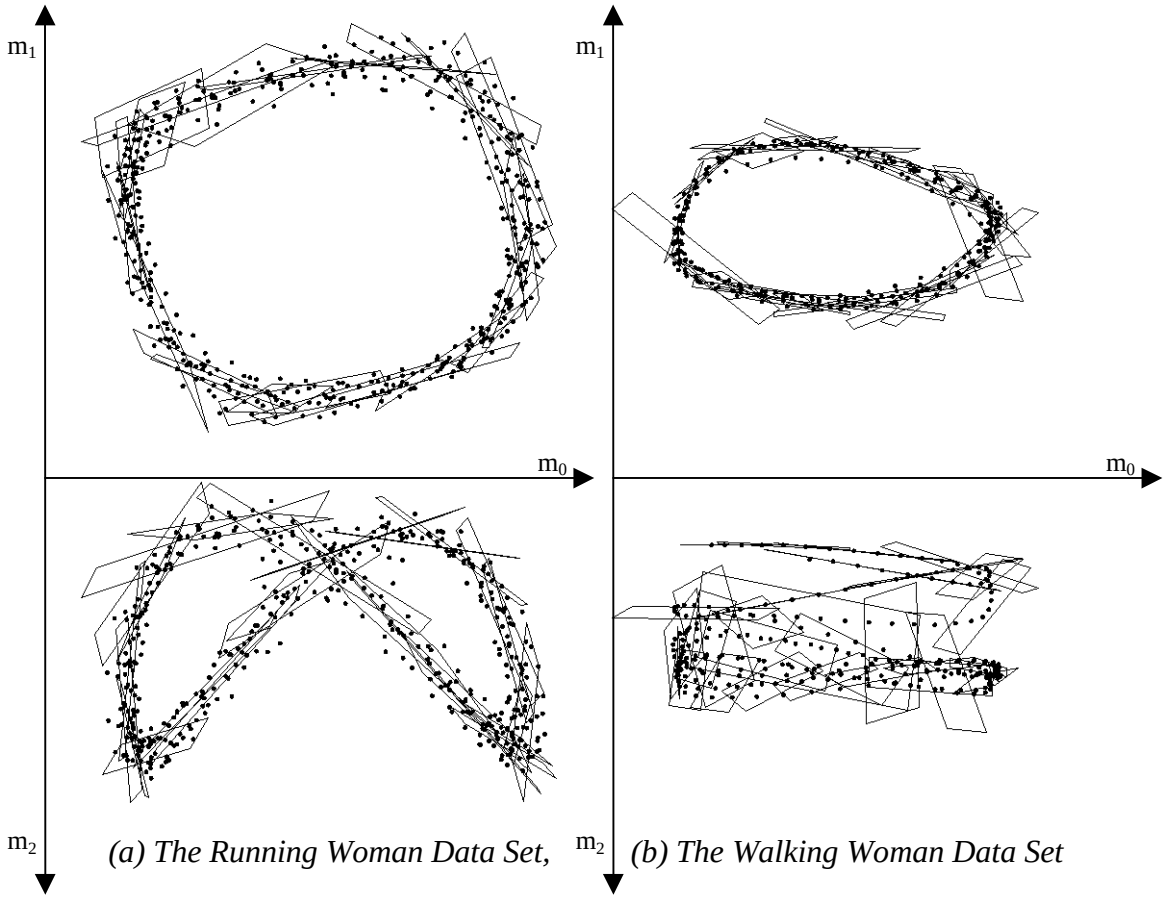


Figure 7.2.5 - Cost files for Trajectory Data

Using the natural number of clusters for each data set, the fuzzy k-means algorithm was used to segregate each data set into its composite clusters. Each cluster was then modelled by performing further PCA upon its members. The final non-linear constraints can be seen in Figure 7.2.6 with the bounds of each cluster drawn as a rectangle over the reduced data set.



(a) The Running Woman Data Set, *(b) The Walking Woman Data Set*

Figure 7.2.6 - Dimensionally Reduced Data sets with the Cluster Based Constraints

From this diagram it can be seen that the clustering algorithm has smoothly estimated the natural curvature of the data set through piecewise linear patches. Each cluster better estimates the model locally as each linear patch must encode less information.

The CSSPCA has ***learnt*** the *Motion Capture Space* and can be used to reproduce viable shapes from the model. However, in computer animation this is insufficient. For animation purposes, the ability to model the trajectory through shape space is also required, allowing the motion to be reproduced.

7.2.4 Learning Temporal Constraints

Thus far the techniques have been used to learn the shape and size of the trajectory space, temporal analysis must be performed to estimate how the model moves through space with respect to time.

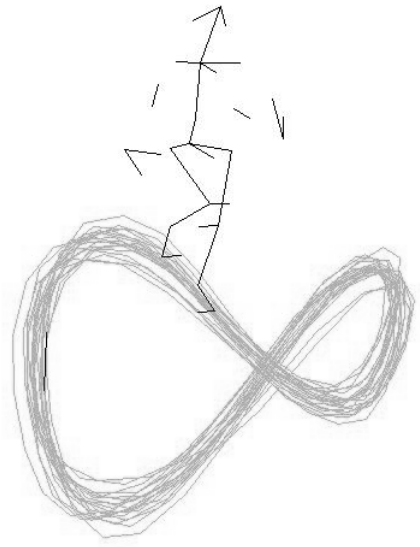


Figure 7.2.7 - Trajectory through Reduced Shape Space

Figure 7.2.7 shows the 3D trajectory of the reduced dimensional running data set projected down into 3 dimensions. Using simple animation techniques it is possible to watch the model move throughout the space as the animation sequence iterates. It is apparent that the motion is cyclic and consistent in nature and repeats in accordance with the period of the stride of the actor. Therefore,

given any point within the space it is possible to predict where the model will move to next, based upon this observed motion.

The model has been estimated in a lower dimensional space; if the trajectory can also be modelled in this lower space then it is likely that paths of motion throughout the space could be determined and reconstructed. The key again is in this probabilistic analysis of the training set. The deformation constraints have already broken the shape space down into linear patches with the centre of the clusters being the mean shape of the transition at that point in time. It is also known that, due to the cyclic nature of the data set, the pattern of movement repeats at regular intervals for fixed speeds of motion. Although this is not a necessary condition, it can effectively be modelled as a self-starting, finite state machine. This lends itself naturally to a discrete, time dependent, probabilistic analysis of the motion.

The reduced training set can therefore be used to analyse the model and probabilistically learn the transition of the model between clusters. This can be done with a state transition matrix of conditional probabilities, otherwise known as a Markov chain.

7.2.5 Modelling Temporal Constraints as a Markov Chain

A Markovian assumption presumes that the present state of a system (S_t) can always be predicted given the previous n states (S_{t-1}, S_{t-2}, ..., S_{t-n}). A Markov process is a process which moves from state to state dependent only on the previous n states. The process is called an order n model where n is the number of states affecting the choice of the next state. The simplest Markov process is a first order process, where the choice of state is made purely upon the basis of the previous state. This likelihood of one state following another can be expressed as a conditional probability $P(S_t|S_{t-1})$.

A Markov analysis looks at a sequence of events, and analyses the tendency of one event to follow another. Using this analysis, a new sequence of random but related events can be produced which have properties similar to the original.

The probability mass function $P\left(C_j^{t_n}\right)$ denotes the unconditional probability of being in cluster *j* at time t_n, or being in state *j* after *n* transitions (time steps). A special situation exists for *n=0* where $P\left(C_j^0\right)$ denotes the probability of starting in state *j*. However, due to the assumption that the motion is cyclic and the trajectory file starts and ends mid-cycle, no information is available for these initial probabilities.

The conditional probability mass function is therefore defined as

$$P\left(C_j^{t_n}\middle|C_k^{t_m}\right)$$

$P\left(C_j^{t_n}\middle|C_k^{t_m}\right)$ gives the probability of being in cluster *j* at time t_n conditional on being in cluster *k* at time t_m. In the trajectory file example it is fair to make the assumption that the next state of the model can be determined from the previous state. This can be confirmed by observing the trajectory taken through shape space by the training set (see Figure 7.2.7). Provided stationary elements of the chain are ignored, i.e. where $P\left(C_j^t\middle|C_j^{t-1}\right)\geq \max_k\left(P\left(C_j^t\middle|C_k^{t-1}\right)\right)$ and therefore choosing the 2nd highest probability move at each time step, the continuous transition through shape space can be achieved. If this assumption is made, then the process becomes a first order *Markov process* or *Markov Chain* and $p_{j,k}$ a one step transition probability

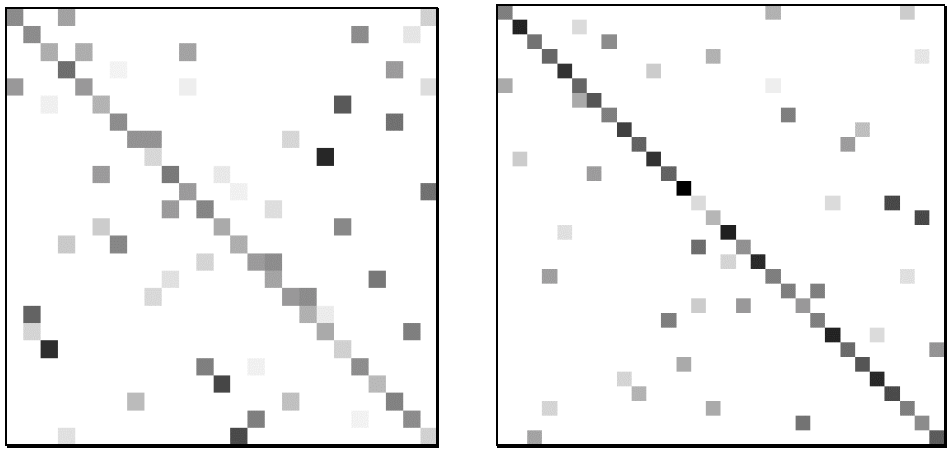
$$p_{j,k} = P\left(C_j^t\middle|C_k^{t-1}\right)$$

If there are *n* clusters in the model, then there are *n* states in the chain, hence a state transition matrix is an $n\times n$ matrix of one step transition probabilities. This is constructed in a similar manner to the classification probability matrix constructed in section 6.5.6, and is a discrete probability density function (PDF).

$$P\left(C_j^t \middle| C_k^{t-1}\right) = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{j,1} & \cdots & \cdots & p_{j,k} \end{pmatrix},$$

where $p_{j,k} \geq 0$ for all *j,k*, and $\sum_k p_{j,k} = 1$ for all *j*.

After construction of the PDF its content can be visualised by converting the matrix to a grey-scale image. Figure 7.2.8 shows the resulting images for both the running and walking data sets. It can clearly be seen that high probabilities exist along the diagonal of the image. This diagonal, when $i=j$ or $S_t=S_{t-1}$, demonstrates that the model always has a high probability that it will stay within the same local patch. This can be attributed to the discrete nature of the model, and the fact that each patch is constructed to model local deformation. The darker diagonal in the walking model shows that this model has a higher probability of remaining within a local patch and is a result of the speed of movement. As both sequences were captured at the same rate, the slower movement of the walking model generates more frames in each local patch and hence a lower probability that the model will make a transition to another patch. However, as the numerical identity of each local patch within the matrix is randomly generated by the k-means algorithm, no further conclusions can be drawn from the patterns within the image, hence the random distribution.



(a) The Running Woman Data Set *(b) The Walking Woman Data Set*

Figure 7.2.8 - Discrete Probability Density Functions

The *PDF's* shown in Figure 7.2.8 provide a conditional probability that, given a cluster at time *t*, the system will move to another cluster at the next time step. By taking the highest probability move at each time step the highest probability path can be modelled throughout the space.

Using this information and the mean shape of each cluster as key frames, the motion of the training set can be reconstructed. If any cluster of the model is chosen at random and the next highest probabilistic transition made at each time step $\mathbf{argmax}_i\left(p_{i,j}\right)$ where $i \neq j$, the model should settle within a natural path through the space. This is similar to a finite state machine that has a circular path and is self starting. If the natural number of clusters selected is correct then the cyclic period of the model should be equal to that of the training set. If the cluster number is too high then non-equidistant cluster centres result and the model appears to 'jerk'. If the cluster number is greater than twice the natural number then the model risks having a cyclic period of multiples of that of the true motion.

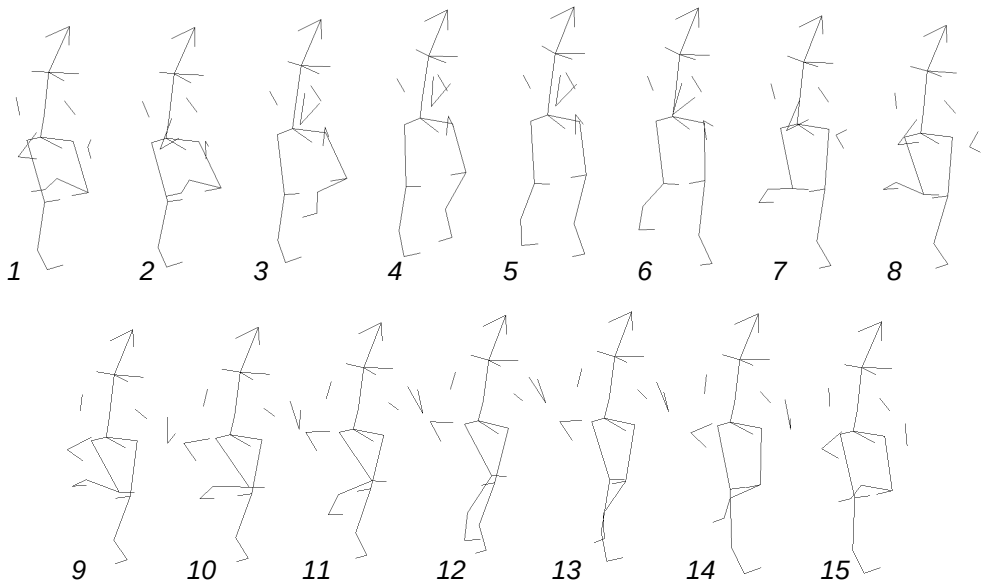


***Figure 7.2.9* - Extracted Trajectory for Running Model**

Figure 7.2.9 shows the highest probability path for the running model that consists of 15 clusters. Each pose of the model is the mean shape (exemplar) of a cluster. This model is reconstructed from the information that has been learnt

from the motion file and accurately reproduces the original motion. The animation can be further refined by linearly interpolating between these key frames (exemplars), as the linear interpolant along a line between exemplars is equivalent to linearly interpolating all points on the model between key frames. This does however introduce slight non-linear deformities. These deformities can be reduced by projecting the interpolated model into the constrained space to extract the closest allowable model for rendering.

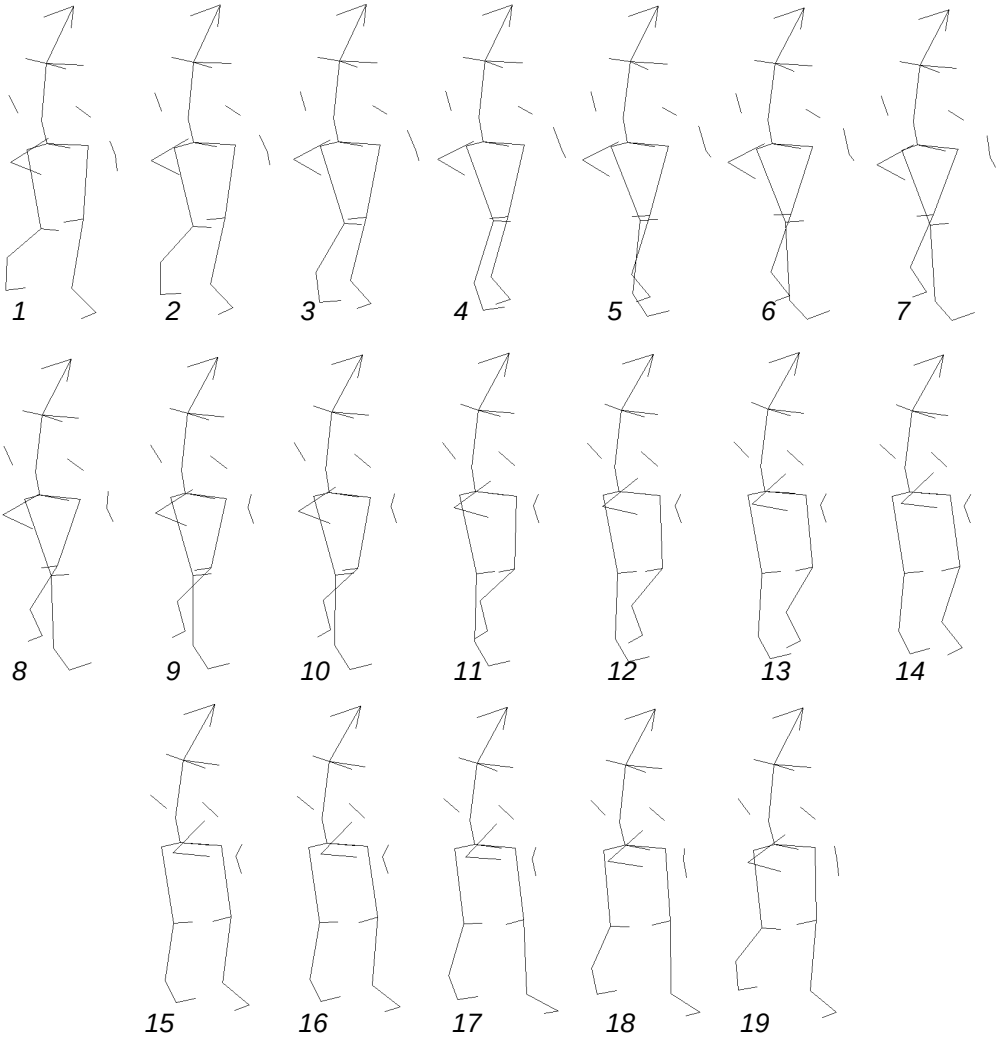


Figure 7.2.10 - Extracted Trajectory for Walking Model

Figure 7.2.10 shows the highest probability path through the walking model, consisting of 19 key frames that produce a cyclic path of high probability through the Markov chain. The original model contained 30 clusters and the redundant 11 clusters partly model the introductory gait acceleration, which can

be seen in Figure 7.2.11. The red line shows this high probability path extracted from the Markov chain. Acting like a self-starting finite state machine, if the model is initiated within the low probability startup area of the space, the chain quickly moves the model to the circular region, where constant cyclic movement occurs.

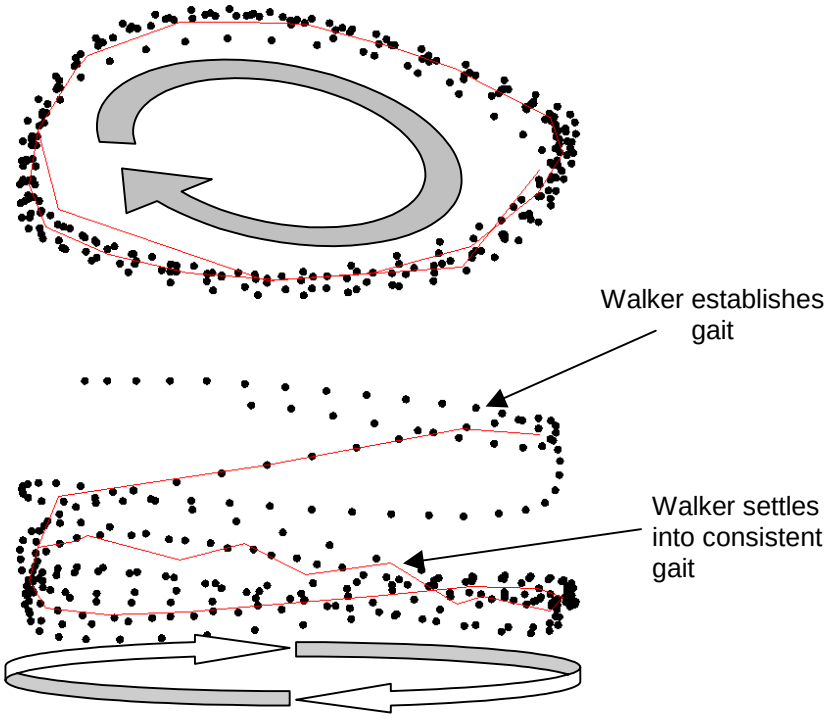


Figure 7.2.11 - High Probability Path through Walking Model Shape Space

7.2.6 Conclusions

In this section it has been shown how the reduced dimensionality and discrete representation of the Constrained Shape Space approach to modeling non-linear data sets can be used to provide simple analysis and reconstruction of motion. This is done by analysing the training set and constructing a Markov Chain, which is a discrete, probabilistic representation of the movement of the model through shape space. It has also been shown how, using this learnt temporal information, animated models can be produced which encapsulate the temporal information learnt from a training set.

7.3 Tracking with Temporal Dynamics

7.3.1 Introduction

In the previous section, temporal information was learnt from a training set in addition to deformation. It has been shown how this temporal deformation can be used to represent and reproduce motion. However, for many computer vision techniques this is not the ultimate goal. What is beneficial is using this *learnt* temporal information to further constrain the model, or predict the movement and deformation of an object, thus producing more robust tracking and classification.

A large body of work has been performed on the temporal mechanics of tracking. Many researchers have attempted to use predictive methods such as those based within a Kalman filter framework [Blake 98]. Hill *et al* proposed using genetic algorithms to model the discontinuous changes in shape space/model parameters [Hill 91][Hill 92].

Of particular interest to the work presented in this thesis is the CONDENSATION algorithm [Isard 98] [Blake 98] which is a method for stochastic tracking where a population of model hypotheses are generated at each iteration. These populations are generated from pre-learnt PDFs generated over the model parameter space to provide a hypothosis-and-test approach to model prediction and tracking. A more comprehensive introduction to Condensation is given in Section 2.5.

Condensation is a powerful tool in deformable model tracking for several reasons:

1. It supports multiple hypotheses and therefore produces robust results for tracking with occlusion and discontinuous movement.
2. It uses *a priori* knowledge about the object to predict its movement.
3. It recovers well from failure, allowing the model to 'jump' out of local maxima/minima.

It has been shown that, due to the discrete nature of the piecewise linear approach to modeling non-linearity, the approach directly lends itself to a discrete PDF with the addition of the Markovian assumption.

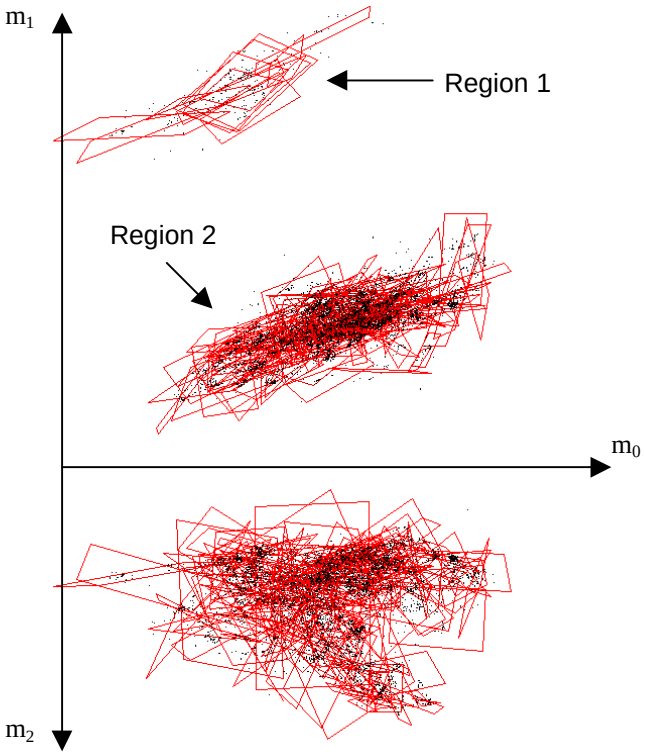


Figure 7.3.1 - Constrains on PCA space for the ASL Model

This temporal information can be used to augment the CSSPDM model with conditional probabilities, which allow the support of multiple hypotheses similar to that used in Condensation. This is important due to the discrete nature of the piecewise linear model. If the discontinuous shape space constructed for the American Sign Language (ASL) alphabet is considered from Section 6.5.6 (see Figure 7.3.1), it can be seen that shape space is segregated into **at least** two separate regions due to the movement of landmark points around the boundary (see section 2.4 for a description of these types of non-linearity). Furthermore, connected patches of the model may not represent consistent movement of the model in the image frame. This leads to the model *jumping* between patches, even when within region 2. Under these circumstances it is not possible for the

iterative refinement algorithm used for the classic PDM/ASM (section 3.3) to provide the *'jump'* between regions.

An image sequence was recorded of a hand signing the word 'gesture' which consisted of 170 frames. Figure 7.3.2 shows the model attempting to track the image sequence for the letters 'e' and 'u'. The model successfully tracks the letter 'e' but when the image sequence reaches the letter 'u' and the fingers elongate, the model is unable to make the jump to the new cluster responsible for modeling this letter. This problem is fundamental to the operation of the least squares iterative refinement algorithm and is due to two reasons:

1. Only a small section of the contour (marked in frame 'u') is responsible for 'pulling' the contour up to follow the elongated fingers. As this section is relatively small, compared to the remainder of the contour, it has less influence over the overall movement.

2. The maximum movement of the contour per iteration is governed by the length of the normal used to search around the contour. Hence this factor limits the distance the model can move through shape space at each iteration.

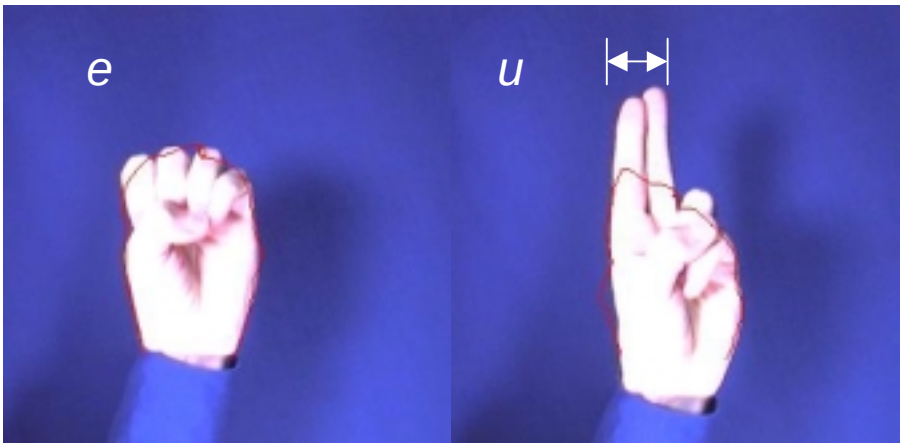


Figure 7.3.2 - ASL model Tracking an Image Sequence of the word 'gesture'

An obvious solution to these problems is to increase the search length along normals. Figure 7.3.3 shows the results of various parameters for the least squares iterative refinement algorithm on the ASL model. The graph

demonstrates the effect of varying the number of iterations per frame and the length of the normal (in pixels) either side of the contour. The cost at each iteration is the sum of the pixel difference between the desired movement of the model (gained from the assessment of the normals) and the final shape (after the constraints of the model have been applied). Where multiple iterations per frame were performed, these are displayed as fractions of a frame to visualise the resulting error cost of iteration. The corresponding letters of the sequence are shown with the vertical lines denoting the approximate transition between letters. At these transitional frames, the model error rises due to the increased speed of movement of the hand. During these faster movements the iterative refinement procedure must make larger movements through shape space to deform with the image. This produces the increase in error due to the limiting factor of the localised normal search.

Increasing the number of iterations produces a resulting reduction in cost up to a certain threshold, at which point the cost begins to rise again. This can be attributed to the finer iterations allowing the model to achieve poses from which it can not easily extract itself and is a further drawback of using the least squares iterative refinement approach to fitting a non-linear model. Although the increased normal length allows the model to achieve the aforementioned transition to the letter 'u', the resulting cost demonstrates a reduction in the overall performance of the model. The larger normal search allows the contour to affix to incorrect features in the image and hence results in degradation. Where image sequences with heavy background clutter are considered, this problem becomes more acute.

Another drawback of large normal searches is the resulting computational cost in assessing the additional pixel intensity gradients. It is therefore necessary to use a tracking paradigm that allows these quantum leaps in shape space to be made while retaining the localised searching and constraints of the model.

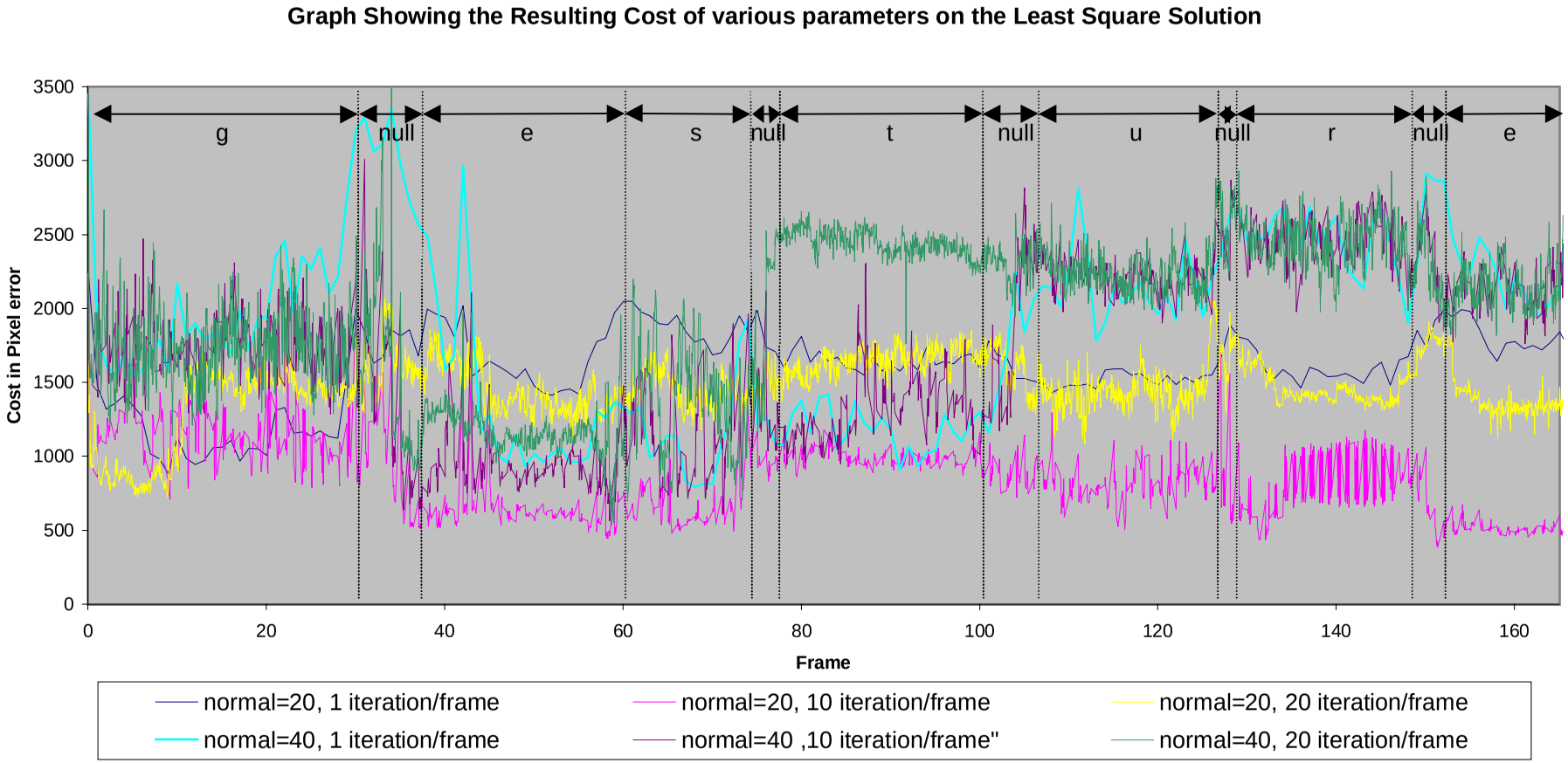


Figure 7.3.3 - Graph of error cost for Least Squares Fitting with Various Parameters

7.3.2 Finding the Optimal Ground Truth for Tracking

To locate the optimum solution (i.e. the closest allowable shape from the Constrained Shape Space PDM, CSSPDM) for each iteration of the model, the space was exhaustively searched. If the assumption is made that any local patch of the CSSPDM can indeed be treated as a linear model, then the iterative refinement procedure can be used to move locally within that patch to the closest possible shape. Therefore, if the best match within each patch (cluster) is located for each frame, the resulting lowest cost solution must be the (near) optimum.

This exhaustive search was performed on the 'gesture' image sequence. For every frame, each of the 150 clusters were assessed in turn. The mean shape of the cluster was used as a starting shape and the iterative refinement of the model, within the cluster, performed until the model converged (typically 40 iterations). The cluster that produced the lowest cost solution was deemed to be the optimum and the resulting costs plotted in Figure 7.3.4 along with the lowest of the least squares approaches from Figure 7.3.3.

The two smoothed plots are polynomial trendlines fitted to the data to help visualise the overall efficiency of the approaches. The optimum solution produces a lower error than that of iterative refinement, which would be expected. However, both exhibit similar trends. From this it can be inferred that some of the errors produced during tracking are not the result of the algorithm's inability to track successfully but are due to the constraints of the model. The higher error rates that result from letters such as 'g' and 'r' suggest that more training examples for these letters are required so as to increase the ability to model unseen shapes.

By analysing the optimum path through shape space and comparing this with the path taken by the least squares approach, the notion of discontinuity within shape shape can be confirmed.

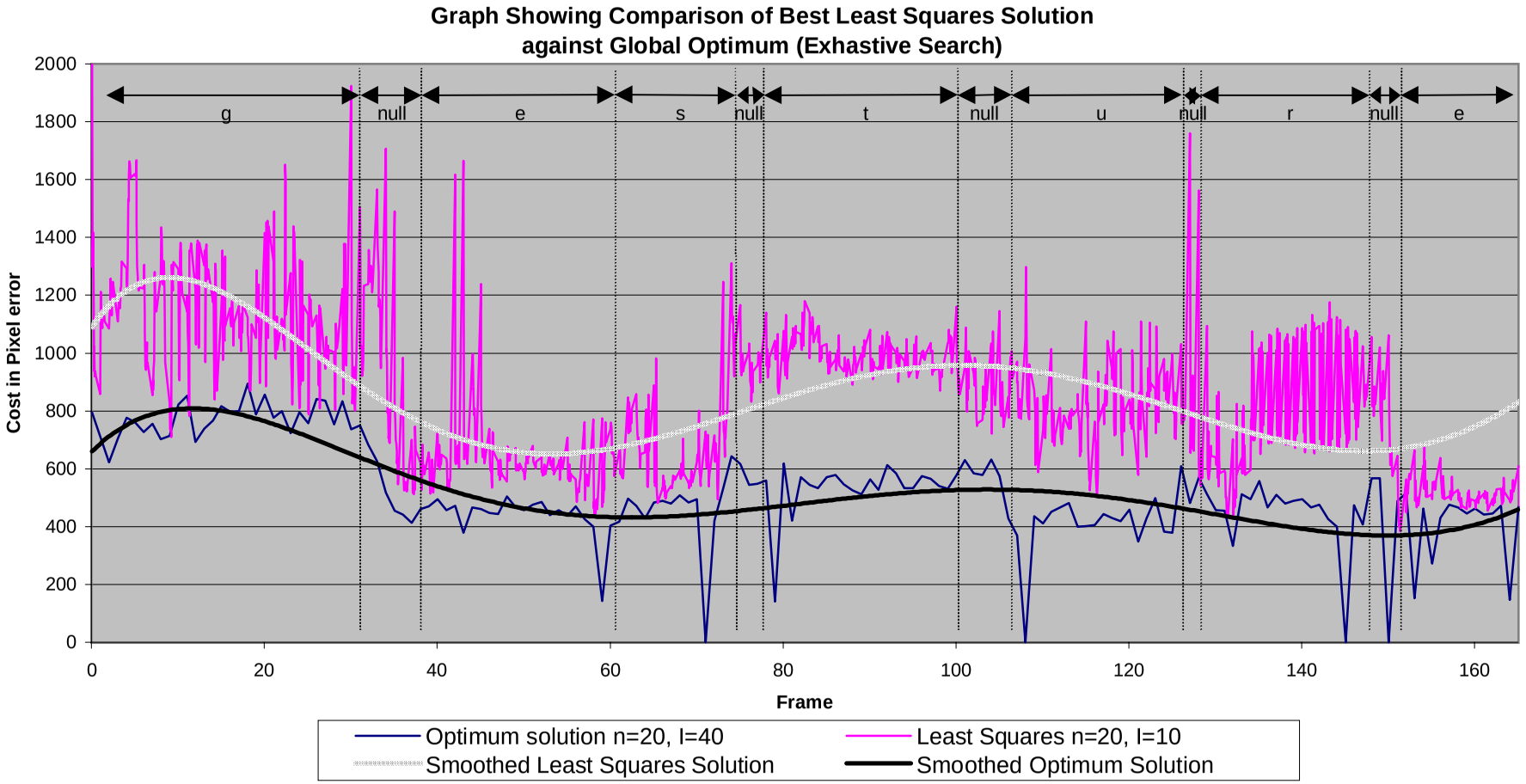


Figure 7.3.4 - Comparison of Least Squares Solution against Optimum Solution

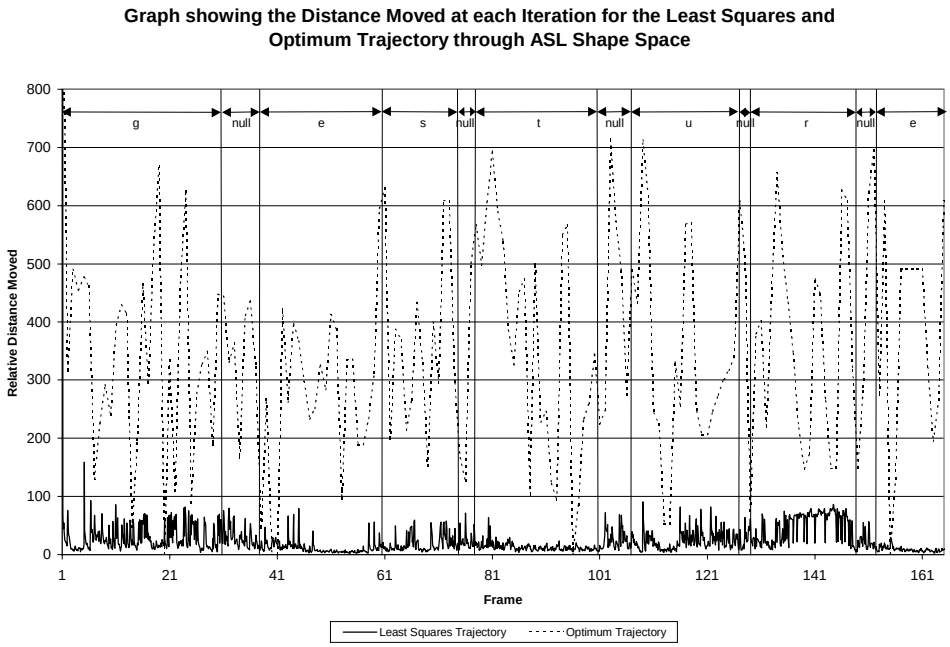


Figure 7.3.5 - Graph of Distance Moved at each iteration for Least Squares Solution and Optimum Solution

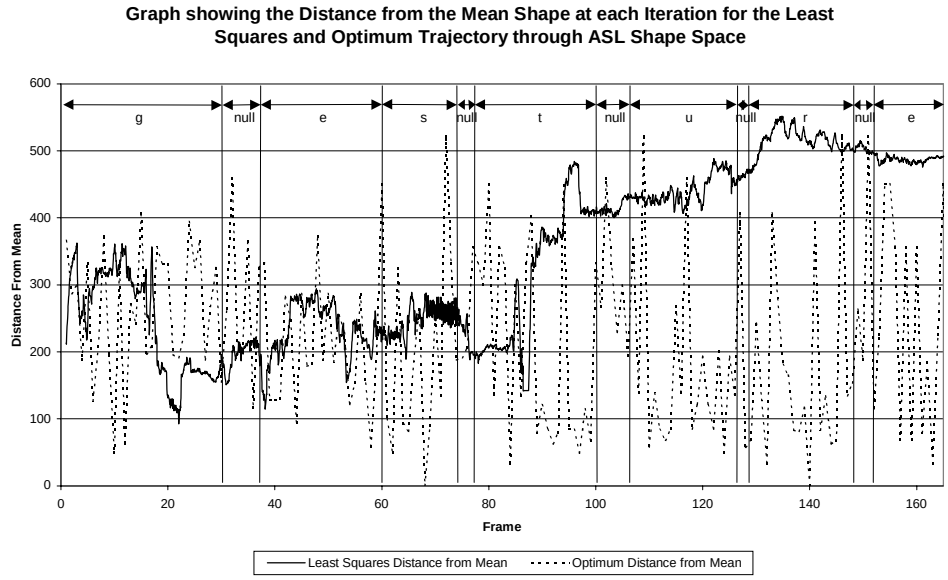


Figure 7.3.6 - Graph of Distance from Mean of Shape Space at each frame for Least Squares Solution and Optimum Solution

Figure 7.3.5 shows the distance moved through shape space at each iteration for both the optimum trajectory and the iterative refinement algorithm. From this it can clearly be seen that the least squares iterative refinement algorithm makes small incremental movements at each iteration, whereas the optimum trajectory

makes large 'jumps' at every frame. During the letters 'e' and 't' the least squares approach almost stops moving, which demonstrates that the model has converged upon a stable solution. However, the lack of such trends for other letters shows that the model is constantly struggling to better refine itself. Figure 7.3.6 shows distance from the centre of shape space for the two trajectories at each iteration. Again this demonstrates that the optimum path jumps violently within the space whereas the least squares approach makes small movements. The high values achieved by the least squares approach for the letters 'u' to 'e' show that the model is at the extremity of shape space making small movements. However, the relative movement of the model in Figure 7.3.5 for frames 100-150 show that it is moving considerably at each iteration attempting to find a better solution.

The most interesting aspect of these figures is within Figure 7.3.6. The letter 'e' occurs twice during the sequence. However, during the first occurrence the least squares approach is at a distance of around 200 units from the mean whereas during the second occurrence it is at around 500. This demonstrates two facts:

1. That there are at least two areas of shape space responsible for modeling the letter 'e' and these are distinctly separated in shape space.
2. The least squares approach can only use the local 'e' part of shape space and is incapable of jumping between them.

This confirms that not only is the non-linear shape space discontinuous but the least squares iterative refinement approach is incapable of providing a robust method for tracking. Instead a new method of applying CSSPDMs must be devised.

7.3.3 Supporting Multiple Hypotheses

By taking advantage of the Markovian assumption, a similar model of temporal dynamics can be generated for the ASL model as was constructed for the motion capture data previously discussed, where the conditional probability $P\left(C_i^{t+1} \middle| C_j^t\right)$ is calculated. As has been discussed, the major discontinuities of the shape space occur when landmark points jump around the boundary and hence result in a

jump in shape space (Figure 7.3.5 and Figure 7.3.6). However, within each patch, the model still makes small iterative movements. This can be confirmed by visualising the resulting PDF as a grey scale image.

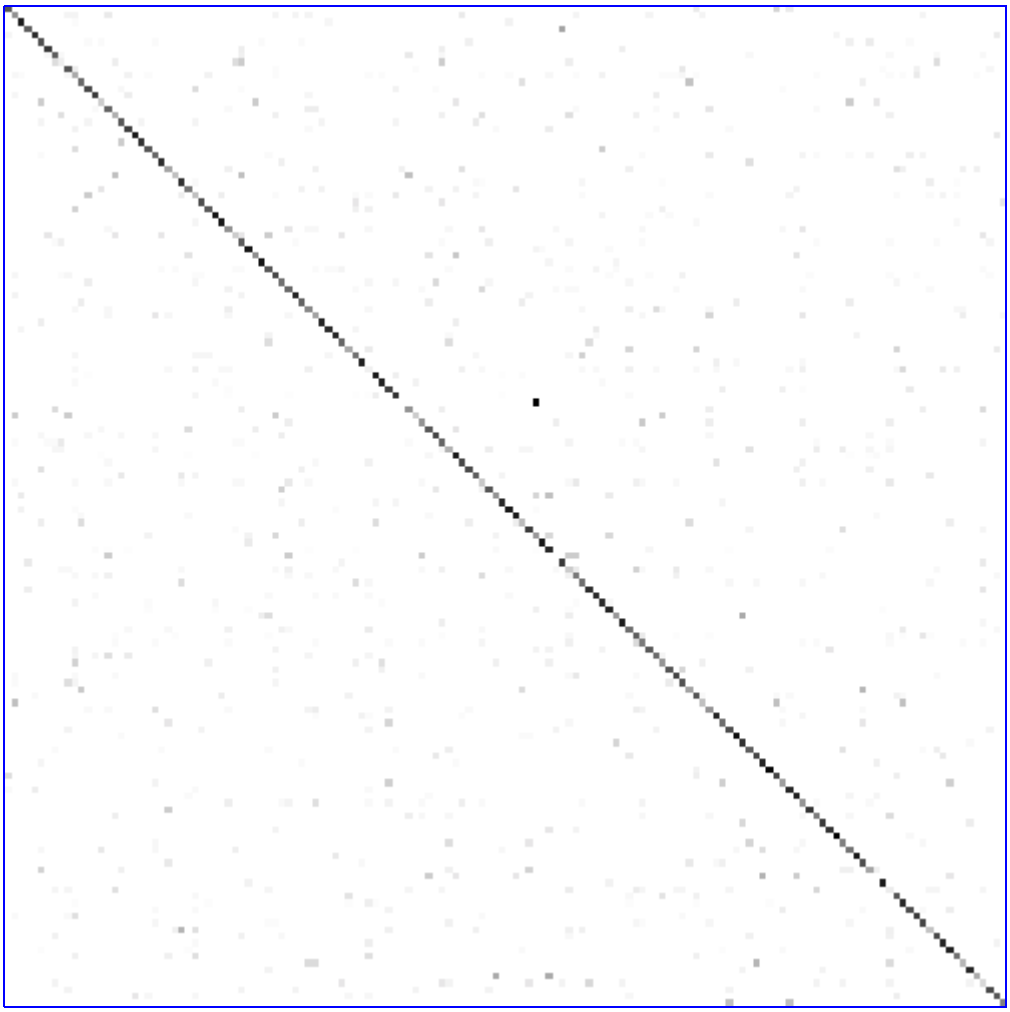


Figure 7.3.7 - Discrete Probability Density Function for ASL Model

Figure 7.3.7 shows the ASL PDF, which again has a heavy diagonal dominance. This dominance is when $\mathbf{argmax}_i\left(P\left(C_i^{t+1}\middle|C_j^t\right)\right)$ and $i=j$ i.e. the highest probability is that the PDM will usually stay within the present cluster. The assumption can therefore be made that within any local patch the model can iterate to a local solution. This confirms the assumption used when calculating the optimum model shape. This assumption also provides two benefits:

1. The iteration to convergence of any global optimisation technique can be enhanced by allowing each hypothesis to iterate to a better solution within the present cluster.
2. A smaller population is required, as only global differences in hypotheses need to be supported.

This is a common procedure in speeding up the convergence on solutions for many optimisation techniques such as in neural networks or clustering [Boyle 95]. By combining a gradient descent method with a global optimisation approach the speed to convergence is increased and the problem of oscillating down narrow energy wells to local minima reduced.

From the 'learnt' probability density function, a sample population can be generated at each iteration of the model. Given a good initialisation of the model (see section 3.3.2) and the associated cluster $C^{t=0}$, which encompasses that shape, the procedure is summarised thus:

Algorithm 7-1 - Simple CSSPDM Condensation

- From the PDF $P\left(C_i^t \middle| C_j^{t-1}\right)$, extract the probability vector $P\left(C_i^{t=1}\right)$, which is the probability distribution of the first iteration, given $C_j^{t-1} = C^{t=0}$.
- Generate a randomly sampled distribution of *k* hypothoses $\mathbf{x}_\rho [\rho = 1,\ldots,k]$, where $\mathbf{x}_\rho$ is the mean shape of cluster C_i and $P(C_i) = P\left(C_i^{t=1}\right)$
- While still tracking,
  - Fit the *k* hypothoses to the image frame using the least squares gradient descent algorithm (section 3.3) and iterate, applying CSSPDM constraints and assess fitness using error metric (section 7.3.2)
  - Sort hypothoses into descending order according to error
  - Take lowest error solution and locate closest cluster *c*
  - From the PDF $P\left(C_i^t \middle| C_j^{t-1}\right)$, extract the vector $P\left(C_i^t\right)$, which is the probability distribution of the next iteration, where $C_j^{t-1} = c$

- Generate a new randomly sampled distribution of k hypothoses $\mathbf{x}_\rho[\rho = 1,\ldots,k]$ where $\mathbf{x}_\rho$ is the mean shape of cluster C_i and $P(C_i) = P(C_i^t)$

By repeating this procedure for each frame, iteration allows the model to converge in the least square sense upon local solutions. However, due to the generation of a new population of hypotheses gained from the *a priori* information about movement contained within the PDF, the models are permitted to 'jump' within shape space at each new frame. This allows multiple hypotheses to be supported simultaneously, where the current lowest cost hypothosis is deemed to be the correct one. Figure 7.3.8 demonstrates the error rates produced by this simplified form of the condensation algorithm (Algorithm 7-1). Experiments were performed to assess the result of various parameterizations of the algorithm, where

n is the length of the normal search on either side of the contour
I is the number of least squares iterations used for each hypothosis
k is the size of the population size or the number of hypothosis used

Varying these parameters produces dramatic variations in the resulting error rates produced and the overall performance of tracking. Many of the higher error parameterizations fail to track the image sequence completely producing a zero success rate and hence consistently high error rates. With *n=40* (as with least squares iterative refinement) high failure rates are produced, as do small populations and low numbers of iterations. It is important to note that a population size of one (*k=1*) is effectively least squares iterative refinement due to the diagonal dominance of the PDF.

The best results were achieved using a normal length of 20 pixels, a population size of 10 multiple hypotheses and between 5 and 10 iterations per hypothesis (i.e. *n=20, k=10, I=5/10).* These traces are shown in Figure 7.3.9 along with the results of both the optimum trajectory and the iterative refinement approach for comparison. The trend lines give a good indication of the overall performance of the various approaches.

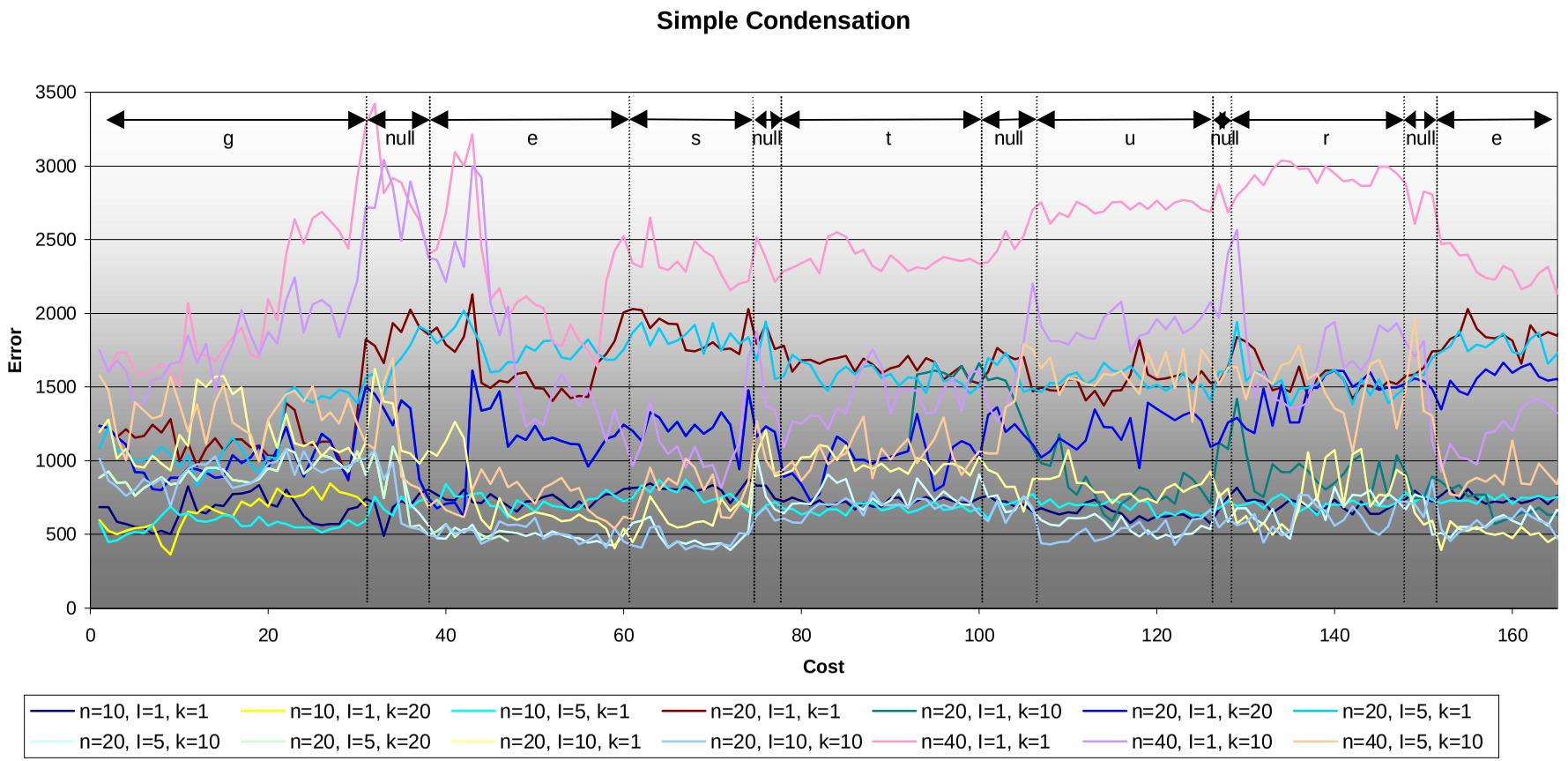


Figure 7.3.8 - Graph showing the Error rates Achieved by Varying the Parameters of the Simplified Condensation Algorithm

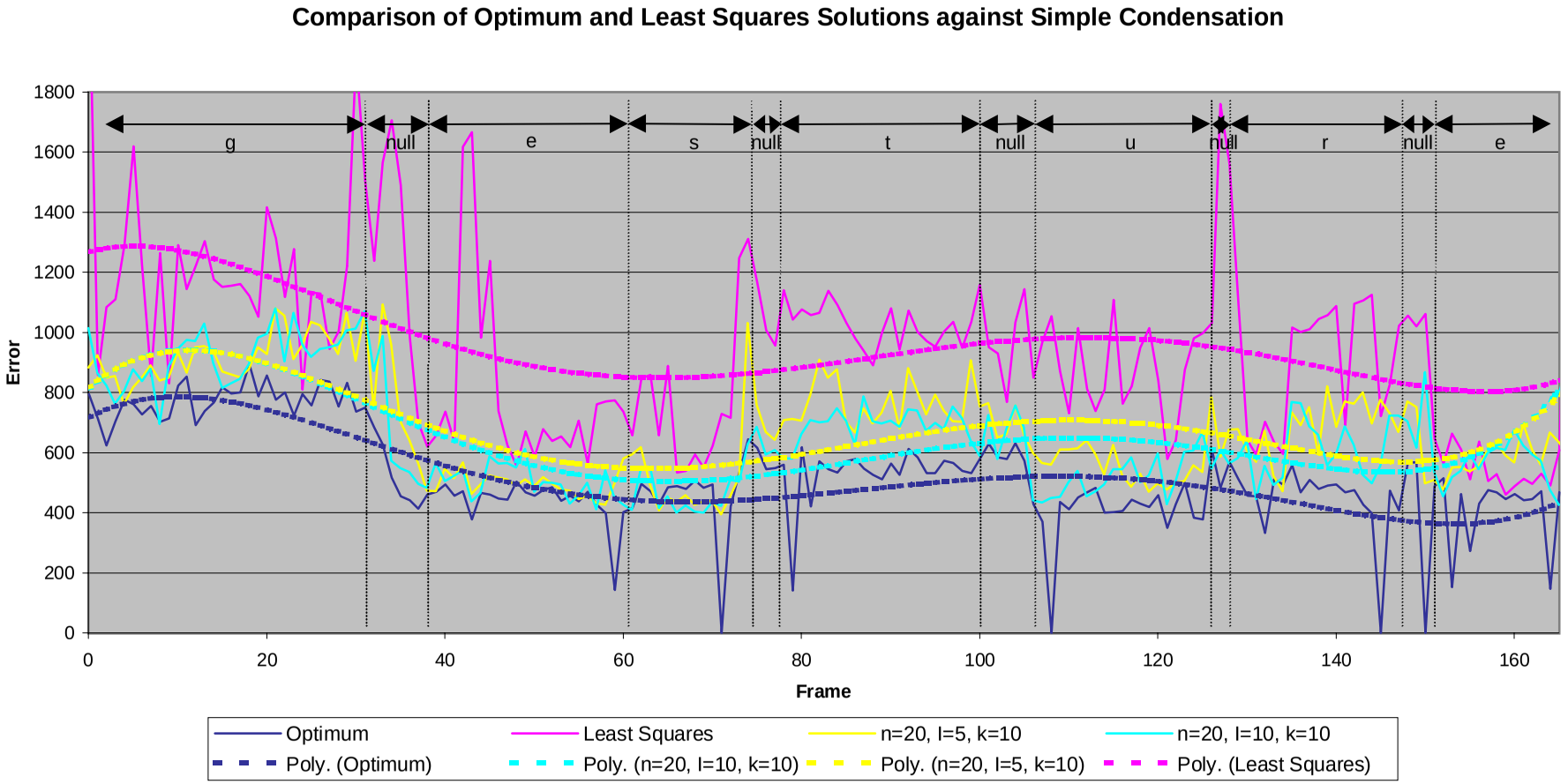


Figure 7.3.9 - Graph Comparing Simple Condensation against Previous Techniques

Figure 7.3.9 shows that both the simple condensation approaches produce significantly better results than the iterative refinement least squares tracking, but not as low as the optimum which would be expected. Increasing the number of iterations performed on each frame from 5 to 10 provides a slight increase in performance but not significant enough to warrant the additional computational overhead.

However, with such a low population size (p=10) and only five iterations required per frame (i=5) a total of (p*i), 50 models are fitted to the image at each frame. This provides a significant computational saving upon standard condensation where typically much larger populations (in the order of hundreds are required) to accurately track objects.

However, this approach, unlike condensation, does not recover well from failures. As the new population is solely based upon the current best-fit cluster the approach is highly sensitive to both an accurate PDF representation of the expected movement and the assumption that the best-fit cluster is actually affixed upon the object. To help overcome this drawback two factors must be addressed.

1. Less emphasis must be placed upon the current best-fit hypothesis being the optimum (and hence correct) solution, thus providing more robustness to failure.
2. The PDF must be an accurate and thorough representation of the expected object movement and hence the training set from which it is constructed must be general in both shape and movement. This is more difficult and will be addressed in the section 7.4.1.

Point 1 can be addressed by creating a new population of hypotheses, not from the current best fit model, but from the weighted sum of the best *n* hypotheses as described thus:

Algorithm 7-2 - Weighted Condensation

- From the PDF $P\left(C_i^t \middle| C_j^{t-1}\right)$, extract the probability vector $P\left(C_i^{t=1}\right)$, which is the probability distribution of the first iteration, given $C_j^{t-1} = C^{t=0}$.
- Generate a randomly sampled distribution of *k* hypothoses $\mathbf{x}_\rho [\rho = 1,\ldots,k]$, where $\mathbf{x}_\rho$ is the mean shape of cluster C_i and $P(C_i) = P\left(C_i^{t=1}\right)$
- While still tracking,
  - Fit *k* hypotheses, applying CSSPDM constraints and assess fitness using error metric
  - Sort hypotheses into descending order according to error
  - Iteratively refine first *n* hypotheses and resort
  - Apply the CSSPDM constraints and determine the *n* clusters C_η^{t-1}, where $\eta = 1,\ldots,n$ which produce the lowest error
  - From the PDF $P\left(C_i^t \middle| C_j^{t-1}\right)$, extract the vector $P\left(C_i^t\right)_\eta$ using the *n* extracted clusters. Take the weighted sum using a Gaussian weighting distribution to form a new distribution $P'\left(C_i^t\right)$, where

$$P'(C_i^t) = \sum_{\eta=1}^{n} \omega_\eta P(C_i^t)_\eta \text{ and } \omega_\eta = \exp\left[\frac{-9(1-\eta)^2}{2n^2}\right]$$

- Normalise probability distribution $P'\left(C_i^t\right)$.
- Generate a new random population of *k* hypotheses from the distribution $P'\left(C_i^t\right)$.

The results of applying this *weighted* approach to condensation are shown in Figure 7.3.10. This graph shows that, by using the best 5 models to generate the new population, lower error rates are achieved. Using the best 6 models produces less clear benefits but does provide increased ability to recover from failure.

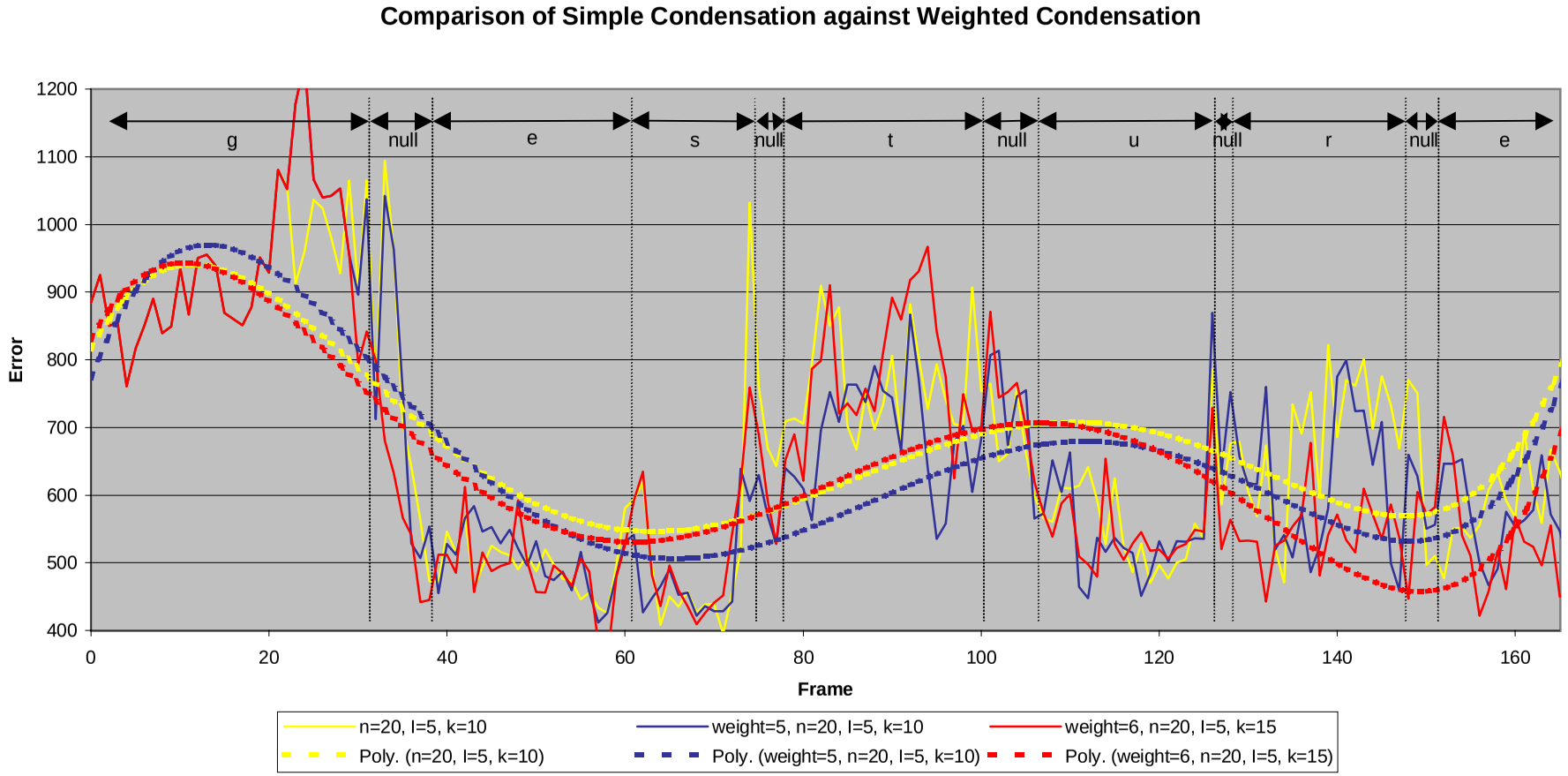


Figure 7.3.10 - Graph Comparing Simple Condensation against Weighted Condensation

7.3.4 Conclusion

This section has demonstrated that the nature of shape space need not be continuous. Under these circumstances it has been shown that the least squares, iterative refinement approach to PDM tracking fails. It has also been shown how the Markovian assumption can be applied to the CCSPDM to provide a fast tracking paradigm, which is less computationally expensive than standard condensation, while allowing multiple hypotheses to be supported.

7.4 Extending Temporal Dynamics to Classification

7.4.1 Introduction

It has been shown how, with the addition of a first order Markov chain to the CSSPDM, a hybrid approach to condensation can be used to provide robust tracking where either:

- The non-linearity of the PDM along with the discrete representation of the non-linear approximation leads to a discontinuous shape space.
- Rapid movement of the object produces large changes in the model parameters.

This Markovian model of dynamics can be used to explicitly constrain the movement of the model within shape space, or implicitly, using the hybrid condensation approach. However, the use of temporal constraints relies upon one major assumption, as mentioned earlier:

> ***The training set from which the model is built contains a thorough representation of all-possible deformation and movement.***

For simple models this is often true. However, for ASL it is not, and it is important to ask the question,

'What exactly is the temporal model representing?'

The ASL PDF represents two aspects of motion,

1. The non-linear representation of shape space, how the individual clusters relate and how the model moves throughout the space to form letters.
2. It also contains information about the English language and how letters relate to form words and sentences.

As the PDF encodes both of these attributes it must be constructed from a training set which has a good representation of how the model deforms and be representative of the English language. This is however infeasible.

If the ASL image sequence used previously is considered, it took 165 frames to record the 7 letter word 'gesture'. Konheim reported a statistical study where the 1-state transition probabilities of the English Language were determined using 67,320 transitions between two successive letters [Konheim 82]. As the 165 frames previously used produced an average of 20 frames per letter, this would constitute a training set in excess of 1.3 million frames not including transitional shapes between letters. As each frame produces a training shape this results in a training set which is of infeasible size. At 12.5 frames per second it would require almost 30 hours of continuous video capture. Of course smaller numbers of both transitions and frame sampling could be used but would result in a less reliable PDF.

The current ASL PDF (see Figure 7.3.7) contains valuable information about how the model moves within shape space, but due to the deficiency in training it does not contain sufficient information to accurately model the transitions between the letters of the English language. Fortunately, it is relatively simple to gain a transition matrix for the English language as it can be constructed in a similar manner to previously described PDF's by analyzing large samples of electronic text and calculating the 1-state transitions. What is required is a method of combining this knowledge of English into the ASL PDF, producing a more generic and accurate model for tracking and classification.

7.4.2 The Temporal Model

The ASL PDF $P\left(C_i^t \middle| C_j^{t-1}\right)$, constructed from the training set, provides the probability that the model will move to cluster C_i given it was at cluster C_j at the last time step. This is illustrated by Figure 7.4.1, and provides the necessary information of how the model moves within shape space. However, as discussed, this information is incomplete and does not correctly contain the transitional information about the letters and how they relate to form words.

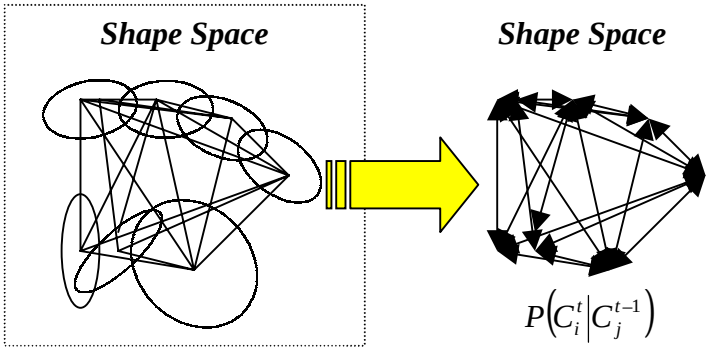


Figure 7.4.1 - Temporal Constraints upon Shape Space for the ASL Model

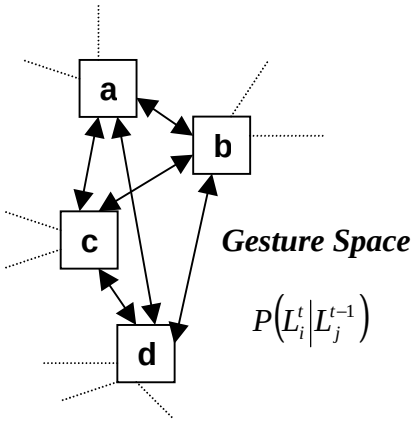


Figure 7.4.2 - 1st Order Markov Chain in Gesture Space

Similarly a 1st order Markov Chain can be constructed for the English language which provides a new PDF $P\left(L_i^t \middle| L_j^{t-1}\right)$ (see Figure 7.4.2). Figure 7.4.3 shows the PDF gained from this Markov Chain as taken from Konheim and shows the 1-

state transitions calculated from a sample text of over 67 thousand letters [Konheim 82].

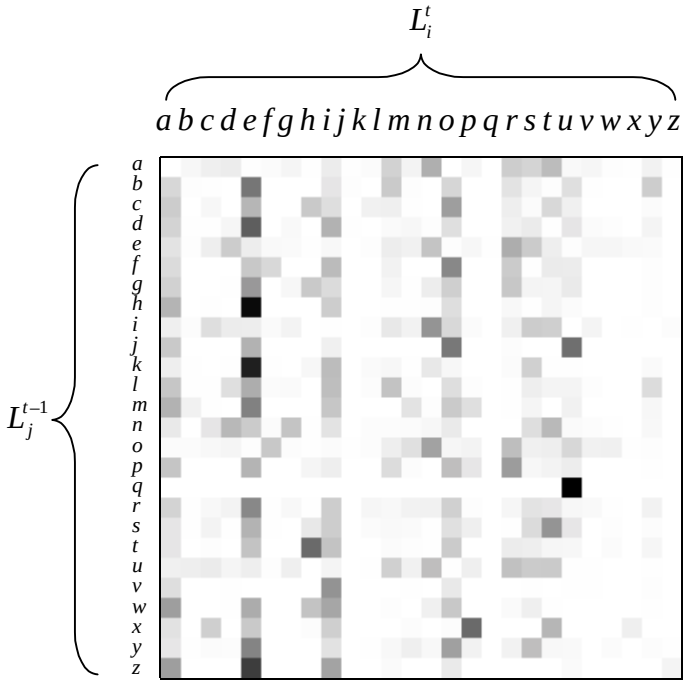


Figure 7.4.3 - Discrete Probability Density Function for the English Language

Figure 7.4.3 does not demonstrate a diagonal dominance, unlike previous PDF's. This is because the English language has few occurrences of repetitive letters in words whereas previous PDFs resulted from operations involving a high degree of repetition. The main trend that can be seen are the vertical stripes that occur for many of the letters. This shows letters which have a high occurrence and are proceeded by almost any other letter in the alphabet. The highest probabilities occur for the letter 'e' confirming that 'e' is the most commonly used letter in the English language. Another observation is the single transition from the row 'q' to the column 'u' as 'q' is always followed by a 'u' in standard English.

In order to incorporate this additional information learnt from sample text, a new ASL PDF must be constructed $P'\left(C_i^t \middle| C_j^{t-1}\right)$. To do this a mapping must be achieved which allows shape space to relate to gesture space.

7.4.3 Extending to a Hidden Markov Model

It has already been shown how a mapping can be achieved between the gesture space and shape space for use in classification (see section 6.5). Here the conditional probability $P\left(L_i^t \middle| C_j^t\right)$ provides a probability of the occurrence of a letter *L* given the model is in cluster *C* in shape space at any time.

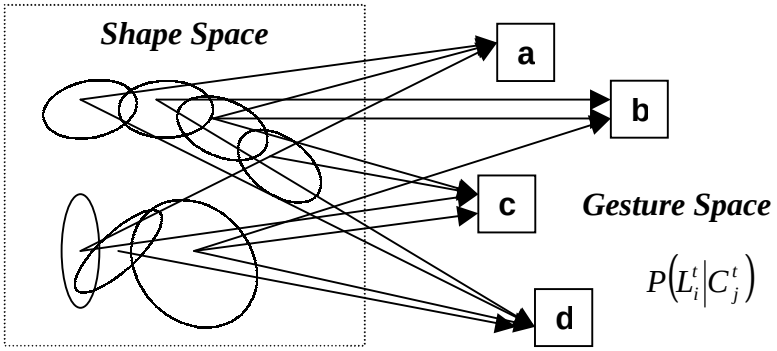


Figure 7.4.4 - Conditional Probabilities Connecting Cluster Exemplars in Shape Space to Specific Letters in Gesture Space

This conditional probability provides a mechanism to relate the shape space to the gesture space where the constraints of the English language (as learnt) can be applied. However, for this to be of use, a method that allows this information to be mapped back into the shape space must be provided. This can be done using the common form of Bayes theorum,

$$P\left(A \middle| B\right) = \frac{P(A)P\left(B \middle| A\right)}{P(B)} or \frac{P(A)P\left(B \middle| A\right)}{\sum P(A)P\left(B \middle| A\right)}$$

Therefore, placing this in the context of the ASL CSSPDM

$$P\left(C_i^t \middle| L_j^t\right) = \frac{P\left(C_i^t\right)P\left(L_j^t \middle| C_i^t\right)}{P\left(L_j^t\right)}$$

However, where $P\left(C_i^t \middle| L_j^t\right)$ and $P\left(C_i^t\right)$ can both be gained from the training set, $P\left(L_j^t\right)$ (the probability of the occurrence of a letter) can only be gained from analyzing English text. As it is known that the training set does not fully represent the English Language this equation would lead to biasing of the final

conditional probabilities. Instead, a variation of Bayes Theorem can be used, where

$$P\left(C_i^t \middle| L_j^t\right) = \frac{P\left(C_i^t\right)P\left(L_j^t \middle| C_i^t\right)}{\sum P\left(C_i^t\right)P\left(L_j^t \middle| C_i^t\right)}$$

Using this form, $\sum P\left(C_i^t\right)P\left(L_j^t \middle| C_i^t\right) \equiv P\left(L_j^t\right)$ but all probabilities are gained from the training set, and hence no bias occurs from mixing unrelated probabilities. This is possible as, although the training set does not contain a thorough representation of English, it does provide an accurate representation of the mapping between the two spaces.

7.4.4 Augmenting the Hidden Markov Model to Increase Constraints

All the necessary tools are now available which allow a new ASL PDF to be constructed which incorporates the 1-state transitions of the English Language.

- $P\left(L_i^t \middle| C_j^t\right)$, is the conditional probability that the model is representing a letter *L* at time *t*, given the CSSPDM is in cluster *C* and time *t*.
- $P\left(C_i^t\right)$, is the probability of the occurrence of cluster *C*.
- $P\left(C_i^t \middle| L_j^t\right) = \frac{P\left(C_i^t\right)P\left(L_j^t \middle| C_i^t\right)}{\sum P\left(C_i^t\right)P\left(L_j^t \middle| C_i^t\right)}$, is the conditional probability that the CSSPDM is in cluster *C* at time *t*, given the current letter that is being represented is *L*.
- $P\left(L_i^t \middle| L_j^{t-1}\right)$, is the 1-state transition that a letter L_i will occur given the previous letter was L_j.

A new ASL PDF can therefore be constructed which incorporates the 1-State transitions of English, by

1. Taking the current cluster of the model
2. Calculating the corresponding letter(s) associated with this cluster
3. Applying the 1-state transition matrix to extract the most likely next letter
4. Then locating the cluster(s) associated with this transition.

Where,

$$P'\left(C_i^t \middle| C_j^{t-1}\right) = P\left(L_i^t \middle| C_j^t\right) P\left(L_i^t \middle| L_j^{t-1}\right) P\left(C_i^t \middle| L_j^t\right)$$

This produces a new ASL PDF which is shown in Figure 7.4.5.

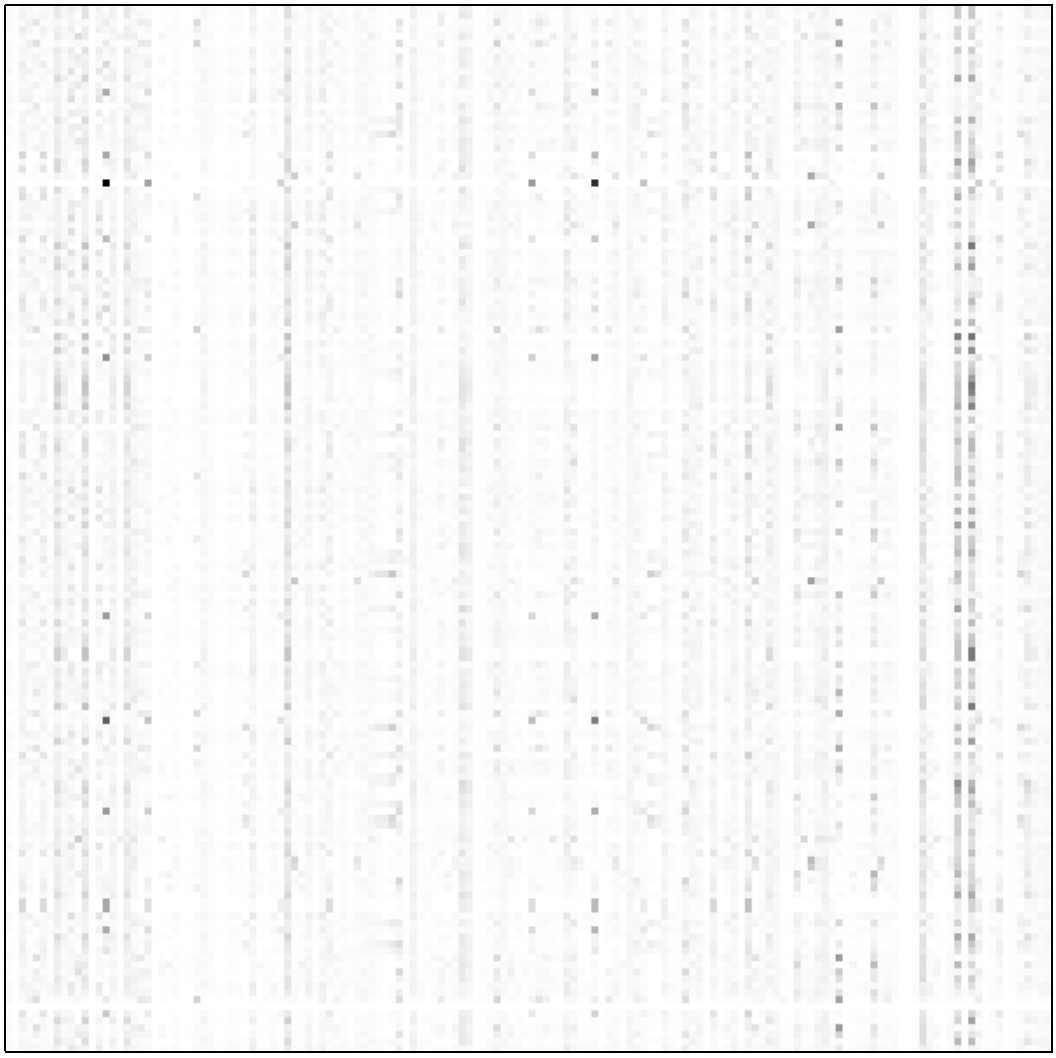


Figure 7.4.5 - Discrete Probability Density Function for derived ASL Model

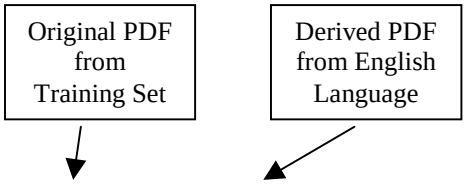
Figure 7.4.5 demonstrates the same characteristic vertical strips seen from the English Language PDF, which it has inherited, and as such differs from the original ASL PDF in two ways.

1. Each cluster exhibits far more transition to other clusters.
2. The diagonal dominance, which is important to tracking, is missing.

Diagonal dominance can be forced upon the PDF by imposing diagonal dominance on either $P\left(L_i^t \middle| L_j^{t-1}\right)$ or $P'\left(C_i^t \middle| C_j^{t-1}\right)$. However, this is haphazard and risks over-biasing the hypothesis generated at each frame. An alternative is to simply ensure that the population generated at each step always includes at least one hypothesis from the current cluster.

In order to explore the validity of these assumptions and assess the success of the derived PDF a new set of tests were performed upon the 'gesture' image sequence.

The PDF used for each test was the weighted sum of the original PDF gained from the training set and the derived PDF from English, where



$$P''\left(C_i^t \middle| C_j^{t-1}\right) = (1-\alpha)P\left(C_i^t \middle| C_j^{t-1}\right) + \alpha P'\left(C_i^t \middle| C_j^{t-1}\right), \text{ for } 0 \le \alpha \le 1$$

and hence

$$P''\left(C_i^t \middle| C_j^{t-1}\right) = (1-\alpha)P\left(C_i^t \middle| C_j^{t-1}\right) + \alpha P\left(L_i^t \middle| C_j^t\right)P\left(L_i^t \middle| L_j^{t-1}\right)P\left(C_i^t \middle| L_j^t\right)$$

Using this method, the performance of both approaches can be assessed. Figure 7.4.6 shows the results of varying α. When $\alpha = 0$ the PDF is that gained from the training set; but as α increases, the resultant error rate decreases. When $\alpha = 0.6$ the resulting error rate is only slightly higher than that produced by the optimum path shown in Figure 7.3.4. However, as α approaches 1 an increase in error rate results. This is attributable to the absence of diagonal dominance for the derived PDF, and hence lack of support for hypotheses that remain static within shape space. However, even in light of this fact, the overall error is still lower than that gained form the original ASL PDF.

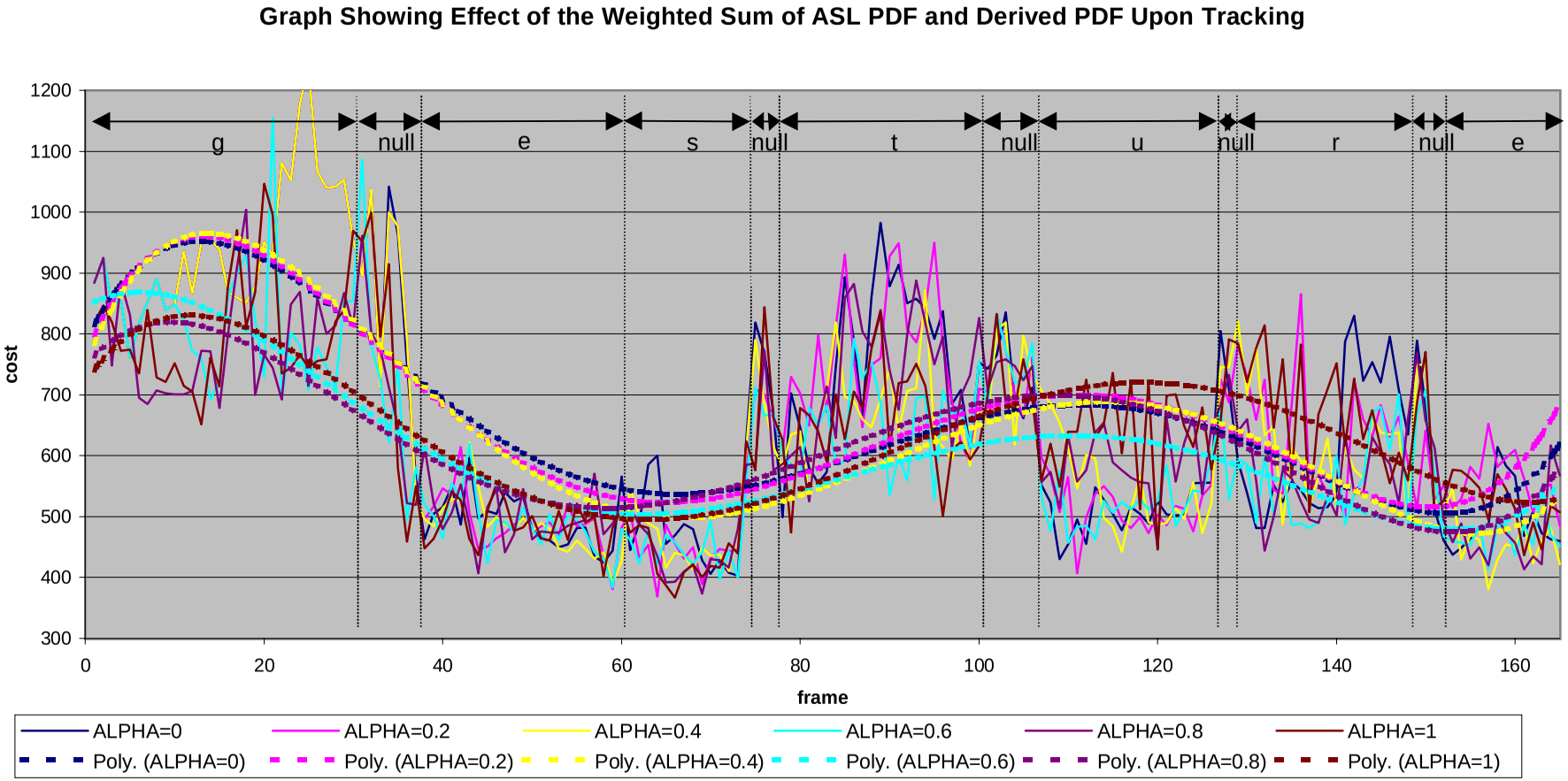
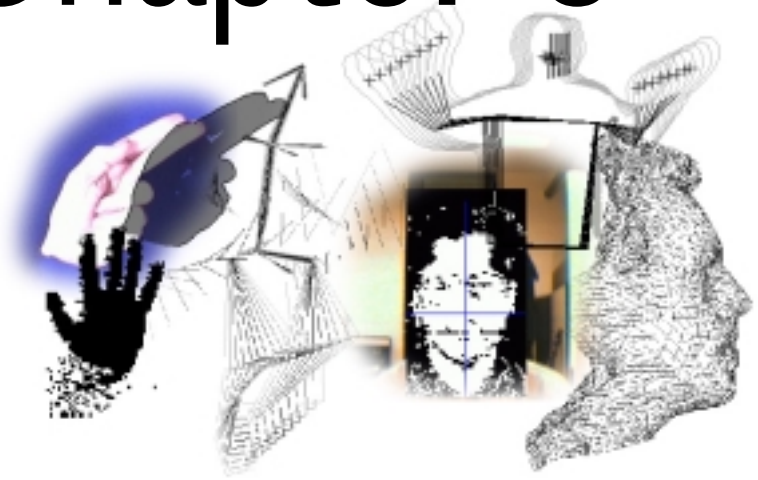


Figure 7.4.6 - Graph Comparing Simple Condensation Against Weighted Condensation

7.5 Conclusions

This chapter has looked at augmenting statistical models with temporal dynamics gained through the probabilistic analysis of the training set and how this relates to movement within shape space. It has been shown how the discrete segregation of shape space used in the CSSPDM directly lends itself to a Markov chain approach to modeling temporal dynamics. This additional analysis has been used to reproduce motion indicative of the training sets in the form of key frame animations and how the motion of the CSSPDM can be further constrained during tracking. It has been shown that the nature of shape space is often complex and discontinuous and how, using these additional learnt temporal constraints, tracking can be improved by supporting a population of multiple hypotheses. Lastly a method of combining additional constraints into the model was presented which provides more robust tracking and classification, while reducing the necessity for large training sets.

Chapter 8



8 3D Point Distribution Models

8.1 Introduction

It has thus far been demonstrated how a Point Distribution Model can be constructed for a 2D contour or shape (Chapter 3) and grey scale images (Section 5.5.2). Chapter 7 introduced a simple 3D PDM in the form of a stick human figure. This chapter will extend upon this to 3D eigensurface models which are constructed from polygonal surface representations and are the analogous extension into 3D of the 2D contour.

For a 2D contour, consisting of *n* points, a training example **x** is constructed by concatenating the constituent points of the contour into a single *2n* vector $\mathbf{x} \in \Re^{2n}$. As was shown in section 7.2, for 3D the procedure follows a similar procedure. Each point of the model differs only in its dimensionality. Therefore a 3D model consisting of *m* points (vertices) will form a vector $\mathbf{x} \in \Re^{3m}$. In chapter 7, where the 32 points consisted of key-points of a simple human skeletal model, this produced a 96 dimensional vector. However, more realistically the target data represents a surface, where each vertex of the surface represents a key point within the model. This results in extremely high dimensional spaces i.e. for a 3D mesh of 100 x 100 points, $\mathbf{x} \in \Re^{30000}$. Under these conditions it is often the

case that the number of training examples is less than the dimensionality of **x**, and hence technique 2 for PCA (detailed in section 3.2.5) is invaluable in the construction of 3D PDMs.

Although the construction of 3D PDMs is a simple extension to the 2D case, one of the major problems associated with their construction is the acquisition of training data and its alignment. Due to the complexity of constructing 3D surfaces by hand, automated procedures are essential. As has been discussed in chapter 8, many techniques such as isosurfacing produce complex discontinues surfaces which are unsuitable for statistical analysis. These 3D surfaces must be aligned and resampled in a similar manner to the 2D contour. However, the problem is compounded by high dimensionality and the resulting computational complexity of the procedure.

Section 8.2 demonstrates the construction of a 3D PDM using a synthetic drinking glass example. Sections 8.3 will show how this can be extended to real data and describe approaches to the resampling and alignment problem in 3D. This will be demonstrated by a 3D PDM of a human head. Finally conclusions will be drawn.

8.2 The Eigen Glass Model

8.2.1 Introduction

Point Distribution Models attempt to model the deformation of a class of objects or shapes with simple statistical analysis. The example shown here is that of a class of drinking vessels. This synthetic example data provides a data set with which to explore the construction of 3D PDMs and will be used in chapter 10 as an example for statistical inference.

8.2.2 Constructing the Training set

The eigen Glass training set consists of 7 types of glass shape (see Figure 8.2.1). Each example was created by sweeping a 2D contour around a central y-axis.

This forms a rotationally symmetric glass of varying shape and size. Since each example was constructed in a similar manner, with the same number of rotational steps and points along the contour, each example contains the same number of vertices.

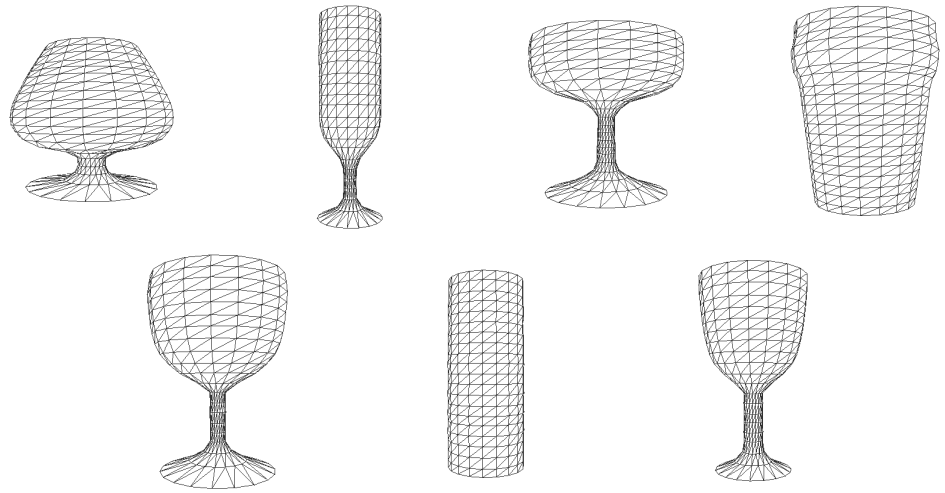


Figure 8.2.1 - Eigen Glass Training Set

The acquisition of the training set provides examples that have a direct correspondence of landmark points and therefore no further alignment or resampling is necessary.

8.2.3 Building the Eigen Model

Each glass example consists of 440 vertices which, when converted to a vector, produces a training example $\mathbf{x} \in \Re^{3n} \Rightarrow \Re^{1320}$. As there are only seven examples in the training set, technique 2 (section 3.2.5) results in a large computation saving during shape analysis. The use of this technique allows decomposition to be performed upon a 7x7 matrix. This produces a significant computational saving over performing a full decomposition upon the 1320 x 1320 covariance matrix.

Figure 8.2.2 demonstrates the primary 3 modes of variation of the resulting 3D PDM rendered in wire frame with hidden line removal. The primary mode is also shown in Gouraud shaded form. The maximum number of modes of deformation for the model is 6 (ie. 100% of the deformation present within the training set is

contained within the first 6 eigenvectors). This is because the number of eigenvectors can never exceed *N-1*, where *N* is the number of training examples. In fact, 99% of the deformation is contained within the primary 4 modes of variation.

This high reduction of the shape space is similar to that shown in earlier cases. However, it is important to note that, due to the rotational symmetry of each of the objects, the training examples contain no additional information after the contours had been swept into a 3D surface. The model could equally have been constructed by performing PCA upon the original contours and sweeping the reconstructed contour, generated from the PDM, around the central axis. This is demonstrated in Figure 8.2.3 where PCA has been performed upon the contours and the resulting 2D PDM extracted.

If Figure 8.2.3 is compared to Figure 8.2.2, it should be apparent that the deformation contained in the modes of variation of the 2D PDM are exactly the same as those of the 3D object. Since both models contain the same information the resulting PDMs have the same characteristics with a total of 7 modes where the first 4 encompass 99% of the deformation. The redundant dimensionality introduced when the contour is swept into a 3D surface does not introduce any additional information and this additional dimensionality is disregarded by PCA demonstrating that both models lie upon the same dimensional sub space.

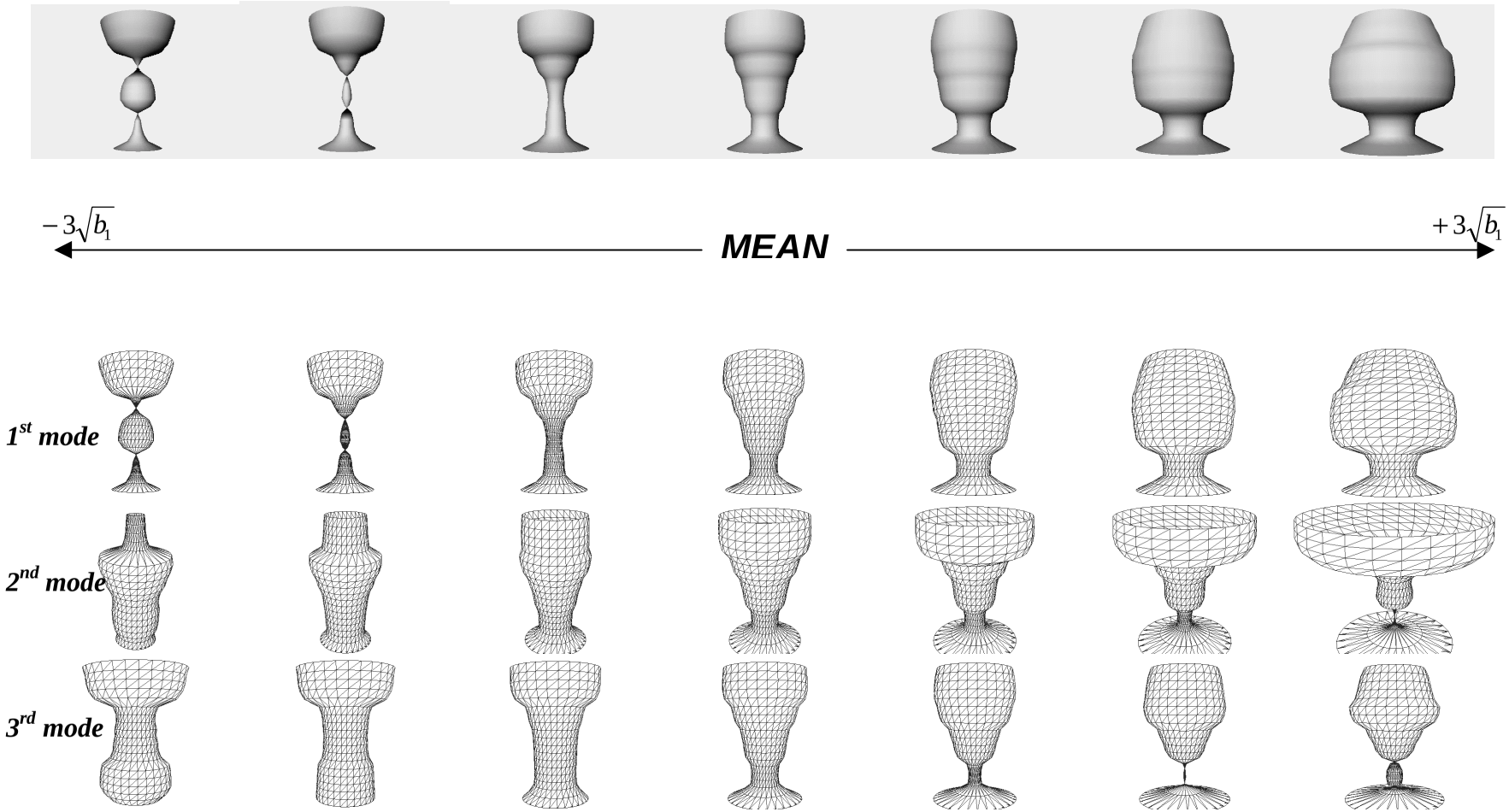
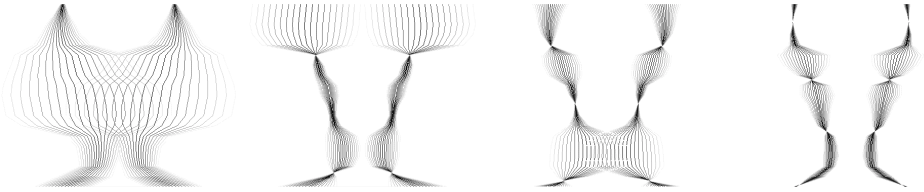


Figure 8.2.2 - The Primary Modes of the eigenGlass Model



1st mode *2nd mode* *3rd mode* *4th mode*

Figure 8.2.3 - The Primary Modes of the 2D eigenGlass Model

8.3 Resampling Meshes

8.3.1 Mesh Alignment

In the previous synthetic eigenGlass demonstration, the simplicity of construction was due to the direct correspondence of landmark points throughout the training set and the artificial way in which it was created. However, this is seldom the case and to ensure the construction of a PDM is successful, careful alignment and resampling must be performed to provide a good correspondence of landmark points between examples.

As with the 2D contour, to ensure a good correspondence between training examples each must be aligned. Techniques like those presented by Cootes *et al* [Cootes 95] for 2D alignment become infeasible due to the high dimensionality of the models. A similar, but less time consuming, alignment process can be performed by treating it as an optimisation problem, solved using an approach to optimisation such as Simulated Annealing or Genetic Algorithms. Such approaches rely upon a fitness function being formulated which assesses what is a good (optimum) match.

For two meshes $\mathbf{x}$ and $\mathbf{y}$, where
$\mathbf{x} = \left(\mathbf{v}1_1^{xyz}, \mathbf{v}1_2^{xyz}, \ldots, \mathbf{v}1_n^{xyz}\right)$, $\mathbf{y} = \left(\mathbf{v}2_1^{xyz}, \mathbf{v}2_2^{xyz}, \ldots, \mathbf{v}2_m^{xyz}\right)$ and $\mathbf{v}_n^{xyz} \in \Re^3$ is the n^{th} vertex of the mesh, a suitable fitness function to be minimised would be the mean distance between the vertices of each mesh,

$$f = \frac{1}{n}\sum_{i=1}^{n}\min_{j=1}^{m}\left\|\mathbf{v}1_i^{xyz} - \left(M\left(s_x, s_y, s_z, \theta_x, \theta_y, \theta_z\right)\left[\mathbf{v}2_j^{xyz}\right] + t_{xyz}\right)\right\|$$

where s_x is a scaling in x, θ_x is a rotation around x, and $t_{xyz} \in \Re^3$ is a translation vector in Euclidean space.

However, this function must be assessed for each pose $\left(s_{xyz}, \theta_{xyz}, t_{xyz}\right)$ of the model in order to find the optimum mapping of one mesh to another and quickly becomes an unfeasible solution as the size of the mesh increases. In addition to this complexity, the procedure must be repeated for all meshes in the training set.

If known features exist upon the surface and the position of these features can be accurately located (such as large planar segments or areas of high curvature), these features can be used in the fitness function rather than every vertex of the mesh.

The simplest method of alignment is similar to that suggested in Section 3.2.4 where the mesh is treated as a cloud of points in $\Re^3$. The centre of gravity of the cloud, $\mathbf{C}^{xyz}$, can then be calculated and subtracted from each vertex to translate the mesh to the origin, where

Equation 8.3-1
$$\mathbf{C}^{xyz} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}_i^{xyz}$$

To normalise the mesh, and hence avoid numerical instability during PCA, each vertex is then scaled by the mean distance of all the vertices from the origin, where

Equation 8.3-2
$$\left|\mathbf{v}_i^{xyz}\right|' = \frac{\mathbf{v}_i^{xyz} - \mathbf{C}^{xyz}}{l(\mathbf{v})} \text{ , and } l(\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n}\left|\mathbf{v}_i^{xyz} - \mathbf{C}^{xyz}\right|$$

By then performing PCA upon the cloud (as done in 3.2.4) principal moments of the shape and therefore the primary axes can be extracted. Once done, the shape can be projected onto these axes to align the principal moments of the shape with

the axes of Euclidean space. Providing the shape does not vary too extensively this approach provides a fast and simple method for object alignment and scaling.

8.3.2 Nearest Neighbour Resampling

Once all training examples have been aligned, they must be resampled to provide a direct correspondence for each vertex, and the associated connectivity across all training examples. It is also important that each example has the same number of vertices so that all training examples have the same dimensionality.

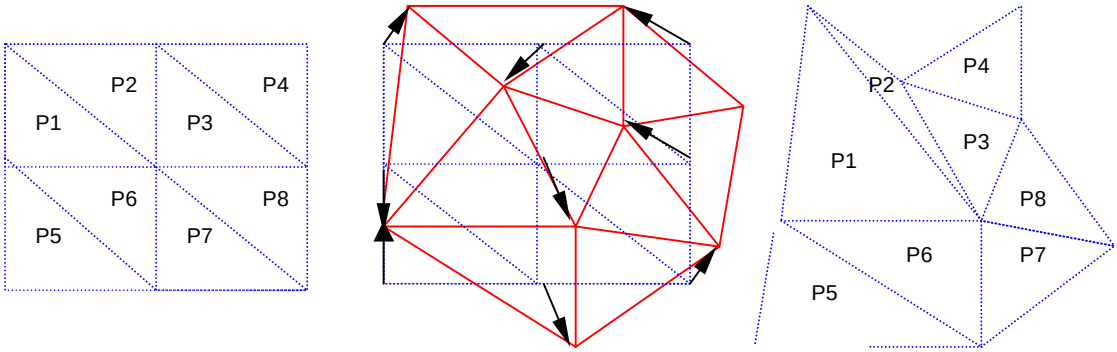


Figure 8.3.1 - Nearest Neighbour Resampling

This can be accomplished by taking a known mesh and deforming it to fit to each example in turn. Figure 8.3.1 demonstrates this procedure using a nearest neighbour approach, a regular mesh (blue) is constructed which has a known number of vertices and connectivity. The regular mesh is then deformed by moving each vertex to the closest vertex of a training example (red) in $\Re^3$. The resulting mesh has the same basic overall shape of the training example but has the connectivity and number of vertices of the regular mesh. This procedure can be repeated for each aligned training example to provide a consistent training set on which statistical analysis can be performed. However, this procedure results in the loss of information as the regular mesh may not contain the local density of vertices required to successfully model high curvature. If the number of polygons is increased further to accommodate this, then unnecessary dimensionality is introduced for areas of low curvature. This approach also introduces problems when mesh elements on the regular mesh are smaller than

those on the training example mesh. Under these circumstances multiple vertices of the resampled mesh may be attracted to a single vertex resulting in polygons of zero area (this will be shown shortly).

Another major disadvantage is that the procedure relies upon the correct alignment of the training examples. If sufficient difference is present between examples then it is possible that vertices will be assigned to completely unrelated features across the training set. This effect can be minimised by utilising the assumption that training examples do not vary extensively between individual examples, although the overall variation may be considerable. Using this assumption a mesh can be deformed to fit a training example and the same mesh applied to the next example until the whole training set has been processed. However, this approach requires user intervention to ensure that an optimum ordering is used for the resampling sequence.

8.3.3 K-nearest Neighbour Resampling

An alternative approach is to use a variation of a clustering algorithm. This results in a consistent mesh with known connectivity, but provides the advantage that vertices on the resampled mesh attempt to best mimic the local features of the surface by averaging the position of the vertices locally.

A mesh $\mathbf{y} = \left(\mathbf{v}1_1^{xyz}, \mathbf{v}1_2^{xyz}, \ldots, \mathbf{v}1_k^{xyz}\right)$ of known connectivity and size *k* is to be fitted to second mesh $\mathbf{x} = \left(\mathbf{v}2_1^{xyz}, \mathbf{v}2_2^{xyz}, \ldots, \mathbf{v}2_m^{xyz}\right)$ of variable size *m*. The vertices of **x** are treated as a cloud of points in $\Re^3$ and the vertices of **y** as exemplars in a k-means algorithm (see Appendix 1). Each vertex of **x** is assigned to an exemplar of **y** in a nearest neighbour sense using the crisp membership function

Equation 8.3-3 $$u_j\left(\mathbf{v}2_i^{xyz}\right) = \begin{cases} 1 & if \left|\mathbf{v}2_i^{xyz} - \mathbf{v}1_j^{xyz}\right| = \min\left|\mathbf{v}2_i^{xyz} - \mathbf{v}1_j^{xyz}\right| \\ 0 & otherwise \end{cases}$$

Each vertex of y is then moved to minimise the distance from its assigned members where

Equation 8.3-4

$$\mathbf{v}1_i^{xyz} = \frac{\sum_{i=1}^{m} u_j\left(\mathbf{v}2_i^{xyz}\right)\mathbf{v}2_i^{xyz}}{\sum_{i=1}^{m} u_j\left(\mathbf{v}2_i^{xyz}\right)} \quad \textbf{and} \quad j = 1,2,\ldots,k$$

This procedure is repeated until the total displacement of the vertices of **x** has dropped below a threshold (i.e. equals zero); at this point the algorithm has converged upon a solution.

8.3.4 K-cluster Elastic Mesh

Both nearest neighbour and k-nearest neighbour approaches are subject to the same problem i.e. the incorrect convergence on local minima. This is largely a problem of model initialisation. Features upon the meshes must be close if a good correspondence is to be achieved as each vertex is only attracted to the closest corresponding point in both techniques. Again, this approach places a large emphasis on the accurate alignment of examples.

This can be overcome to an extent by extending the k-nearest neighbour approach to an elasticised k-cluster approach, which provides the same mechanism for local resampling, but allows global constraints to be placed upon the shape of the mesh.

In addition to the local attraction of the regular mesh to vertices upon the training mesh, elastic properties are added to the connectivity as described in Section 8.5. As the mesh is deformed to fit the training data the elasticity of the mesh attempts to retain as small and as planar a mesh as possible, thus smoothing the mesh and ensuring that the connectivity is preserved.

If the elastic force from section 8.5.2 (equation 8.5-4) is taken and placed in the context of the mesh **y,** the displacement of a node $\mathbf{v}1_i$ from the elastic force is

Equation 8.3-5

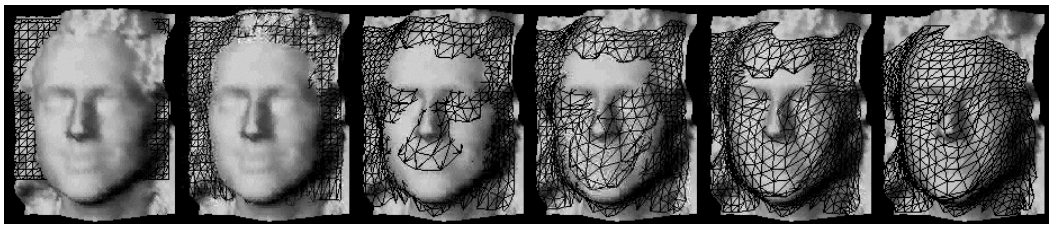
$$\Delta s_i = \frac{\alpha}{n} \sum_{j=0}^{n} r_{ij}$$

where α is the stiffness, $r_{ij} = \mathbf{v}2_j^{xyz} - \mathbf{v}2_i^{xyz}$ the vector separation of two connecting nodes and p is the number of nodes connecting to node $\mathbf{v}1_i$. Combining this force with that of the k-means displacement (Equation 8.3-4) the total movement of a the node $\mathbf{v}1_i$ at each iteration is

Equation 8.3-6

$$\mathbf{v}1_i^{xyz} = \frac{\sum_{i=1}^{m} u_j \left(\mathbf{v}2_i^{xyz}\right) \mathbf{v}2_i^{xyz}}{\sum_{i=1}^{m} u_j \left(\mathbf{v}2_i^{xyz}\right)} - \Delta s_i$$

In order to balance the attraction force and the surface tension of the mesh a weighting parameter which balances the two influences is required. However, the stiffness parameter α can be used for this purpose as it controls the strength of surface tension. This weighting parameter determines the influence of the two forces on the movement of the mesh. When $\alpha = 0$ the mesh operates as the k-nearest neighbour resampling procedure described earlier. When $\alpha \to \infty$ the mesh will not converge on any solution, remaining rigid. Upon initialisation the force is set to allow surface tension to dominate i.e $\alpha = 2$. This parameter and hence the effect of surface tension is decreased at each iteration of the procedure allowing the surface to deform to the data while retaining the constraints of connectivity.



i=0 *i=5* *i=10* *i=15* *i=20* *i=25*

Figure 8.3.2 - Elastic k-cluster mesh

Figure *8.3.2* demonstrates the use of the elastic *k*-cluster mesh technique to resampling a surface of a human head. The shaded surface consists of an irregular mesh consisting of 3896 vertices, which represent the shape of a face. The wire frame mesh is a flat regular tri-mesh of known connectivity and 625 vertices. The flat wire frame mesh is located close to the face mesh and is rendered slightly in front so the shape can be seen as the algorithm iterates. At each iteration α is decreased by 10% and after 25 iterations ($i=25$) the wire frame mesh has deformed to best fit the original face mesh while retaining its connectivity and smoothness. Without this elastic surface tension which smoothes the resulting surface, the mesh would instantly crease and deform as the initial attraction of the k-means algorithm is initially very large. As k-means will only find a local optimum, this initial creasing of the surface remains throughout the fitting. The elasticity ensures that the mesh retains its original shape and connectivity while trying to best deform to resample the mesh.

However, this approach has two major drawbacks

1. The speed of the algorithm is prohibitive, as the computation complexity at each iteration is considerable for even the simplest of surfaces.
2. The rate at which the weighting parameter is decreased is an unknown. Since the rate at which the parameter decreases is responsible for the number of iterations required (and hence the overall speed), an optimum rate must be determined which provides the best time to convergence while allowing the correct convergence on the shape. This is similar to the annealing schedule used in simulated annealing but is beyond the scope of this work.

8.4 3D Head PDM

8.4.1 Constructing the Training set

To illustrate the alignment and construction of a 3D PDM, a model of the human head was built. The head data set consists of 25 surface meshes of varying size and structure acquired using a C3D[10] scanning device. Each mesh has between 4000 and 5000 vertices and differing local mesh densities modelling local

curvature. The examples were first aligned using the alignment procedure outlined in section 8.3 such that each lies within a left handed co-ordinate system with the z-axis is aligned with the direction of the gaze of the face. Once done, each mesh was translated to ensure that the apex of the nose was at the origin. The nose can easily be estimated as the point on the mesh which has the greatest z-value. Each mesh was then normalised to lie within a unit cube as shown in Figure 8.4.1.

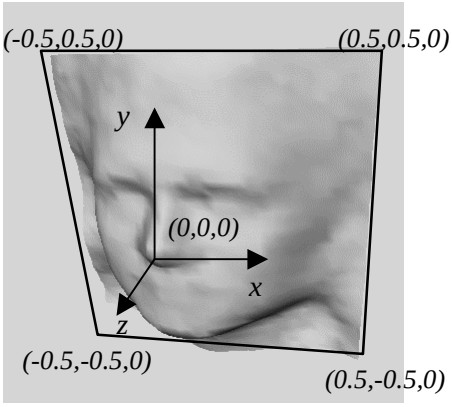


Figure 8.4.1 - Aligning the Face Training Set

Once all the example meshes have been transformed in this way, the next step is to resample each to a uniform mesh structure. A regular triangular faceted mesh was generated as shown in Figure 8.4.2. The regular mesh consists of 1849 vertices and is a unit square with its centre at the origin and aligned with the *x* and *y*-axis.

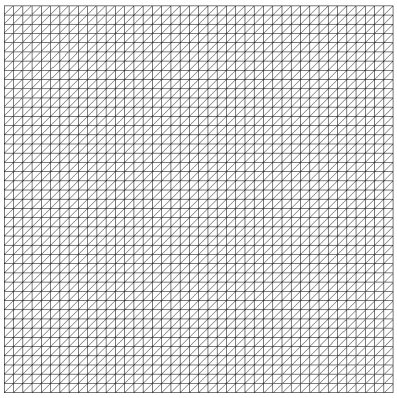


Figure 8.4.2 - Regular tri-mesh

[10] C3D Scanner model courtesy of the Turing Institute, all head models are freely available via the web at http://www.turing.gla.ac.uk

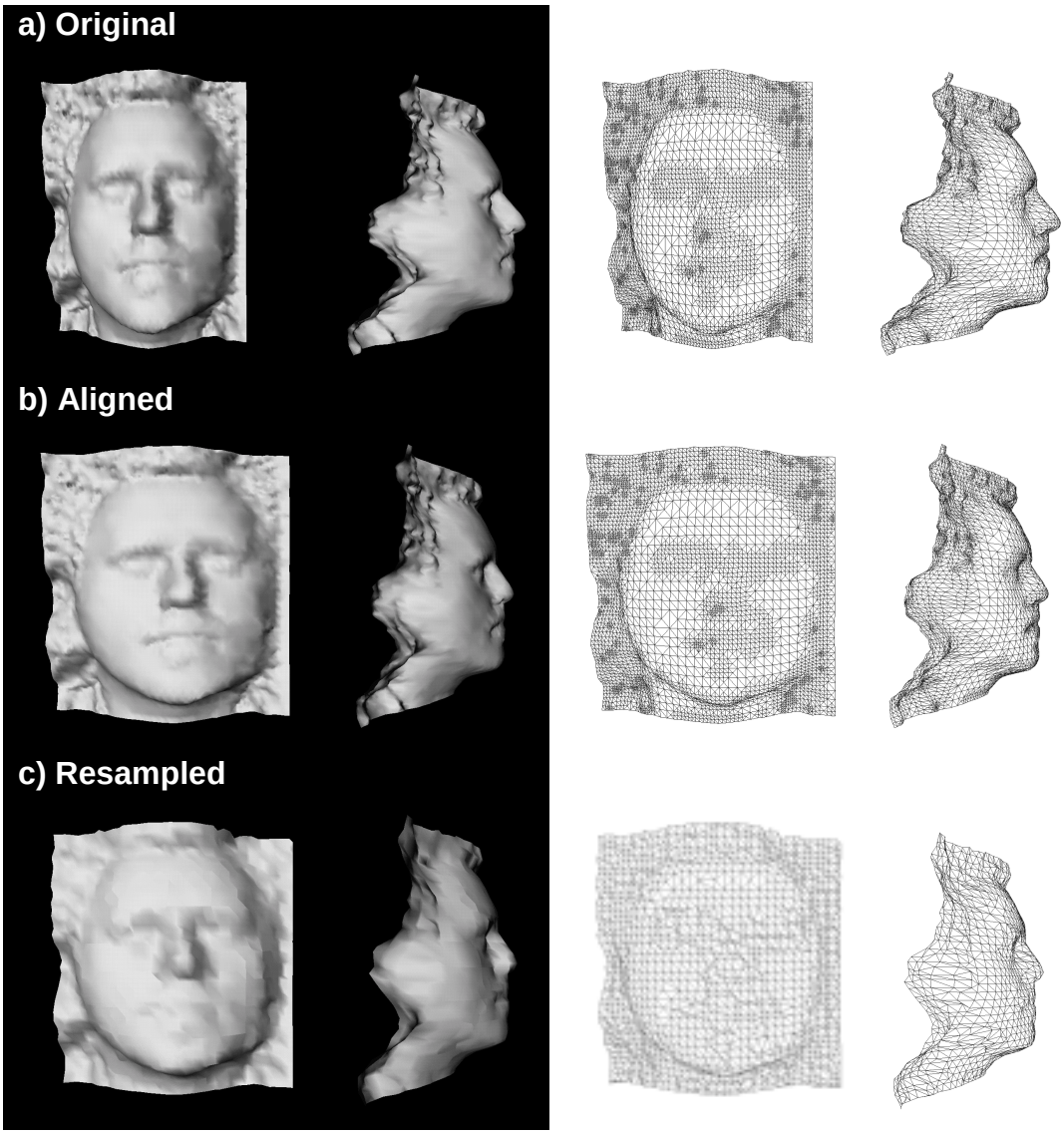


Figure 8.4.3 - Resampling a 3D Mesh

(a) The original mesh (b) The aligned mesh (c) The resampled mesh

For each mesh in turn, the regular mesh is deformed to fit using the nearest neighbour approach described previously. Figure 8.4.2 shows the regular mesh, Figure 8.4.3 (a) the original training example, (b) shows the aligned mesh, and (c) shows the resampled mesh after each vertex has been deformed to fit the example. It should be noted that the final resampled wire frame mesh does not look dissimilar to the original. However, the shaded version shows a step effect to the mesh. This is due to two reasons

1. The local surface density of patches is not optimum to model the curvature hence areas of high curvature have less polygons and consequently a less smooth appearance i.e. the number of vertices has been reduced from around 5000 to 1849.
2. Many polygons have zero area. Where this occurs normal calculations are ill-defined and hence Gouraud shading fails and reverts to a flat shading algorithm.

The problem of zero area polygons, where multiple vertices of the regular mesh have been assigned to a single vertex on the example mesh, is one of the disadvantages that were mentioned in section 8.3. It is not possible to simply remove these polygons as all training examples must have the same dimensionality. A polygon could therefore only be removed if it had zero area in all training examples. However, it will be shown later that the smoothing properties of PCA will remove some of these inaccuracies (see section 8.4.2).

8.4.2 The Face Eigen Model

Upon completion of the resampling procedure a training set is now available on which statistical analysis can be performed. The results of which can be seen in Figure 8.4.4. However, it is difficult to see the overall effect of these modes of deformation except at the extremities of the eigenvectors where the greatest deformation is apparent. Figure 8.4.5 shows the primary 21 eigenvectors corresponding to the 21st largest eigenvalues which encompass 99.998% of the deformation. Each mode is colour coded to represent the deformation. Red, Green and Blue coloured areas represent deformation in x,y and z respectively. The intensity of the image is proportional to the size of the local deformation.

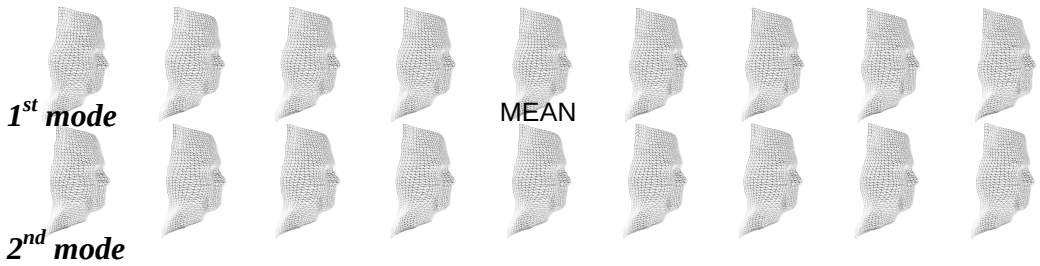


Figure 8.4.4 - Primary two modes of the 3D eigenFace model

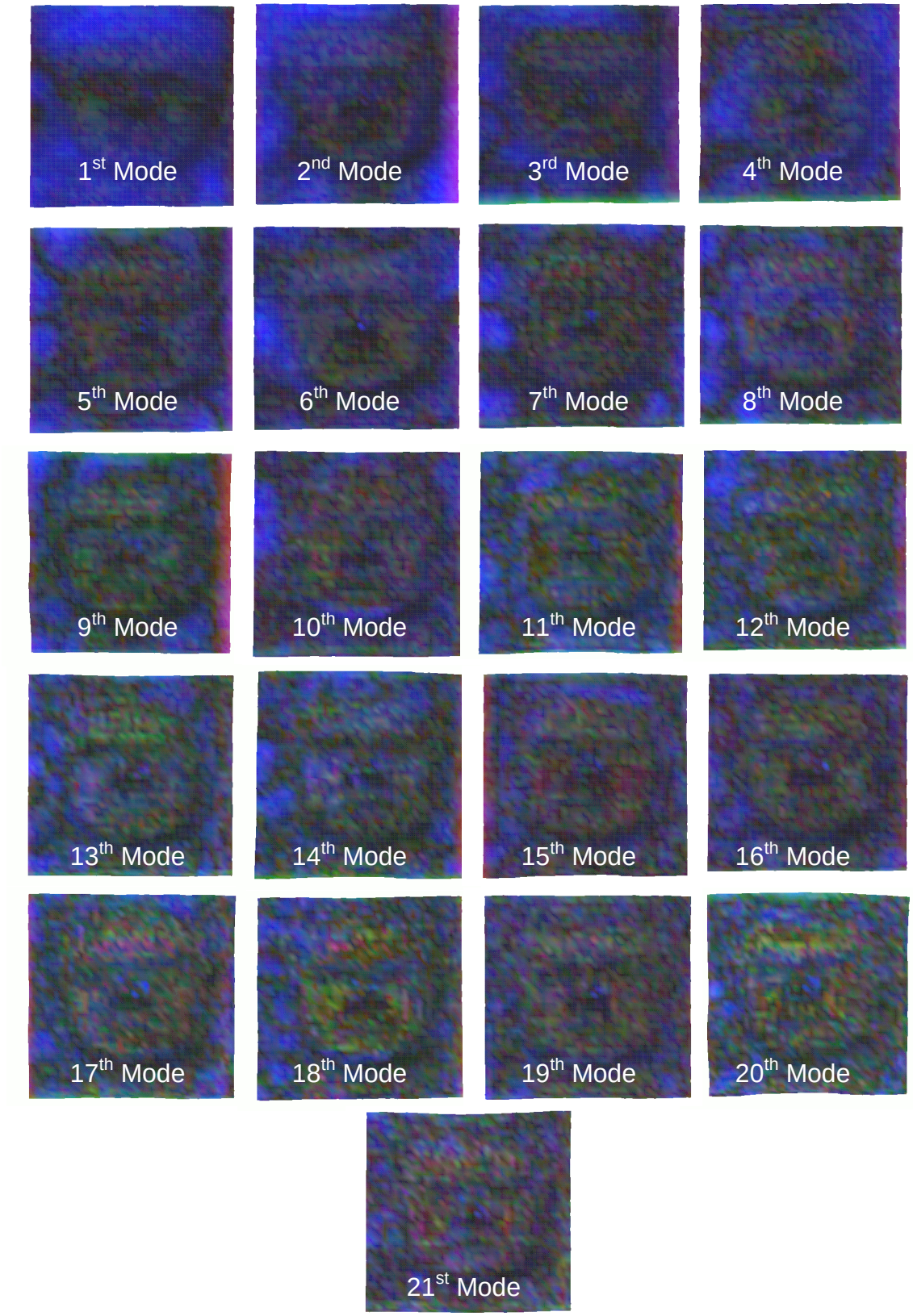


Figure 8.4.5 - Colour map showing deformation of primary modes for eigenFace model

By examining these colour maps it is far easier to infer specific functions for various modes. From the shading on the 6th mode it can be deduced that this mode is responsible for the movement of the eyebrows and cheek areas. The 8th mode however is clearly responsible for the movement of the eyes and mouth. It can be seen that the primary mode contains mainly deformation in 'z' along the top and bottom of the mesh surface. This is due to the large variation in background depth, hair and neck between individual examples. Indeed, the primary modes display large areas of blue showing that they mainly contribute to the depth information of the mesh. As the number of the modes increases a more speckled effect is observed. These effects are the high frequency oscillations, which are typically picked-out by the lower modes of variation. However, much of these high frequency oscillations are due to the nearest neighbour resampling which resulted in zero area polygons.

The original training example mesh size were of the order of 5000 vertices. With 3 dimensions for each vertex this generates examples in a 15000 dimensional space. Resampling each example to a mesh with 1849 vertices provides a consistent dimensionality of 5547 throughout the entire training set. However 90% of the deformation is contained within the primary 10 modes of variation. So, although the training set was originally in 15000 dimensional space, the data actually lies upon a subspace of only 10 dimensions. The most important aspect of the PDM is the predominant z-deformation (blue) in these primary 10 modes. This demonstrates that the alignment and resampling procedure has been successful. During resampling the simplicity of the resampling scheme lead to zero area polygons. After PCA these do not occur as vertices are statistically smoothed by the model. The perturbations of vertices in the x-y plane, which were generated by zero area polygons, are expressed within the lower modes of variation and effectively removed from the model.

8.5 Conclusions

This chapter has demonstrated how the techniques for the assembly of 2D PDMs can easily be extended to 3D. Approaches to the alignment and resampling

procedure have been proposed and a 3D PDM of a human face constructed. Due to the high dimensionality and corresponding complexity of these techniques, variations on the resampling method have been proposed which can be used depending upon the extent and complexity of the training data. It has also been demonstrated that errors introduced during resampling are statistically smoothed and manifest themselves as high frequency oscillations of the model contained within the lower modes of deformation. Since these lower modes are typically discarded it can be deduced that the smoothing effect of the PDM can help reduce errors introduced during assembly.

Future work is to apply these techniques to volumetric segmentation techniques detailed in Appendix 2 to construct 3D PDMs from medical imaging data.

Chapter 9

9 Extending the Point Distribution Model

9.1 Introduction

Thus far, statistical models of deformation have been considered where the vector **x** consists of related features such as the co-ordinates of a connected contour, the vertices of a surface or the grey level intensity of each pixel of an image. The principle relies upon the variation of elements with regard to others and attempts to generalise the **relative** movement of the constituent components. It therefore holds that similar statistical linkage of features could be achieved even if they lie within different co-ordinate frames and represent quite different elements providing that there is still some linear relationship between the various elements. This chapter will discuss the use of this technique to link together related information from differing sources. Section 9.2 will discuss combining shape information with abstract parameters and using this to infer unseen information from examples of shape. Section 9.3 will present the application of this technique to inferring the shape and position of a human body from an image sequence. Finally conclusions will be drawn.

9.2 Combining Features Statistically

9.2.1 A Linear PDM with an Abstract Parameter

The linear 3D PDM of an eigenGlass, as constructed in chapter 9, provides an ideal example to demonstrate the hypothosis that related information can be combined into a PDM. It has already been shown that this PDM is essentially the same as the 2D contour of the glass due to the rotational symmetry of the object. Thus, the two dimensional vector that describes the glass profile can be used as a training vector for PCA and the final reconstructed model swept around the central axis to attain the full 3D model. This training vector **x** describes the shape of the glass for each example in the training set. However, additional parameters can be concatenated to the vector for each example in the hope that some mapping which links the shape with other features can be achieved. For each training example an abstract parameter *MF* was estimated. The parameter corresponds to the masculinity or femininity of a specific training example. This provides a rather subjective scale but provides an illustrative demonstration that a link between shape and aesthetic appearance can be achieved. Figure 9.2.1 shows each training example with the corresponding *MF* parameter estimated, $0 < MF < 1$, where 0 corresponds to feminine and 1 to masculine.

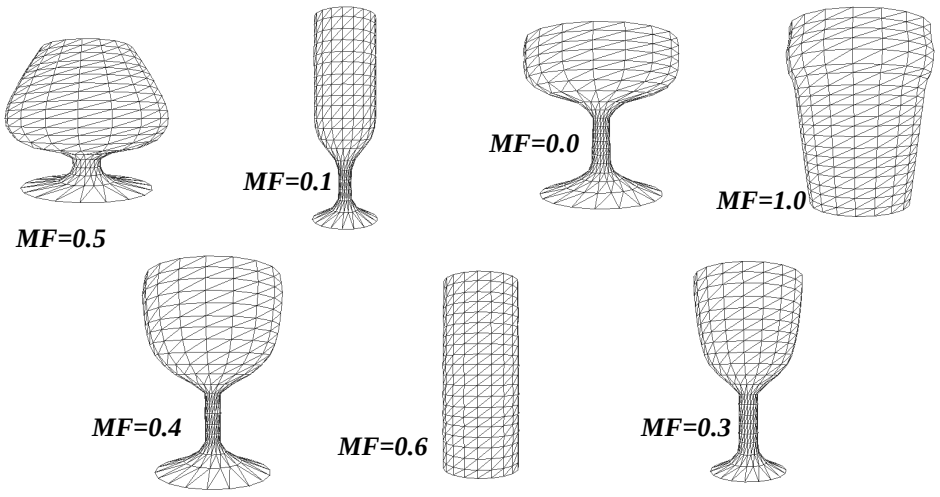


Figure 9.2.1 - MF Parameter for eigenGlass Training Set

For each n-dimensional training vector **x** a new training vector is constructed by concatenating the *MF* parameter to the existing vector producing and *n+1* dimensional vector **x'** = *(***x,** *MF)* = (x1,y1, x2, y2, ..., *MF*).

After PCA has been performed on the training set the resulting PDM can be used to reconstruct new drinking vessels of various shapes along with a corresponding *MF* value. Figure 9.2.2 shows the primary mode of variation of the eigenGlass model from the mean shape along with the corresponding *MF* value.

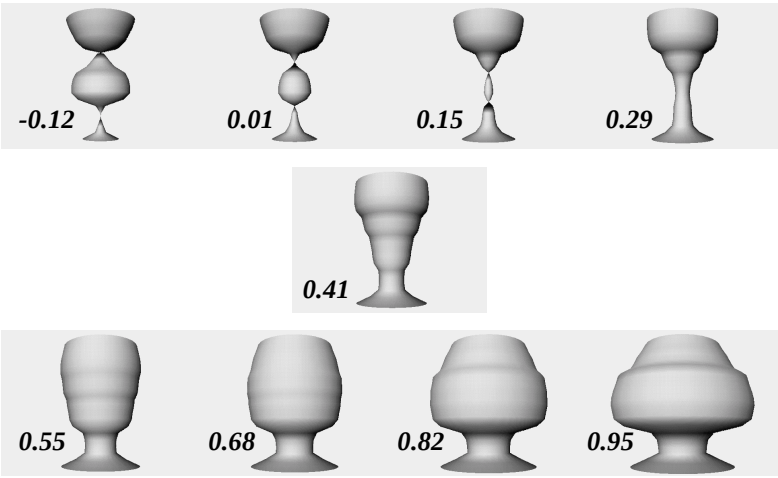


Figure 9.2.2 - Primary mode of variation of Augmented eigenGlass PDM

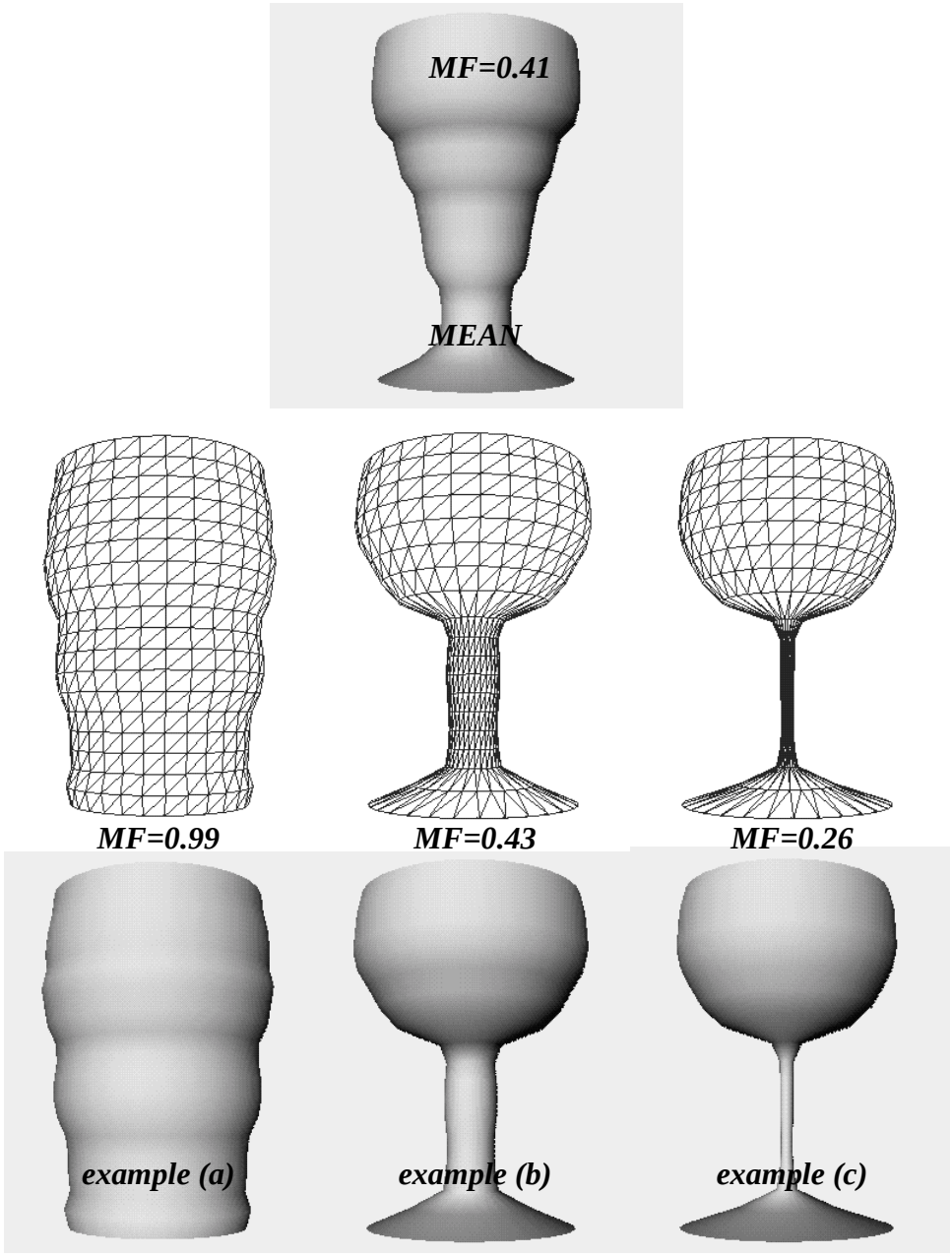


Figure 9.2.3 - Reconstructed glasses and MF value from Augmented eigenGlass PDM

Figure 9.2.3 shows the results of reconstructing various glass types from the eigenGlass model along with the corresponding *MF* value. This is achieved by manipulating the weighting parameters of the model. As the overall shape changes, so the additional *MF* parameter changes accordingly. It can be seen that the pint glass produces a high *MF* value which corresponds to the training set. Similarly the wine glass example c) produces a low *MF* value, demonstrating that the PDM has successful achieved some mapping between the elements. As

with all PDMs, the ability of the final model to reproduce examples from the training set is augmented by the ability to generalise the shape information and produce *unseen* shapes, not present within the training set. When this is done an *MF* value is also produced and by observing this parameter it is possible to draw some conclusion about what the PDM has encoded.

The model demonstrates a high correlation between the size (volume) of the glass and the *MF* value. This is to be expected, as the high *MF* examples were the larger types of glass. However, example (b) shows the results of attempting to make a 'more' masculine wine glass and results in a thicker stem. So it could also be concluded that the more delicate the stem of a glass the more feminine its appearance. This would seem a fair assumption given that in the training examples the two extremities of *MF* were a pint Beer glass and a Champagne glass where the major difference between the examples was the stem thickness (see Figure 9.2.1).

This is an extremely subjective example but demonstrates how additional information can be incorporated into a PDM.

9.2.2 Scaling Issues and Eigen Entrophy

One of the important issues when elements are to be combined for statistical analysis is that of scaling. If an element contains too much variation across the training set (due to the incorrect scaling of that component) then that element will bias the PCA and dominate the principal modes of variation. In some cases this may be desirable, e.g. when it is intended that the primary mode correlates directly to the variation of a specific feature. However, more often, this is an undesirable effect.

The premise of the PDM is that the largest variation of the training set should be represented within the eigenvector corresponding to the largest eigenvalue. By artificially biasing the PCA with an incorrect scaling the information content of the PDM is destroyed.

If the eigenGlass model is considered, the construction of the training vector should contain a scaling parameter α where

Equation 9.2-1 $\mathbf{x}' = (\mathbf{x}, \alpha MF) = (x_1, y_1, x_2, y_2, \ldots \alpha MF)$.

When the training set is assembled this additional parameter can be scaled appropriately to ensure incorrect dominance does not occur. However, it is generally not apparent what this scaling value α should be for any particular example.

Sumpter, Boyle and Tillett [Sumpter 97] proposed a method for estimating the scaling of parameters by calculating the eigen entropy (*E*) of the normalised eigen vectors *(p)*, and estimating the value α which maximises this entropy $E(\alpha)$,

Equation 9.2-2 $E = E(\alpha) = -\sum_{i=1}^{n+1} p_i \log_2(p_i)$, where

Equation 9.2-3 $p_i = \dfrac{\lambda_i}{\sum_{i=1}^{n+1} \lambda_i}$, $E(\alpha) \leq \log_2(n+1)$,

$$\text{and } E(\alpha) \to 0 \text{ as } \alpha \to \infty$$

Figure 9.2.4 shows the results of performing this procedure upon the eigen glass example. From this graph it can be seen that the optimum eigen entropy is achieved with a scaling of around $\alpha = 137$.

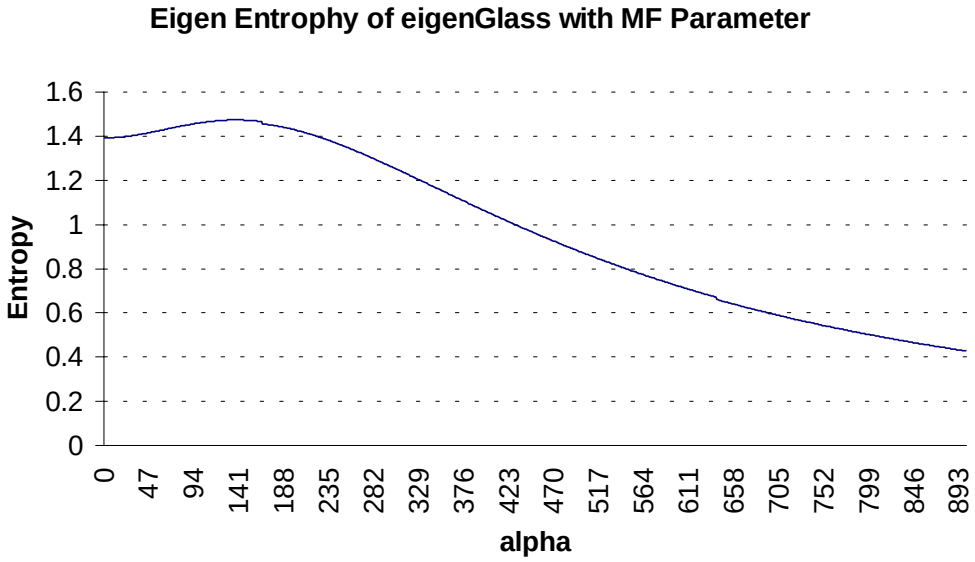


Figure 9.2.4 - Graph of eigen entropy for varying parameter scaling

If PCA is now repeated upon the eigenGlass example with $\alpha = 137$, a new PDM is constructed. It is hoped that the *MF* parameter presents increased significance within the primary modes of variation. This hypothesis can be confirmed by examining the eigenvalues and the variance of *MF* for models constructed with $\alpha = 1$ and 137.

Figure 9.2.5 shows the histogram of normalised eigenvalues in percentile form (see chapter 3.2) for the eigenGlass example with the two aforementioned MF scalings. As would be expected the addition of this parameter and its increased significance within the primary modes (for $\alpha = 137$) has removed some of the information content from the primary modes of variation, with a small increase in the significance of the latter modes. However, the resulting model still retains 99% of the variance within the first four modes so the information content is preserved, unlike $\alpha \to \infty$ which results in only a single mode of variation (due to the dominance of *MF* over the PCA), destroying the information content of the model.

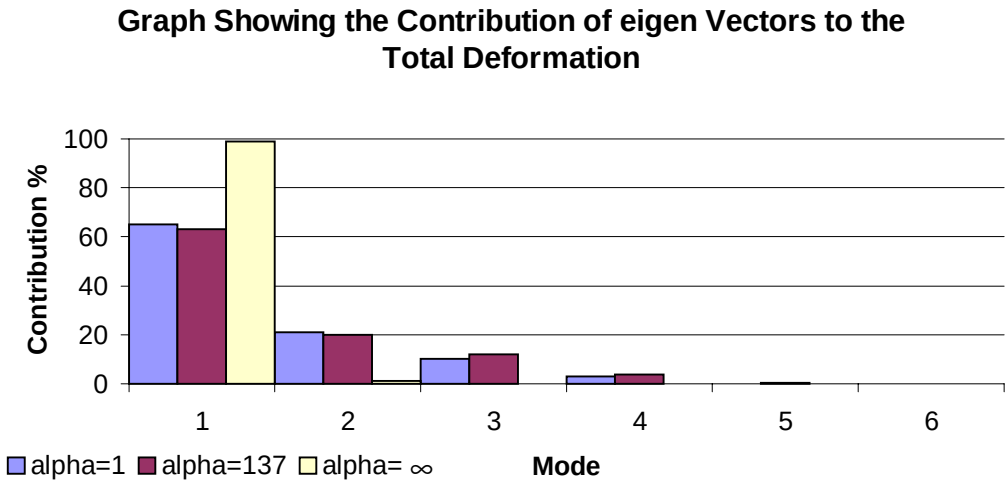


Figure 9.2.5 - Graph demonstrating the normalised eigen values for the eigenGlass example with different parameter scaling

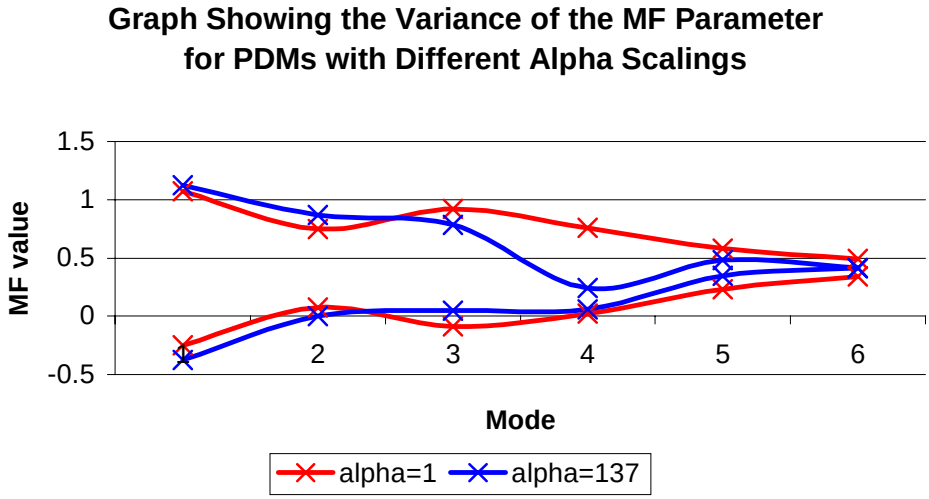


Figure 9.2.6 - Graph demonstrating the increased variance in eigenGlass example for correct parameter scaling

Figure 9.2.6 demonstrates this increase in the variance of the *MF* parameter by plotting the bounds of the variance for each of the primary modes of the two PDMs. Both variances are based around the mean *MF* and regress to this mean as the contribution of a mode diminishes. It can be seen for $\alpha = 137$ that the variance of *MF* is increased within the first two modes, with a significant reduction of this variance in the latter modes. This demonstrates that the

increased scaling has forced the statistical correlation into the primary three modes while more evenly distributing the overall variance of the model.

From Figure 9.2.5 and Figure 9.2.6 it can be concluded that the parameter scaling increases the correlation between shape and parameter without destroying the information content of the resulting PDM. However, another important consideration is 'how has this affected the primary modes of shape deformation?'. This can be answered by comparing the deformation of the original eigenGlass PDM to this new weighted model.

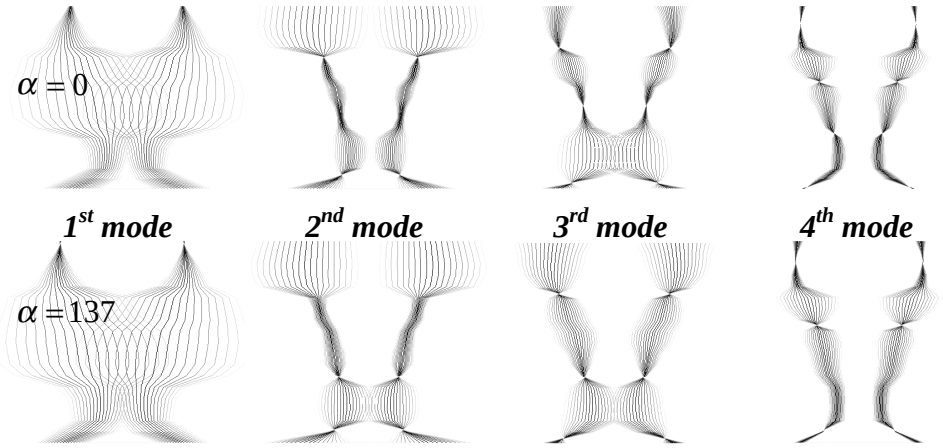


Figure 9.2.7 - Primary modes of eigenGlass PDM with different alpha scalings

It can be seen from Figure 9.2.7 that the increased significance of the *MF* parameter has done little to effect the overall deformation of the eigenGlass shape. It has increased the shape deformation to accommodate the *MF* parameter which shows that, although a correlation is being achieved, it is not a simple linear correlation. This could be addressed by using a non-linear model as previously developed, this will be discussed in more detail in Section 9.3.

9.2.3 Statistical Inference

It has been shown how additional information can be incorporated into a PDM which does not necessarily have to lie within the same co-ordinate frame as the shape deformation. It has also been shown how this information can be statistically linked to the other features of the model. When a shape is

reconstructed, so the additional parameters of the model are estimated due to the statistical linkage that occurs between the elements during PCA. However, what is desirable is to be able to use this model to estimate the parameters for unseen objects or even predict shapes that correspond to specific parameter values.

Using the matrix form of a linear PDM the shape **x** of a model is equal to the mean shape plus the weighted sum of the eigenvectors

Equation 9.2-4 $$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}$$

where **x** is the shape vector, $\bar{\mathbf{x}}$ is the mean shape, $\mathbf{P} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ is a matrix of the first *t* eigenvectors and $\mathbf{b} = (b_1, b_2, \ldots, b_t)^T$ is a vector of weights.

Given a new shape $\mathbf{x}'$, the closest allowable shape from the model is constructed by finding **b** such that

Equation 9.2-5 $$\mathbf{b} = \mathbf{P}^{-1}\left(\mathbf{x}' - \bar{\mathbf{x}}\right) \text{ and } -3\sqrt{\lambda_i} \le b_i \le 3\sqrt{\lambda_i}$$

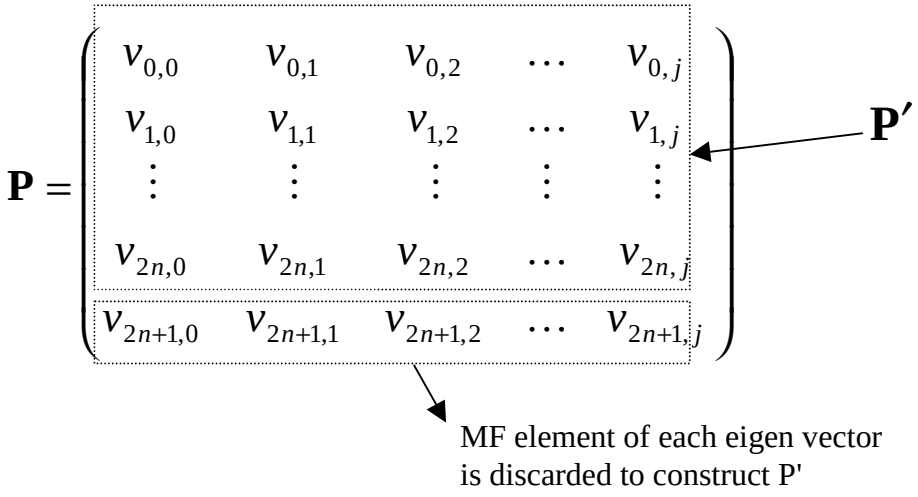
The closest allowable shape can then be reconstructed as

Equation 9.2-6 $$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}$$

If the eigenGlass example is now considered, it is feasible that given a new 'unseen' glass example ($\mathbf{x}'$) the PDM could be used to estimate a value for *MF*. As the PDM has encoded a statistical link between the shape and parameter this model can be used to predict this estimate. However, the two elements have different dimensionality. The unseen example has dimensionality of *2n*, where the PDM has a dimensionality of *2n+1*. The new example $\mathbf{x}'$ could be converted to a *2n+1* vector by the addition of a zero, and the vector then reconstrcuted using the procedure above. However, in finding the closest allowable shape from the PDM, weighting parameters would be extracted that best fit the shape and provide an *MF* of zero. For non-linear mappings where the correlation between these elements is complex and the linear formulation of the PDM is over

generalising, this mapping will lead to unreliable results, i.e. **the zero parameter will bias the reconstruction**. As the number of unknown parameters increases this zero bias will begin to dominate the reconstruction and the resulting reconstructed vector will begin to degrade. Instead the model must be reduced to the dimensionality of the vector.

This is achieved by taking the matrix **P** which is a $2n+1\times j$ matrix of eigen vectors and extracting a smaller matrix **P'** which is a $2n\times j$ matrix.



This is done by discarding the elements of each eigen vector which correspond to the unknown elements of the model (in this case the *MF* parameter). A similar procedure must be performed on the mean shape $\bar{\mathbf{x}} \in \Re^{2n+1}$ by discarding the unknown parameter to obtain $\bar{\mathbf{x}}' \in \Re^{2n}$. The weightings which produce the shape can then be calculated in a similar manner with the reduced dimensional model, where

Equation 9.2-7 $$\mathbf{b}' = \mathbf{P}'^{-1}\left(\mathbf{x}' - \bar{\mathbf{x}}'\right) \text{ and } -3\sqrt{\lambda_i} \le b_i \le 3\sqrt{\lambda_i}$$

However, as only the dimensionality of the eigen vectors was changed and not the number of eigen vectors, **b'** has the same dimensionality as $\mathbf{b} \in \Re^{j}$. The weighting vector **b'** can therefore be placed directly into Equation 9.2-4 to reproduce the shape **x**, by

Equation 9.2-8 $$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}'$$

The closest allowable shape vector $\mathbf{x} \in \Re^{2n+1}$ to $\mathbf{x}' \in \Re^{2n}$ has now been reconstructed. However, the additional information in $\mathbf{x}'$, contains the missing *MF* information which has been estimated from the available shape information and the *a priori* information contained within the model of about shape and how this relates to the *MF* parameter.

9.3 Extending the Model to Inferring Human Motion

9.3.1 Introduction

The human vision system is adept at recognising the position and pose of an object, even when presented with a monoscopic view. In situations with low lighting conditions in which only a silhouette is visible, it is still possible for a human to deduce the pose of an object. This is through structural knowledge of the human body and its articulation.

A similar internal model can be constructed mathematically which represents a human body and the possible ways in which it can deform. This is the premise of model based vision, and as has been previously shown, this deformation can be learnt using a Point Distribution Model. By introducing additional information to the PDM that relates to the anatomical structure of the body, a direct mapping between skeletal structure and projected shape can be achieved.

This section uses the previously presented techniques to statistically combine the 2D silhouette of a human body projected onto the image frame with the 3D pose of the body. To further aid the tracking and reconstruction process, additional information about the location of both the head and hands is combined into the model. This helps disambiguate the model and provides useful information for both its initialisation and tracking.

9.3.2 Constructing a Combined Non-linear Point Distribution Model for a Human

The point distribution model is constructed from three components: the position of the head and hands within the image frame; the 2D contour which represents the shape of the body silhouette; and the 3D structure of the body (see Figure 9.3.1). Each of these components are generated separately from the training image sequence and then concatenated to provide a training vector representing all these attributes.

The relative position of the head and hands is represented as the location of these features in the image frame. When concatenated, this generates a six dimensional feature vector $V_H=(x_1,y_1,...x_3,y_3)$. The body contour, once extracted from the image, is resampled to a list of 400 connected points. These are concatenated into an 800 dimensional feature vector $V_C=(x_1,y_1,...x_{400},y_{400})$. Lastly the skeletal structure of the 3D model is represented by 10 3D points which produce a 30 dimensional feature vector V_S. The relative location of the hands and head helps to disambiguate the contour during tracking. It can also be used to estimate an initial location and shape for the body contour.

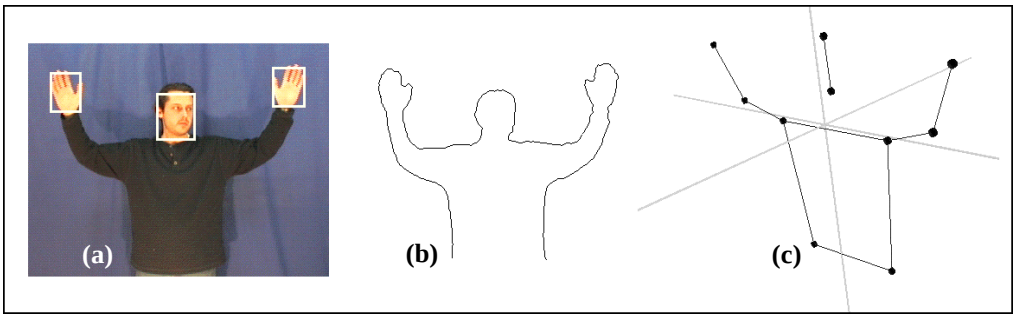


Figure 9.3.1 Composite elements of human body PDM

(a) Position of head and hands V_H (b) Body Contour V_C
(c) Corresponding 3D model V_S

The position of the head and hands is extracted from the training image sequences using the Hue-Saturation colour thresholding technique described in Chapter 4.

For the purpose of simple contour extraction from the training set, shape extraction is facilitated through the use of a blue screen and chroma keying. This allows the background to be simply keyed out to produce a binary image of the body silhouette. As the figure always intersects the base of the image at the torso, an initial contour point is easily located. Once found, this is used as the starting point for a contour tracing algorithm which follows the external boundary of the silhouette and stores this contour as a list of connected points. In order to perform any statistical analysis on the contour, it must first be resampled to a fixed length. To ensure some consistency throughout the training set, landmark points are set at the beginning and end of the contour. A further landmark point is allocated at the highest point along the contour within 10 degrees of a vertical line drawn from the centroid of the shape. Two further points are positioned at the leftmost and rightmost points of the contour. This simple landmark point identification results in non-linearity within the model. The problems associated with this are discussed in Section 9.3.5.

The 3D skeletal structure of the human is generated manually. Co-ordinates in the xy (image) plane are derived directly from the image sequence by hand labelling. The position in the third dimension is then estimated for each key frame.

9.3.3 Scaling the Model

When combining information for statistical analysis via PCA it is important that constituent features (V_H V_C V_S) are scaled to ensure that any particular feature does not dominate the principal axes. This can be done by calculating the eigen entropy as discussed earlier (section 9.2.2). However, as all three components exist within the same co-ordinate frame and are directly linked, such a scaling should be unnecessary.

This assumption can easily be tested by formulating the vector **x** as the weighted combination of the components where $\mathbf{x} = (V_C, \alpha V_H, \beta V_S)$. Using the same procedure as described earlier, the eigen entropy is calculated for

$0 < \alpha, \beta < \infty$ and suitable scaling values determined by maximising the entropy of the resulting PDM.

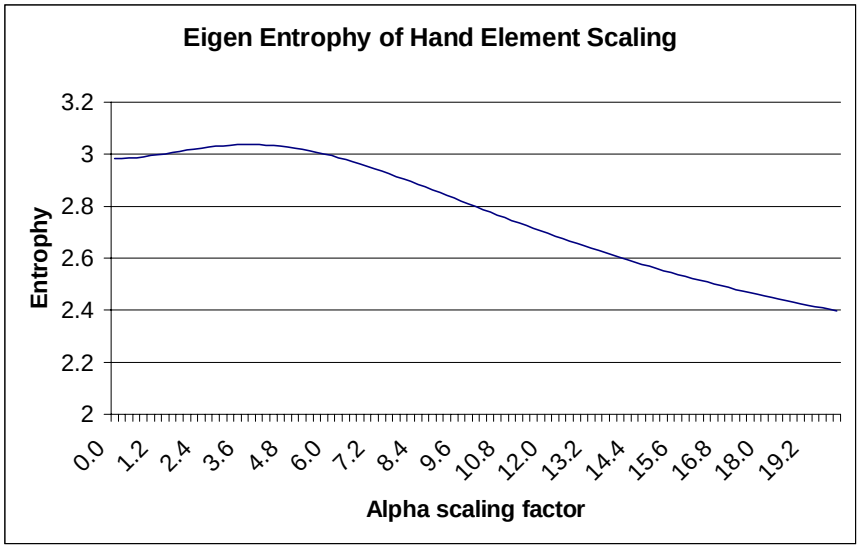


Figure 9.3.2 - Graph showing eigen entropy of hand element in composite body PDM

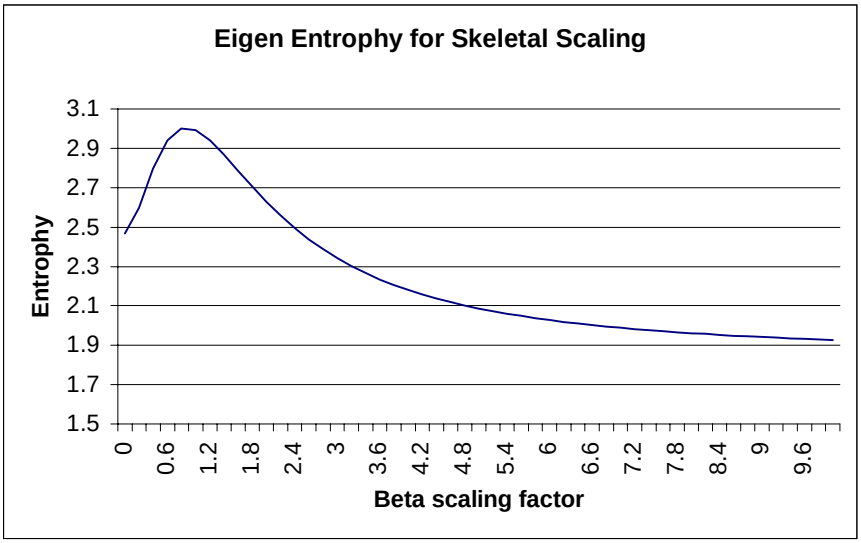


Figure 9.3.3 - Graph showing eigen entropy of skeletal element in composite body PDM

From Figure 9.3.2 it can be seen that the optimum scaling for V_H is around 4. Figure 9.3.3 shows that the skeletal element does not need scaling as the greatest

entropy is achieved when $\beta = 1$. This confirms the assumption that scaling is unnecessary as all the elements lie within the same (image) co-ordinate frame.

9.3.4 The Linear PDM

Once these separate feature vectors are assembled, they are concatenated to form an 836 dimensional vector which represents the total pose of the model. A training set of these vectors is assembled which represents the likely movement of the model. Figure 9.3.4 shows a sample of training images along with the corresponding contour and skeletal models in 2D.

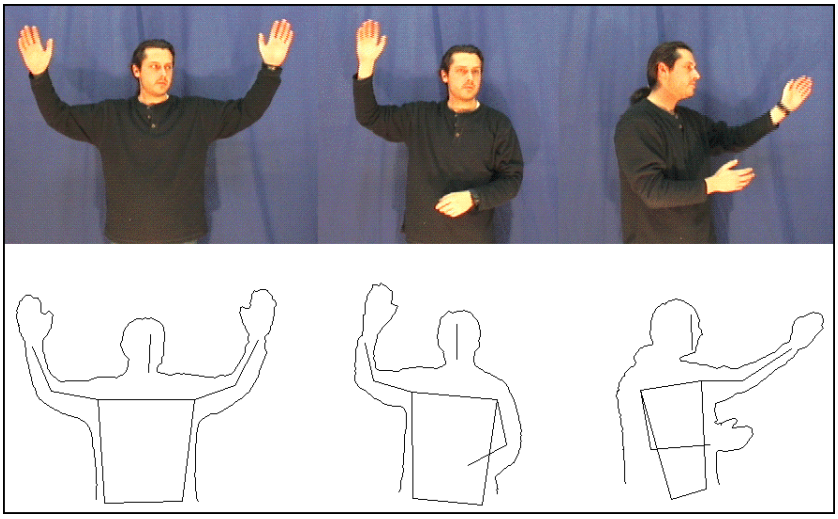


Figure 9.3.4 - Sample training images and corresponding contour and skeletal models

A linear PDM is now constructed from the training set and its primary modes of variation are shown in Figure 9.3.5.

After PCA is performed, it is calculated that the first 84 eigenvectors, which correspond to the 84 largest eigenvalues, encompass 99.99% of the deformation contained in the training set.

Figure 9.3.5 demonstrates the deformation of the composite PDM. The crosses are the locations of the hands and head. It can be seen that although the movement of the three elements are closely related, the model does not

accurately represent the natural deformation of the body. The shapes generated by the primary modes of variation are not indicative of the training set due to its inherent non-linearity. In order to produce a model that is accurate/robust enough for practical applications, a more constrained representation is required.

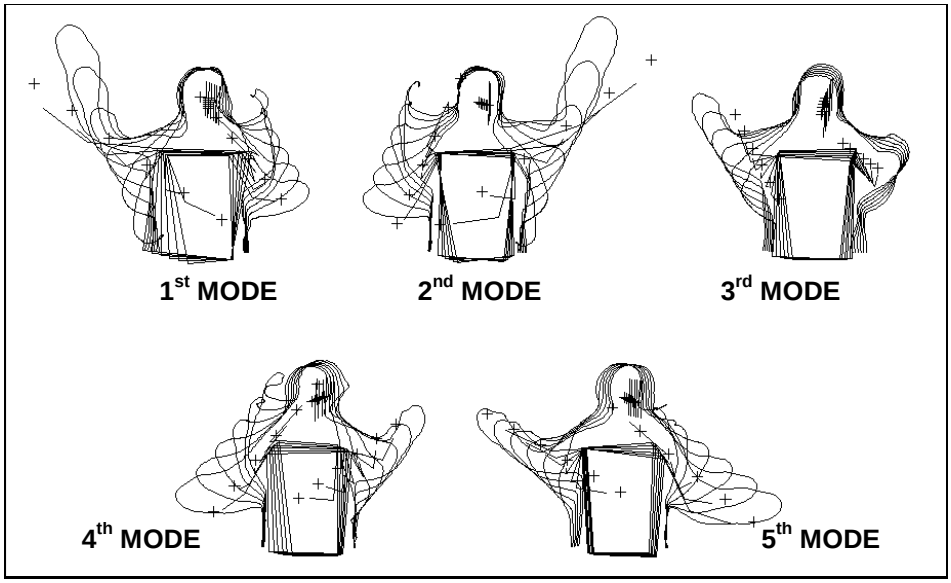


Figure 9.3.5 - Primary modes of variation on the linear PDM

9.3.5 Non-Linear Estimation

As described in chapter 6, to perform non-linear estimation upon the dataset the linear model is first used to reduce the dimensionality. 99.99% of the deformation is contained within the first 84 eigenvectors. However, the primary 40 modes of deformation encompass 99.8% of the deformation. Projecting the entire training set down into this lower dimensional space achieves a dimensional reduction of 836 to 40, which significantly reduces the computation time required for further analysis.

Performing cluster analysis upon the dimensionally-reduced dataset, the natural number of clusters is estimated to be 25. By performing further PCA on each of the 25 clusters, the shape of the model can be constrained by restricting the shape vector to remain within this volume. These constraints upon *shape space* are applied in the same manner as described in earlier chapters.

Figure 9.3.6 shows the training set after dimensional reduction gained from the initial linear PDM, projected into 2 dimensions. The bounding boxes represent the 25 clusters that best estimate the curvature. These bounding boxes are the bounds of the first and second modes of deformation for each linear patch (cluster). The number of modes for each cluster varies according to the complexity of the training set at that point within the space. All clusters are constructed to encompass 99.9% of the deformation within that cluster.

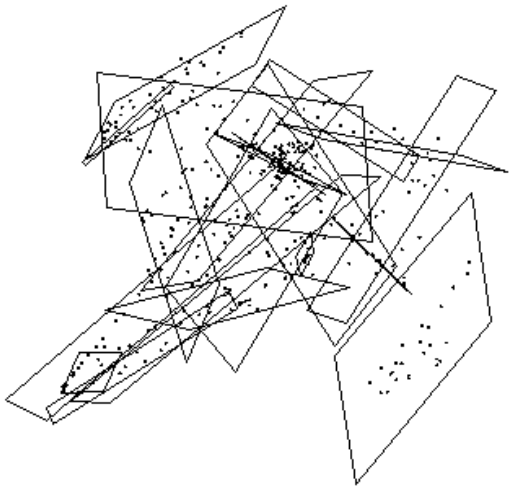


Figure 9.3.6 - Clusters in reduced shape space

9.3.6 Initialising the PDM

Upon initialisation the first step is to locate the position of the head and hands. This can be done by colour thresholding the entire image which, although computationally expensive, does not need to be repeated on every iteration. Once done these positions can be used to initialise the PDM and give an initial guess as to the shape of the contour to be found. As is it not clear which blobs correspond to which features, three possible contours are produced. The contour that iterates to the best solution provides the final state from which tracking proceeds.

9.3.7 Tracking with the PDM

Once initialised the two components must be fitted to the image separately. The contour is attracted to high intensity gradients within the image using local edge detection (chapter 3). The hand and head positions are used as centres in a single iteration of a kmeans-clustering algorithm on the segmented binary skin image. This is possible due to the assumption that the model will not change significantly from the last image frame.

9.3.8 Reconstruction of 3D Shape and Pose

As the shape deforms to fit with the image so the third element of the model, the skeleton, also deforms. By plotting this 3D skeleton, its movements mimic the motion of the human in the image frame.

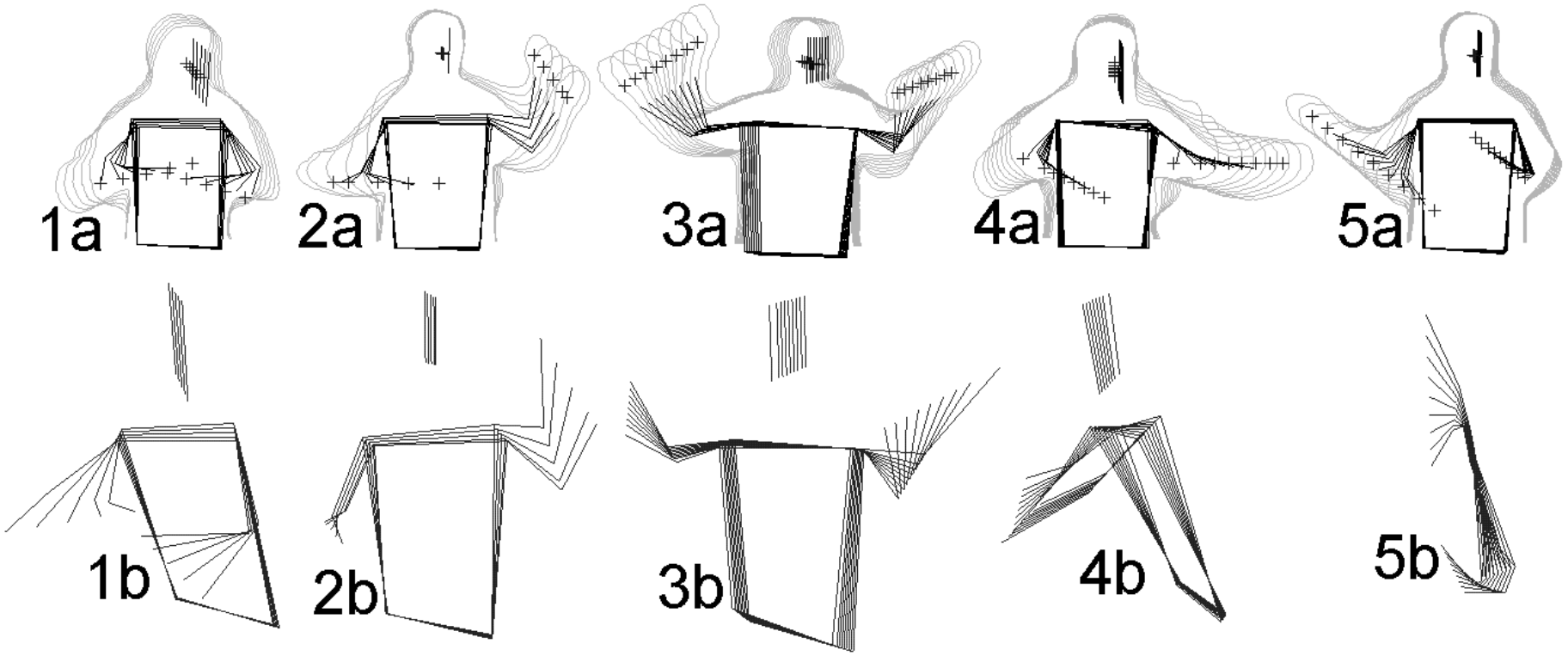


Figure 9.3.7 - How the Model Deforms

Figure 10.3.7 demonstrates the correspondence between the body contour and skeletal structure. Each contour image (a) is generated from a different sub cluster of shape space. The deformation corresponds to the largest mode of deformation for that cluster. The 3D skeletal diagrams (b) correspond to the relevant contour (a), and demonstrate the movement of the skeleton. The orientation of these skeletal models has been changed in order to better visualise the movement in 3D. Skeleton (1b) demonstrates the arms moving in the z direction corresponding to the change in contour (1a) around the elbow region. Contour (4a) represents a body leant toward the camera with moving arms. Skeleton 4b shows the corresponding change in the skeleton with the shoulders twisting as the arms move. The Skeleton 5b is a plan view showing the movement of the hands.

All model points move along straight lines due to the linear clusters used to approximate the non-linear shape space. However, all poses of the models are lifelike human silhouettes, demonstrating the CSSPDM's ability at modelling the non-linearity.

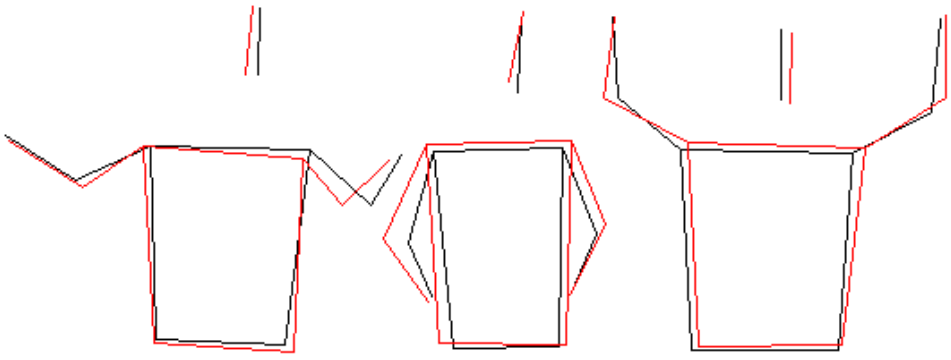


Figure 9.3.8– Reconstructed poses from the model

Figure 9.3.8 shows the original model pose from the training set in red with the reconstructed skeletal model in black. It can be seen that the original and reconstructed models are similar in pose and position with the length of limbs preserved, further demonstrating the absence of non-linear effects. However, as the constraints on shape space are increased, so the performance degrades. Inconsistencies in the original and reconstructed models and the deterioration

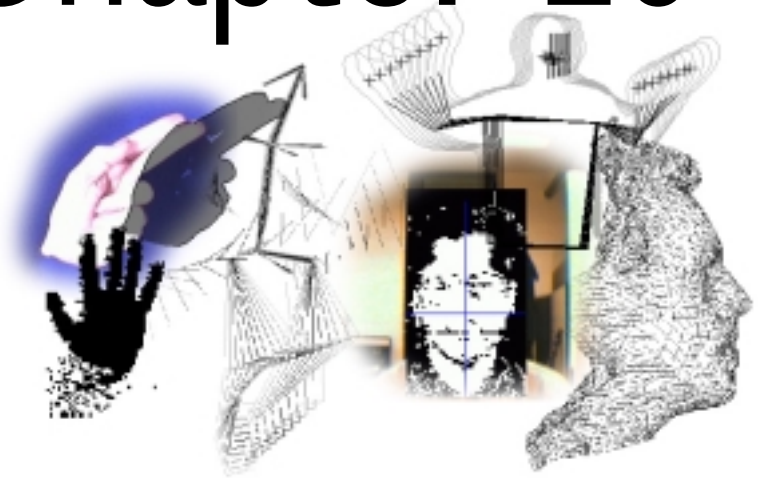
under heavy constraints can be attributed to the hand labelling of the training set. During hand labelling it is impossible to provide consistent models of the skeletal structure throughout the training set. This factor leads to the final model producing mean skeletal ‘smoothed’ shapes which have been ‘learnt’ from the original training set and hence produces the inconsistencies observed in figure 1.3.8.

9.4 Conclusion

This section has shown how information can be statistically linked through PCA to produce point distribution models which contain multiple perspectives of data. These perspectives do not have to lie in the same co-ordinate frame and may be related but abstract in nature. By concatenating features, ensuring that incorrect biases do not occur, models can be constructed which not only learn about shape and deformation and how this relates to other aspects of an object, but also to predict these aspects or other missing information from that which is available.

It has been shown how these techniques for statistical inference can be applied to the extraction of 3D structure of an object, given only a monoscopic view of its outline. The technique uses computationally inexpensive techniques for real time tracking and reconstruction of objects. It has also been shown how two sources of information can be combined to provide a direct mapping between them. Being able to reconstruct 3D pose of a human from a simple contour has applications in surveillance, virtual reality and smart room technology and could possibly provide an inexpensive solution to more complex motion capture modalities such as electromagnetic sensors and marker based vision systems.

Chapter 10



10 Closing Discussion

10.1 Summary

This thesis has attempted to address the problems associated with the construction and application of deformable contour models for real-time tracking and interpretation of scenes. Deformable models were chosen as a research subject due to their power and speed at segmenting objects under normal environmental conditions where few constraints can be placed upon applications to simplify segmentation. By taking deformable models as a starting point, the work has attempted to push current approaches into new domains where existing techniques would fail. In doing so, a fundamental understanding of the associated problems has been gained and these problems addressed.

After reviewing related literature in Chapter 2, Chapter 3 introduced linear Point Distribution Models and discussed their construction and use in object tracking. It was shown that one of the most important aspects of the PDM is the inherent dimensional reduction of the model.

Chapter 4 discussed the use of colour in object tracking and demonstrated how simple colour techniques could be used to enhance object segmentation. This

chapter also demonstrated how object colour could be used in its own right as a powerful feature for tracking.

In Chapter 5 non-linear datasets were introduced and their effects upon linear PDMs discussed. The Cluster Based non-linear PDM (CBNLPDM) was introduced which modelled non-linearity by breaking a dataset down into a piecewise linear approximation to the non-linear data set. It was shown how models could be constructed which better represented non-linearity while retaining the simplicity and speed of the linear PDM. It was also shown that the technique produced superior performance for model representation than other related approaches.

Chapter 6 extended this work and introduced a vital adaptation to the CBNLPDM. By projecting the training set down into a lower dimensional space before non-linear analysis, large computational savings could be made. This approach called Constrained Shape Space PDMs (CSSPDM) allows non-linear analysis to be performed on high dimensional data such as images or 3D structures. It was also shown that the data smoothing effect of this dimensional reduction produces advantages for both model building and reconstructive accuracy. Furthermore the natural segregation of the CSSPDM combined with the low dimensionality provides a mechanism for the static pose recognition of objects. This was demonstrated by using a CSSPDM of the hand to classify letters from the American Sign Language finger spelt alphabet.

In Chapter 7 the important consideration of how objects move with time was introduced. It was shown that this natural segmentation of shape space could be used for discrete time dependent analysis by augmenting the CSSPDM with a markov chain. This was illustrated with 3D motion capture data, where not only the deformation of the model was learnt, but also the motion contained within the training set. Using this motion model plausible mean trajectories of human motion were reproduced which were learnt from recorded motion data and visualised graphically. The temporal CSSPDM was then applied to object tracking and it was demonstrated how it could be used in a simplified CONDENSATION algorithm, which outperformed standard ASM tracking. It

was also shown how the PDF used in the Markov chain could be constructed from sources other than the training data, providing superior results. This is especially important in applications such as gesture recognition where it is not feasible to learn this information by example.

In Chapter 8 the extension of Point Distribution Models to the 3D domain was discussed. Techniques for the construction and alignment of such models were presented and results shown for the automatic construction of large 3D eigen models of the human head.

Finally Chapter 9 took many of the techniques and approaches discussed in this work and applied them to the subject of markerless human motion capture. By linking elements together before PCA is performed, a statistical linkage is achieved which allows unseen information to be inferred from available visual queues. This was demonstrated by tracking a human body in a monoscopic image sequence and extracting a corresponding 3D skeletal model which mimicked the motion of the human.

In order to extend the Point Distribution Models to more complex applications it was necessary to address the problems associated with automated model construction. Namely, the complexities that automated procedures introduce to training sets. Unlike many earlier authors who tackled this problem by trying to attempt to devise complex techniques which would minimise these non-linear effects. This work has tackled the problem by attempting to produce models which can cope with these complexities. In doing so, the resulting developed models have become more reliable and accurate while retaining the simplicity and speed of the original formulation. These accurate, fast non-linear models not only produce superior results, but also allow automated models to be constructed which can have any dimensionality or complexity with almost no user intervention.

10.2 Future Work

This thesis has attempted to address the problems associated with the construction of deformable models. In doing so, it has established a set of generic tools and techniques for the construction and application of complex non-linear models of deformation. By addressing the problems of non-linearity, the approaches provide a solution which, has few constraints upon model assembly and hence opens the application base of the work.

Future work is therefore varied and current work is concerned with further developing the construction and application of models with computer vision and graphics.

Current work into the colour distribution of objects and scenes is extending the work of Chapter 4 to provide an accurate method of locating human motion within complex environments. This work will incorporate models of deformation to address the applications of visual surveillance and monitoring.

The work of Chapter 7 is supporting research into two areas, namely computer animation and gesture recognition. In the field of animation the ability to be able to model the motion of complex surfaces in lower dimensional spaces allows smooth key-frame animations to be achieved. It is also intended that these techniques could be combined with the work in Section 9.2 to allow the abstract parameterisation of human motion in simulation. To fully investigate the applications to gesture recognition, a two handed system must be constructed which allows temporal gestures to be both tracked and classified.

A new model of human motion is currently being constructed, extending the work of chapter 9. This model consists of a tri-camera view of the human subject with the corresponding optical motion capture ground truth. This new model will provide the means to assess the accuracy of the inferred human structure and investigate the associated accuracy of mono, stereo and tri camera reconstruction.

It is also intended that the work described in Appendix 2 on volumetric segmentation be combined with that of the construction of 3D PDMs for medical analysis and diagnosis. In doing so the shape and size of internal organs can be compared with a statistical model to gain an indication of variation from the population mean. It is intended to investigate the use of such approaches in the diagnosis of medical conditions such as hydrocephalus.

Appendix A – K-means and Fuzzy K-means Clustering

11.1 K-Means Clustering

Clustering algorithms attempts to segregate a dataset into distinct regions of membership, this is widely performed by a gradient descent based iterative algorithm that is known as k-means (or c-means) algorithm or the Generalised Lloyd algorithm {Karayiannis 95]. The k-means algorithm begins with a set of *k* initial exemplars, where the data is to be segregated into *k* distinct regions. Each region is evaluated with the exemplar as the centeroid of the region. Data points are assigned to the exemplar in a nearest neighbour fashion and the exemplars moved to minimise the distance between the exemplar and its members. This membership is reassessed at each iteration and repeated until the algorithm converges upon a solution i.e. the movement of the exemplars approaches zero.

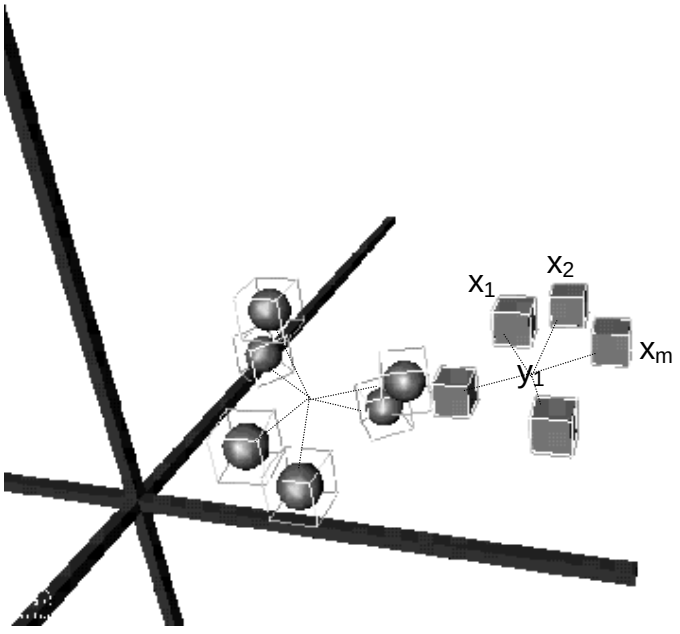


Figure 11.1.1 - K-means clustering

For the clustering of a training set $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$ where $\mathbf{x}_i \in \Re^n$ is an *n* dimensional vector in Euclidean space and $i = 1,2,\ldots,M$. The segregation of the training set into *k* clusters using the exemplars (cluster centres)

$Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$ where $\mathbf{y}_j \in \Re^n$ and $j = 1,2,\ldots,k$ is performed by minimising the cost function *D* where,

$$D = \frac{1}{M}\sum_{i=1}^{M} d_{\min}(\mathbf{x}_i) = \frac{1}{M}\sum_{i=1}^{M} \min_{\mathbf{y}_j \in y}\left(d(\mathbf{x}_i, \mathbf{y}_j)\right)$$

The K-means algorithm assigns each training vector to a certain cluster on the basis of the nearest neighbour condition. According to this strategy, the training vector $\mathbf{x}_i$ is assigned to the j^{th} cluster if $d(\mathbf{x}_i, \mathbf{y}_j) = d_{\min}(\mathbf{x}_i) = \min_{\mathbf{y}_j \in Y} d(\mathbf{x}_i, \mathbf{y}_j)$, where $d(\mathbf{x}_i, \mathbf{y}_j)$ is the squared Euclidean distance between the training vector $\mathbf{x}_i$ and the exemplar $\mathbf{y}_j$, defined as $d(\mathbf{x}_i, \mathbf{y}_j) = \left\|\mathbf{x}_i, \mathbf{y}_j\right\|^2$ [Karayiannis 95].

The nearest neighbour description can be described by the membership function *u*,

$$u_j(\mathbf{x}_i) = \begin{cases} 1 & \text{if} \quad d(\mathbf{x}_i, \mathbf{y}_j) = d_{\min}(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

The algorithm minimises this cost function D through the iterative refinement of cluster centres where the exemplar $\mathbf{y}_j$ is the mean of the vectors assigned to it,

$$\mathbf{y}_j = \frac{\sum_{i=1}^{M} u_j(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{M} u_j(\mathbf{x}_i)} \text{ and } j = 1,2,\ldots,k$$

Although the k-means algorithm is simple and relatively fast to iterate it is a gradient descent method and therefore only capable of finding local energy minima. It will always converge on a low cost solution, but because the energy surface that it traverses is full of local minima, it will not necessarily find the global solution As such, it is extremely sensitive to the initial placement of exemplars. Exemplars are commonly placed randomly within the data space or randomly allocated from the data points themselves. It is therefore necessary to

run the algorithm a number of times with different random initialisations to try and find the best local minima possible.

11.2 Selecting the Natural Number of Clusters k

Often during clustering the natural number of distinct clusters is known. Under these circumstances cluster analysis can be performed using *k*=5. However, more often, little is known about the nature of the data and a method of estimating *k* is required. Furthermore, the nature of the energy minimisation within the k-means algorithm makes the assumption that clusters are hyper-spherical. Where elongated hyper-elliptical clusters are present these may be better modelled using multiple adjoining spherical clusters as demonstrated in chapter 5.2.

The cost function D is commonly used as a metric with which to assess the performance of clustering. As the number of clusters is increased, so the resulting overall cost diminishes in a characteristic way.

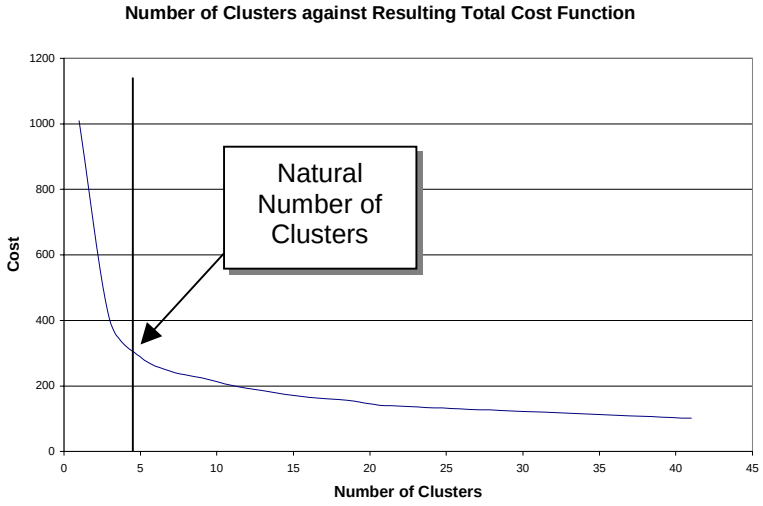


Figure 11.2.1 - Characteristic Cost Graph for k-means for 1 < k < M

Figure 11.2.1 shows the characteristic graph produced for a training set by plotting the resulting overall cost of a converged solution against the number of clusters *k*, where $1<k<M$. The overall cost of a solution decreases as the number of clusters increases, where $k=1$ produces the highest cost and $k=M$ (the number

of training examples) produces a cost of zero. However, as the number of *k* is increased there becomes a point where increasing *k* further does not produce a significant decrease in the resulting cost. This is said to be the natural number of clusters of the data and is a simple but effective method for estimating *k*.

11.3 The Fuzzy K-means Algorithm (FCM)

Fuzzy set theory is a method of representing vagueness in every day life. Bezdeck, Ehrlich and Full proposed a family of fuzzy k-means algorithms [Bezdeck 84]. Fuzzy clustering algorithms consider each cluster as a fuzzy set, while a membership function measures the possibility that each training vector belongs to a cluster. As a result, each training vector may be assigned to multiple clusters with some degree of certainty measured by the membership function. Thus, the partition of the training set is based upon soft decisions [Karayiannis 95].

The fuzzy k-means algorithm uses a fuzzy membership rule where [Bezdeck84]

$$u_j(\mathbf{x}_i) = \frac{1}{\sum_{\ell=1}^{k} \left(\frac{d(\mathbf{x}_i, \mathbf{y}_j)}{d(\mathbf{x}_i, \mathbf{y}_\ell)} \right)^{\frac{1}{m-1}}}$$

The new cluster position $\mathbf{y}_j$ is therefore calculated as

$$\mathbf{y}_j = \frac{\sum_{i=1}^{M} u_j(\mathbf{x}_i)^m \mathbf{x}_i}{\sum_{i=1}^{M} u_j(\mathbf{x}_i)^m} \text{ and } j = 1,2,\ldots,k$$

The "fuzziness" of the clustering produced by these algorithms is controlled by the parameter *m*, which is greater than 1 [Bezdeck84]. As this parameter approaches 1, the partition of the data is nearly the binary decision used in the k-means algorithm. However, as the parameter *m* is increased the membership degrades towards a fuzzy state [Bezdeck84].

Results comparing the partition of space using the k-means algorithm and the FCM algorithm can be found in section 5.2, Figure 5.4.3.

Appendix B – Volumetric Segmentation

12.1 Introduction

The availability and clinical requirements of medical imaging as a source of 3D data set has generated a significant interest in the processing and segmentation of volumetric data. The problems of understanding 3D structure from a discretely sampled volume have shown the benefit of visualisation techniques. Surface approximations, such as isosurfacing, allow surfaces to be extracted that when, rendered and shaded, provide an invaluable insight into a volume's internal structure.

The reconstruction of multi-modal data sets from different sources of volumetric data is greatly simplified by the successful segmentation of surface topology. Surfaces that directly correspond to a volume can be matched far more simply than the original volumes [Moshfeghi 94].

In addition to structural insight, surface approximations are invaluable in reducing the processing time needed for traditional image processing techniques, as processing can be localised to a contour boundary. Furthermore, these surfaces can provide a mathematical representation of shape which can then be used statistically to model and classify shape and deformation [Cootes 95] [Bowden 96].

If a statistical model is to be constructed which represents 3D surfaces or features extracted from medical or other volumetric datasets, a method of extracting surfaces from these datasets is required in order to produce the training examples necessary for statistical analysis.

A common technique for surface extraction is isosurfacing. Isosurfaces are structures that represent surfaces of equal value, normally made out of graphical

primitives such as triangles connected together and rendered using standard graphical techniques.

There are five basic algorithms for Isosurfacing:

1. Opaque cubes or the Cuberille algorithm [Herman 79]
2. Contour connecting [Barequet 96][Fuchs 77][Keppel 75]
3. Marching Cubes [Mullick 95][Cline 88][Lorenson 91]
4. Dividing Cubes [Cline 88]
5. Marching Tetrahedra [Shirley 90].

The Marching Cubes algorithm is by far the most popularly implemented algorithm for iso-intensity surface extraction, efficiently generating isosurfaces with low memory requirements. The Contour Connecting method requires localisation of the contour in each slice of the data and, like the Cuberille algorithm, is prone to artefacts when handling small features and branches in the data. Though the Marching Tetrahedra approach reduces ambiguous topological connections, it generates many more graphical primitives than the Marching Cubes algorithm. Finally, the Dividing Cubes algorithm creates points and corresponding normals requiring special purpose hardware/software for visualisation, making it inappropriate for many applications [Mullick 95].

Barequet *et al* [Barequet 96] propose a technique for piecewise-linear surface reconstruction from a series of parallel polygonal cross sections. As well as the applications of such algorithms in visualisation (isosurfacing) it is an important problem in medical imaging, where contours are often detected in single layers of the volume. By reducing the problem to the piecewise linear interpolation between each pair of successive slices, they use a partial curve matching technique for matching parts of the contours. The major advantage with this over such a scheme as marching cubes is that the size of the resulting polygons compared those produced by marching cudes, where each voxel can produce multiple polygons.

Since the original formulation of Active Contour Models (Snakes) [Kass 88] a significant interest has been shown in extending the technique to dynamic 3D

models. Snakes have been shown to be useful in contour reconstruction, but require large amounts of user intervention to successfully segment complex objects. As has been shown, Point Distribution Models [Cootes 95] can simplify the problem of object recognition and segmentation by statistically constraining the shape of the model within suitable bounds, through the analysis of a training set of shapes. However, in 3D, where models become too large to manufacture by hand, another means of generating training sets for statistical analysis must be found.

Terzopolous and Vasilescu [Terzopoulos 91] extended the snake model to include an inflation force that helps remove the need for initial contour placement and thus avoid convergence on local minima. The inflation force drives the snake model outwards towards the object boundary like an inflating balloon. Terzopolous and Vasilescu formulated the model as a finite element mesh and later extended the model to a thin plate spline, demonstrating successful results in the reconstruction of range data and volumetric CT data surface representations [McInery 93].

This chapter presents an iterative, dynamic mesh model which uses simulated physical forces to segment desired surface approximations from volumetric datasets. The work is based on the work of Chen and Medioni [Chen 95] which is itself a continuation of the work on dynamic balloon models by Terzopoulos and Vasilescu [Terzopoulos 91]. Chen and Medioni applied the work to the constrained problem of reconstruction from pre-registered range images.

It will be shown how simplifications can be made to the model which increases the iterative speed of converging on segmented features. It is also shown how balloon models can be reformulated to remove explicit data attraction forces to image features. The process hence behaves like a region growing technique which locates isointensity boundaries within the image. This removes the need for parameter selection which must be balanced against the internal parameters for standard snake [Kass 88] and balloon [Terzopoulos 91] models and further reduces suseptibility to initial placement and image noise.

The remainder of this chapter is organised as follows, Section 12.2 provides an overview of the dynamic mesh balloon model. Section 12.3 discusses mesh structure and connectivity while Section 12.4 covers dataset scaling and interpolation issues. Section 12.5 then formulates the dynamic mesh structure and subdivision mechanisms. Section 12.6 shows the resulting model applied to sample volumetric data sets. Finally conclusions and further work are discussed.

12.2 Overview of the Dynamic Mesh Model

The mesh structure consists of a triangular mesh which can vary in size, shape and connectivity. Each vertex is connected to other vertices in the model by the edges of the polygonal facets. These interconnections are used to simulate springs that connect the mesh mathematically. The force of these springs gives a resulting surface tension to the model which attempts to keep the surface as smooth as possible. An inflation force is used at each vertex to inflate the overall model, while surface tension attempts to keep the mesh spherical. A simple local feature detection scheme is used at each vertex to remove the inflation force as nodes reach the boundaries of desired structures. A dynamic mesh subdivision scheme is used to subdivide polygons locally if they exceed set size or curvature criteria. This allows the mesh to inflate and grow until a boundary is located. Once the mesh has converged on a solution, a good local edge detection scheme can be used to lock vertex points to the boundary. The process starts with a small polygon object which is inflated from within a volumetric image with the inflation force driving the surface towards the object boundary. The mesh grows in size and complexity to fill the object like an inflating balloon until the mesh vertices lie close to the true object boundary (See Figure 12.2.1). This technique requires no user intervention after the initial placement and provides a simple, fast method for object segmentation, which produces surfaces with a low polygon count.

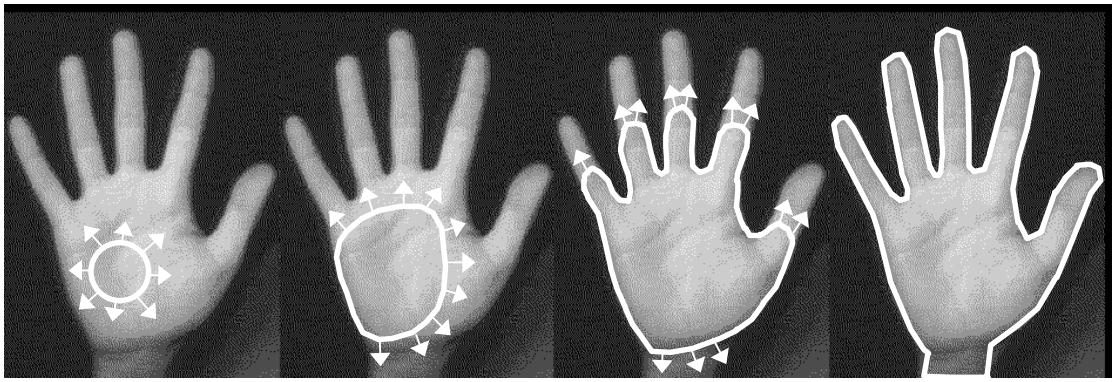


Figure 12.2.1 - Simple 2D Contour Inflating Towards the Object Boundary

12.3 Mesh Structure

To provide the successful extension of the balloon model into 3D, the mesh structure must fulfil a number of criteria:

1. It should allow the dynamic manipulation of a surface and its local properties.
2. It should be structured to ensure render times and processing times are kept to a minimum
3. It should have the ability to represent features accurately by ensuring planar facets and hence reducing mathematical inaccuracies.
4. It must maintain knowledge of its connectivity, to provide a simulated physical model like snakes [Kass 88].
5. It must provide a faithful render of the volume providing accurate visualisation of complex features within a given dataset.

The addition of this final constraint also ensures that the surface will look continuous when rendered with a suitable shading routine such as Gouraud/Phong shading. Perhaps the simplest of mesh structures is that of the simplex mesh, proposed by [Delingette 94]. A simplex mesh is an interconnected set of nodes, where each node is connected to exactly three other nodes (Figure 12.3.1a).

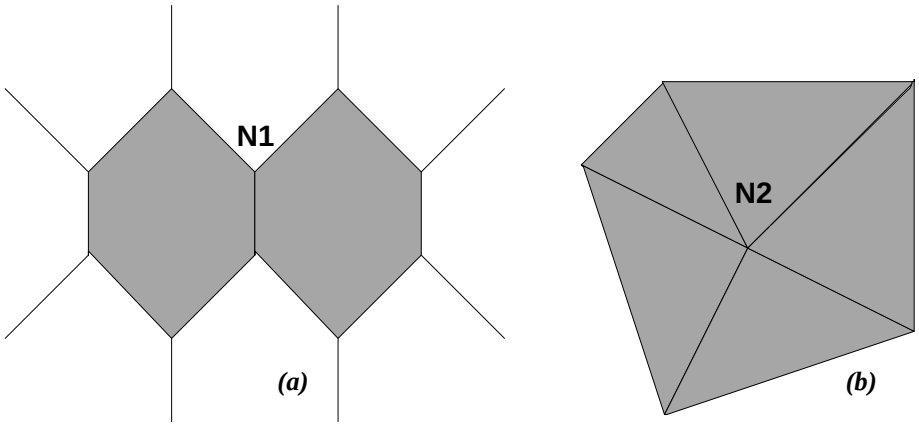


Figure 12.3.1 - Mesh structures

(a) The Simplex Mesh Structure (b) A Planar Mesh Structure

Each node (**N1**) always connects to exactly 3 other nodes producing a simple interconnected surface model from which mathematical simulations of physical properties can easily be implemented. However the polygons bounded by these nodes are non-planar. The calculation of a vertex normal is produced by averaging the normals from the connecting polygons surrounding that vertex. Since non-planar polygons produce inaccurate normal calculations, this mesh formulation will produce inaccuracies in rendering or physical simulation calculations. Inaccuracies in normals will result in non-uniform shading as lighting equations depend upon normals and planar polygons. Surface features will also suffer from the use of non-planar polygons.

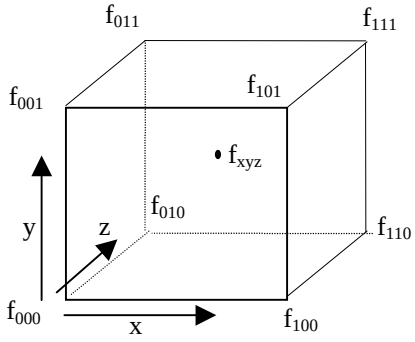
A better solution is to use a mesh that has planar facets. Three points always ensure a unique plane and it is simple to subdivide a triangle into multiple triangles. This does, however introduces problems with the connectivity, as any node must be able to connect to any other number of nodes to ensure a complete and evenly spaced surface. In Figure 12.3.1b the node (**N2**) connects to five other nodes.

This provides a mechanism that represents how each vertex connects to other vertices allowing simple physical properties to be represented, i.e. elasticity can be manifested as the force that each of the connected vertices applies on a

specific vertex by the direction and length of the connections. However other operations such as rendering and normal calculations require polygons to be expressed as the connection of vertices that constitute a surface facet and it is therefore necessary to retain a dual representation of the surface.

12.4 Volume scaling and Interpolation

Volumetric data is commonly stored as a 3D array of discrete values for each voxel (Volumetric Element) of a volume. The resolution of these volumes tends to be far lower than standard images due to the size and memory requirements. A typical 256x256 grey scale image would occupy 64KB of memory, however a 256x256x256 volume using 256 grey levels would occupy 16MB of memory. Due to the low resolution of volumes and non-cubic voxels it is necessary to smoothly interpolate intensities and attempt to estimate missing information. Tri-linear interpolation is used to reconstruct missing data from the discrete data set and allows a value to be estimated for any position within the volume. Higher order interpolation schemes can be used but introduce additional computational complexity for little gain.



$$
\begin{aligned}
f_{xyz} = {} & f_{000}(1-x)(1-y)(1-z) + \\
& f_{100}x(1-y)(1-z) + \\
& f_{010}(1-x)y(1-z) + \\
& f_{110}xy(1-z) + \\
& f_{101}x(1-y)z + \\
& f_{011}(1-x)yz + \\
& f_{111}xyz
\end{aligned}
$$

Figure 12.4.1 - Tri-linear Interpolation

Figure 12.4.1 demonstrates the principal behind tri-linear interpolation. The normalised point within the unit cube is first converted to the discrete volume and its eight discrete corner values determined along with the normalised position within this new sub-unit cube. Placing these values within the equation f_{xyz} gives a linearly-interpolated value for the required point. The equation, although not complex, can quickly become a computational overhead where a

large number of interpolated values are required. This technique does not therefore lend itself well to normal image processing techniques where many samples are required for each iteration of an algorithm. However in the case of meshes/3D-surfaces where the presence of the surface greatly reduces the number of interpolations per iteration, the technique enables the dataset to be treated as a continuous volume, smoothing edges and noise.

Higher order interpolation schemes can be used (e.g. tri-cubic interpolation) however, the additional computational cost involved with such schemes outweighs the benefits gained. It should be pointed out that no matter which interpolation scheme is used it is never possible to reconstruct missing data, the values are merely estimated from the available information.

Volumetric data from the medical imaging field tends to have non-cubic voxels where the in-slice resolution is much smaller than that of the depth resolution, and for this reason the volume should be scaleable. This artefact of acquisition can be overcome by translating and rescaling the volume to a cube of 2 unit size. A scaling in x, y and z can then be applied to rescale the volume and associated voxels in to a cuberville (a volume with cubic voxels). Tri-linear interpolation will then attempt to fill-in this missing inter-slice resolution.

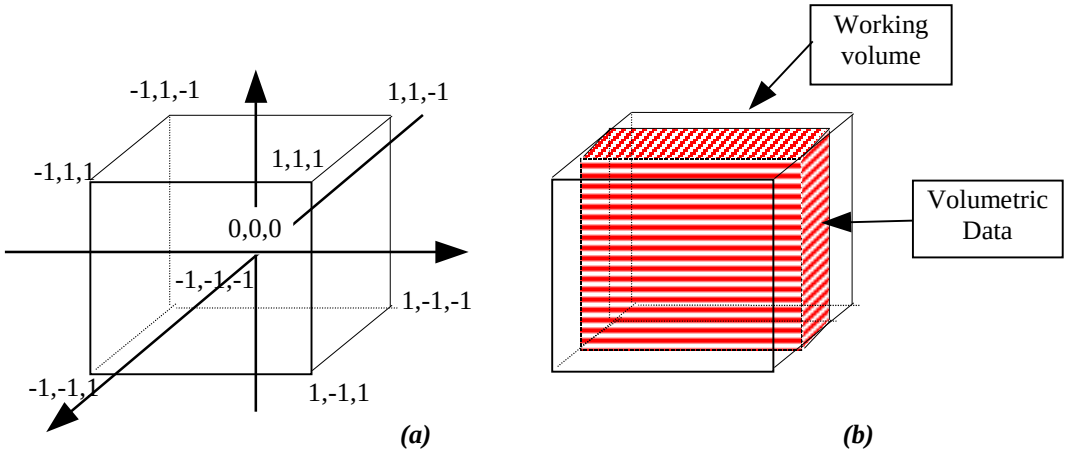


Figure 12.4.2 - The working volume of the 3Dinterpolator

Figure 12.4.2 shows this cube centred about the object, this enables meshes to be built that are of the same scale. For a given dataset the scale is set such that the

largest dimension of the volume occupies the full size of the unit cube centred about the origin. As the same scale applies to all dimensions a non cubic volume (eg 200x200,100 voxels) would produce a scaling demonstrated in Figure 12.4.2b, any attempt to access part of the volume outside the volumetric data as outside the cube results in a value of zero. This allows the dataset to easily be rescaled to suit applications.

12.5 The Balloon Model

The balloon model consists of a mesh of triangular facets or patches. The initial triangulated surface can be any shape or size allowing the re-application of a segmented surface to a new data set. Each node (vertex) has two forces acting upon it. The spring force derived from the sum of the vectors of the interconnections of the mesh, and the inflation force, derived from the weighted normal direction of the surface at each node.

The operation of the inflating balloon model can be encapsulated by the following algorithm.

Algorithm 12.1.

for a given closed form polygonal model do,
    build a connected mesh of vertices
    while number of polygons is not constant do
        compute the normal at each node
        for each node do,
            compute the elastic force using Equation 12.5-4 (See Section 12.5.2),
            test node position in dataset using feature detection scheme,
            if feature not found calculate the inflation force using Equation 12.5-5 (See Section 12.5.3) and add to the elastic force
            compute the new node position $v_i^{t+\Delta t}$ *using Equation 12.5-3 (Section 12.5.1) and update node*
            perform dynamic subdivision using Algorithm (See Section 12.5.4)

12.5.1 A Simple Dynamic Model.

The motion of any element *i* on a finite element mesh model can be described by the set of coupled second order differential equations [Terzopoulos 91]

Equation 12.5-1

$$m_i \frac{d^2 x_i}{dt^2} + \gamma_i \frac{dx_i}{dt} + g_i = f_i \qquad i = 1, \ldots, n.$$

Here, x is the location of the element *i*, m is its mass, g is the surface tension, generated by the interconnections of the elastic mesh, f is the inflation force and γ is the velocity-dependent damping coefficient that controls the rate of dissipation of kinetic energy. Giving the mesh these simulated physical properties provide a robust model that performs well but at a computational cost. The main rationale for the momentum term $\left(m_i \frac{d^2 x_i}{dt^2} \right)$ is its ability to reduce the mesh's susceptibility to noise. Due to the momentum of nodes the damping term γ is necessary to bring the model to rest. The mesh reaches an equilibrium state when $\frac{d^2 x_i}{dt^2} + \frac{dx_i}{dt} = 0$ which can take some time [Chen 95]. Chen and Medioni simplify this model by making *m=0* and $\gamma = 1$ for all *i* reducing Equation *12.5-1* to

Equation 12.5-2

$$\frac{dx_i}{dt} = f_i - g_i \qquad i = 1, \ldots, n.$$

Due to this simplification the equation (2) has a very simple explicit integration [Chen 95]

Equation 12.5-3

$$x^{t+\Delta t} = (f_i^t - g_i^t)\Delta t + x^t$$

Unlike the work of Terzopoulos, the approach described here does not use an explicit data force that attracts the balloon surface to image features. Instead the inflation force is used to inflate the surface until the desired feature is located. In order to overcome the noise inherent in medical imaging datasets, the surface is not anchored to positive data features. When a feature is detected at a node position, the inflation force is removed for that node. The surface is then free to oscillate around features until it converges on a solution.

12.5.2 Simplified Spring Force

The spring force exerted on node *i* by the spring linking node *i* and *j* of natural length l_{ij} can be expressed as [Terzopoulos 91],

$$s_{ij} = \frac{c_{ij} e_{ij}}{\left\| r_{ij} \right\|} r_{ij}$$

where c_{ij} is the stiffness, $r_{ij} = x_j - x_i$ the vector separation of the nodes, $\left\| r_{ij} \right\|$ is the length of the spring and $e_{ij} = \left\| r_{ij} \right\| - l_{ij}$ is the deformation of the spring.

In order to generate a generic technique for the segmentation of objects, and due to the large nature of 3D objects it is not feasible to assign values to c_{ij} and l_{ij} for each node. Further simplifications can therefore be made by setting all stiffness coefficients to a constant value with a minimum spring length of zero, $c_{ij} = c$ and $l_{ij} = 0$.

The total elastic force on a node *i* is therefore,

Equation 12.5-4

$$g_i = \frac{c}{n} \sum_{j=0}^{n} r_{ij}$$

12.5.3 Inflation Force

The inflation force applied to each node *i* is

Equation 12.5-5 $$f = k\hat{n}_i$$

where $\hat{n}_i$ is the normal at node *i* and k is the amplitude of the inflation force. The value of *k* can be selected to be a constant for a specific data set or dynamically generated as $k = \frac{5}{4}\|g_i\|$, which ensures that the inflation force for each node always exceeds the surface tension of the model. Although this removes the parameter selection of *k*, it produces a slower convergence on solutions as non optimum parameter selection results.

Node normals are calculated as the average normal of the surrounding polygons sharing the node *i*, gained from the cross product of polygonal edges between vertices. Other, more complicated schemes as used by Chen and Medioni [Chen 95], give little benefit as errors in this normal estimation technique are reduced by the surface smoothing properties of the surface tension (elastic force). This also gives a significant performance increase as normals must be recalculated at least once every iteration of the algorithm.

12.5.4 Dynamic Subdivision

As the inflation force increases the surface area of the mesh, individual polygons grow in size. Since the elastic force is directly proportional to the size of polygons, there comes a point where the elastic force will not allow the mesh to increase in size further, unless the inflation amplitude is increased accordingly. Dynamic subdivision can be used to subdivide polygons which exceed set size criteria and keep polygons within a suitable limit. Each edge of the mesh is checked in turn at each iteration to see if it exceeds the subdivision threshold. Figure 12.5.1 demonstrates how the process works.

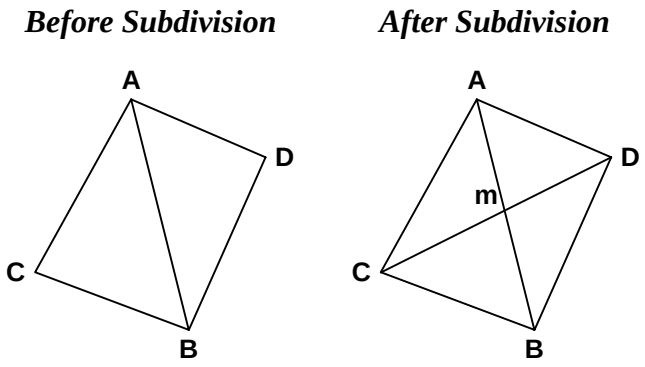


Figure 12.5.1 - Dynamic Subdivision

When the length of a node connection **AB** exceeds set criteria, distance or curvature, the two triangles that contain this edge are located (**ABC, ADB)** and removed from the polygon list. The midpoint **m** of **AB** is calculated and four new triangles constructed **AMC, CMB, ADM, and MDB.** The internal connectivity of the mesh is also altered to reflect this new local structure. Long thin triangles are undesirable, as they do not model local surface properties well. This technique ensures that they never occur, as any edge that exceeds a distance threshold is immediately subdivided. This procedure allows the mesh to grow asymmetrically to fit any feature located within the data set.

The dynamic subdivision procedure can be encapsulated by the following algorithm.

Algorithm 12.2.

- *for each node (**V1**) do*
  - *for each connection to another node (**V2**) do*
    - *if the connection (**V1V2**) matches the subdivision criteria do*
      - *remove connection (**V1V2**)*
      - *remove the two polygons that share this edge*
      - *find the mid point **m** of **V1V2***
      - *construct four polygons using **m** as a common node*
      - *update the connections of the mesh*
- *recalculate the normal at each node*

12.5.5 Subdivision Criteria

Using a distance threshold for subdivision produces an evenly spaced mesh which can alter its structure locally to fit any dataset. It is also possible to use other criteria to provide a more flexible approach. As the normal at each node is known for use with the inflation force, the dot product of two adjacent vertices' normals represents local surface curvature. This can be used to further subdivide the mesh if the dot product drops below a certain threshold value, i.e. the area has a high degree of curvature, allowing more vertices to be placed in these areas of high curvature. This is useful where long narrow features are present in the dataset.

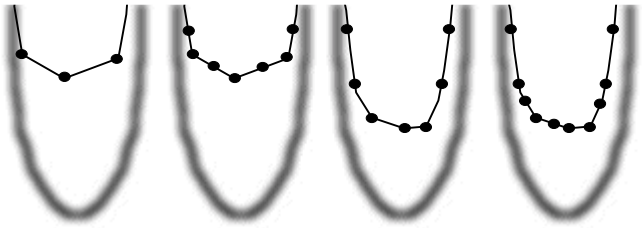


Figure 12.5.2- Curvature Based Subdivision

Figure 12.5.2 demonstrates an image boundary and an inflating balloon front. The boundary shown has found an equilibrium state in the narrow feature. By subdividing the mesh on a curvature basis, in addition to distance, extra vertices are added to the front of the model providing the inflation force needed to successfully segment the long narrow feature.

Both subdivision criteria can be used in conjunction to minimise the polygon count of a mesh, removing the need for post-processing techniques such as Delaunay Triangulation [Soucy 96]. An edge is subdivided only if it exceeds both a distance and a curvature threshold. Polygons on parts of the surface with low curvature grow beyond the threshold keeping polygon counts to a minimum. Therefore, areas of high curvature have larger numbers of small polygons that better model the surface features.

12.5.6 Feature Detection

Edge features within an image are typically identified as a change in intensity from one range to another via an isointensity which depicts the boundary of these two regions.

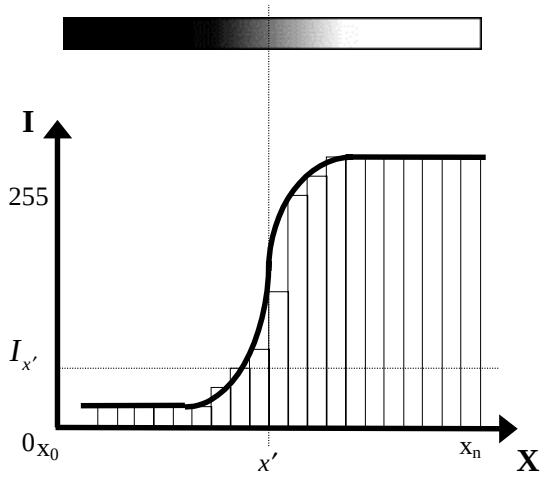


Figure 12.5.3 - The Boundary between Light and Dark

Figure 12.5.3 shows a cross section through an image depicting a sharp boundary between light and dark. The intensity x_i depicts the threshold that would generate an isointensity boundary for this feature within an image. Providing scanning starts within the model boundary, it can be said the boundary (x_i) has been passed when either

$$I_{x'} < I_{x_i} \quad or \quad I_{x'} > I_{x_i} \quad where \quad i = 0,1,\ldots,n$$

depending on the direction of the intensity gradient along the isosurface boundary normal.

This simple thresholding mechanism can be used to detect when the balloon has just passed through a possible isosurface boundary, at which point the inflation force can be removed for that node. Due to the simplicity of this mechanism, many false boundary points are detected and hence results in a noisy segmentation. However, elasticity is a constant force and as such provides the function of a simple momentum term which pulls the nodes away from false boundary points.

Where complicated internal structures are required this approach may not provide adequate results. In this situation, other more sophisticated feature detection schemes can be employed. However as the feature of primary concerned is the external boundary of the model, where a distinct boundary is present, this approach provides an efficient and simple solution.

12.5.7 Robustness to Noise

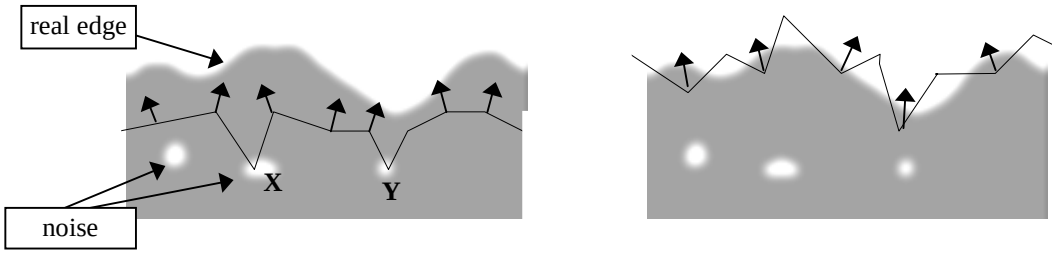


Figure 12.5.4 - Balloon Boundary,
(a) Contour is pulled away from noise (b) Contour oscillates at real edge

Figure 12.5.4 demonstrates this invulnerability to noise spikes. In Figure 12.5.4(a) the boundary moves towards the true boundary through the influence of the inflationary force. Points X and Y are located on noisy areas of the image. Where these false edges are located the inflation force is removed. However, as the remainder of the contour progresses forward under the inflation force the elasticity pulls these points away from the noise. Once a sufficient distance from the noise has been reached the edge detection criteria no longer apply and the inflation force is reapplied. Elasticity then helps smooth these features as the process iterates. Figure 12.5.4b demonstrates what happens when the contour approaches the true boundary. As points are inflated beyond the boundary their inflation force is removed and elasticity pulls the point back within the model, where the inflation force is then re-applied. This causes the contour to oscillate around the true edge. As points oscillate back and forth chaotically their overall movement is at a minimum and therefore mesh subdivision approaches zero. At this point a local edge detection scheme can be used to clamp nodes onto their

closest edge. This creates an evenly spaced mesh that is a good surface approximation to the desired object.

12.6 Results

12.6.1 Synthetic Dataset

A synthetic data set of a 3D-horseshoe shape was constructed. The volume consisted of 20x20x6 cubic voxels where each 20x20 slice is identical throughout the volume. Figure 12.6.1 shows one slice from this volume. An initial diamond-shaped seed balloon consisting of 8 vertices is placed inside the object and the model grown to fill the volume. The resulting surface segmentation is shown in Figure 12.6.2. As the model expands to fill the volume, vertices that reach the outer boundary oscillate as their inflation force is turned on and off. The resulting segmentation has almost a circular cross section although the original data had very distinct straight edges. This is due to the tri-linear interpolation which smoothes the data, and is very apparent due to the low number of constituent voxels within the volume. The ends of the model continue to grow under the inflation force and as the distances between vertices increases the dynamic subdivision introduces addition polygons allowing the model to locally deform to fit the dataset.

$x \rightarrow$

$y \downarrow$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	255	255	255	255	255	255	0	0	0	0	0	0	0
0	0	0	0	0	0	255	255	255	255	255	255	255	255	0	0	0	0	0	0
0	0	0	0	0	255	255	255	255	0	0	255	255	255	255	0	0	0	0	0
0	0	0	0	255	255	255	0	0	0	0	0	0	255	255	255	0	0	0	0
0	0	0	255	255	255	0	0	0	0	0	0	0	0	255	255	255	0	0	0
0	0	255	255	255	0	0	0	0	0	0	0	0	0	0	255	255	255	0	0
0	0	255	255	0	0	0	0	0	0	0	0	0	0	0	0	255	255	0	0
0	255	255	255	0	0	0	0	0	0	0	0	0	0	0	0	255	255	255	0
0	255	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	255	0
0	255	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	255	0
0	255	255	255	0	0	0	0	0	0	0	0	0	0	0	0	255	255	255	0
0	0	255	255	0	0	0	0	0	0	0	0	0	0	0	0	255	255	0	0
0	0	255	255	255	0	0	0	0	0	0	0	0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0	0	0	0	0	0	0	255	255	255	0	0	0
0	0	0	0	255	255	255	255	0	0	0	0	255	255	255	255	0	0	0	0
0	0	0	0	0	255	255	255	0	0	0	0	255	255	255	0	0	0	0	0
0	0	0	0	0	0	255	255	0	0	0	0	255	255	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12.6.1 - Single slice of Synthetic Dataset

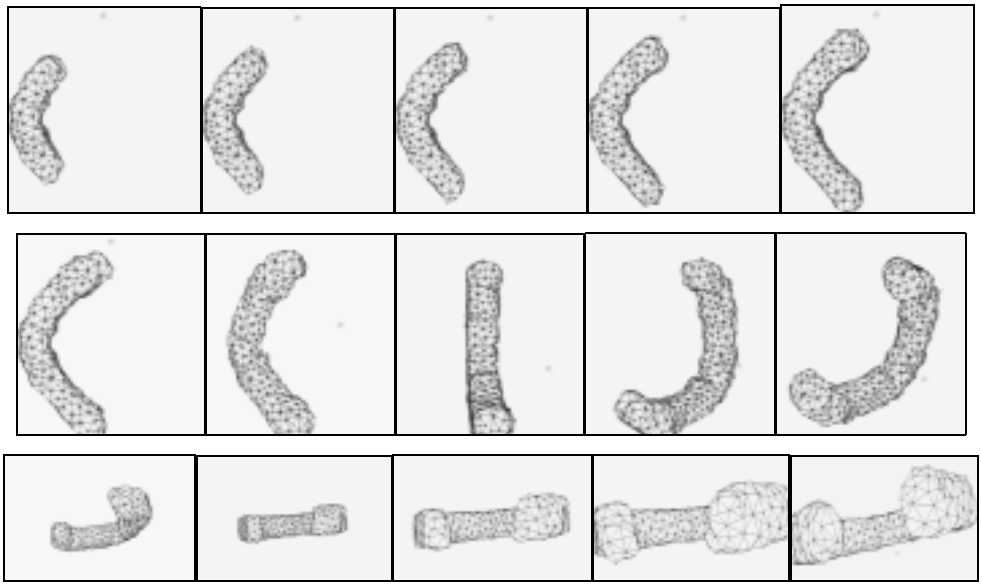


Figure 12.6.2 - Balloon Growing to fill Synthetic Dataset

12.6.2 MRI Dataset

To demonstrate the ability of the balloon model segment a real volumetric dataset the model was applied to a raw MRI scan of a human hand[11]. This is also compared to the results of segmentation gained from a standard isosurface and an 3D elastic mesh model (3D snake). The volume is 256x256x20 voxels in size. This is rescaled by 1x1x2 to reconstruct a cuberville and tri-linear interpolation used to estimate values within the volume. Figure 12.6.3 shows an insosurface generated from the dataset. Although it clearly shows the shape of the hand within the volume the surface is discontinuous and noisy. The background noise in the image is perhaps the most prominent feature and is the cause of the speckled effect of the surface. Another disadvantage of the technique (as mentioned earlier) is that for each voxel, a number of polygons are produced. The isosurface shown in Figure 12.6.3 was generated from a super-sampled volume of 128x128x20 to allow the resulting model to be rendered as a surface generated from the original volume would result in some 235,000 polygons.

[11] MRI data of the hand model was provided by the Centre for Medical Imaging Research (CoMIR) at the University of Leeds

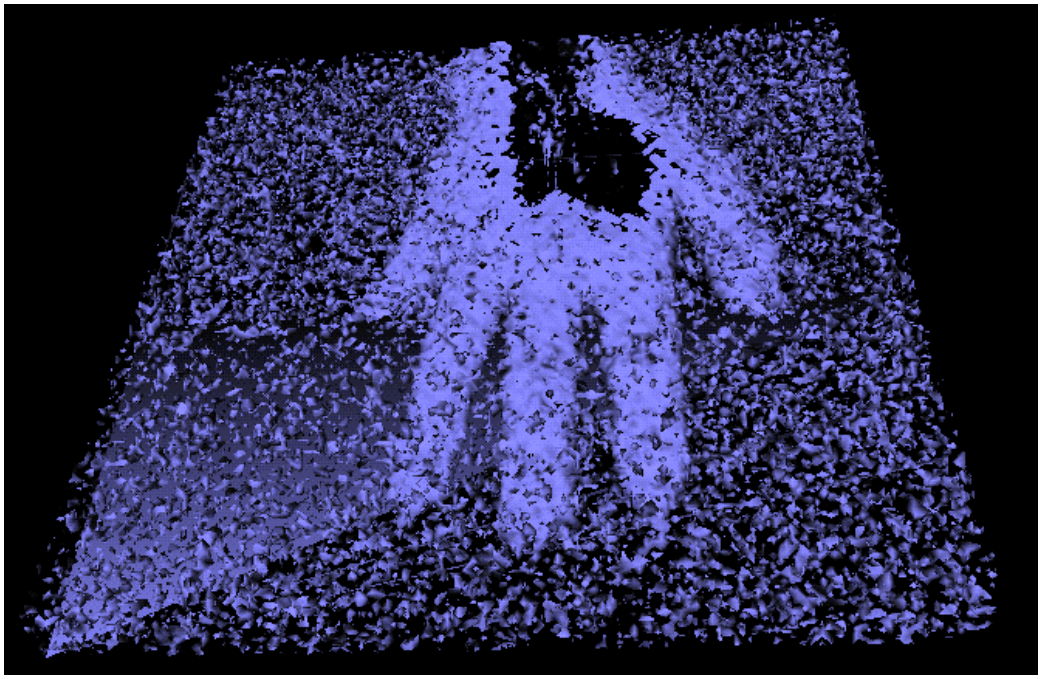


Figure 12.6.3 - Isosurface of MRI Hand Dataset

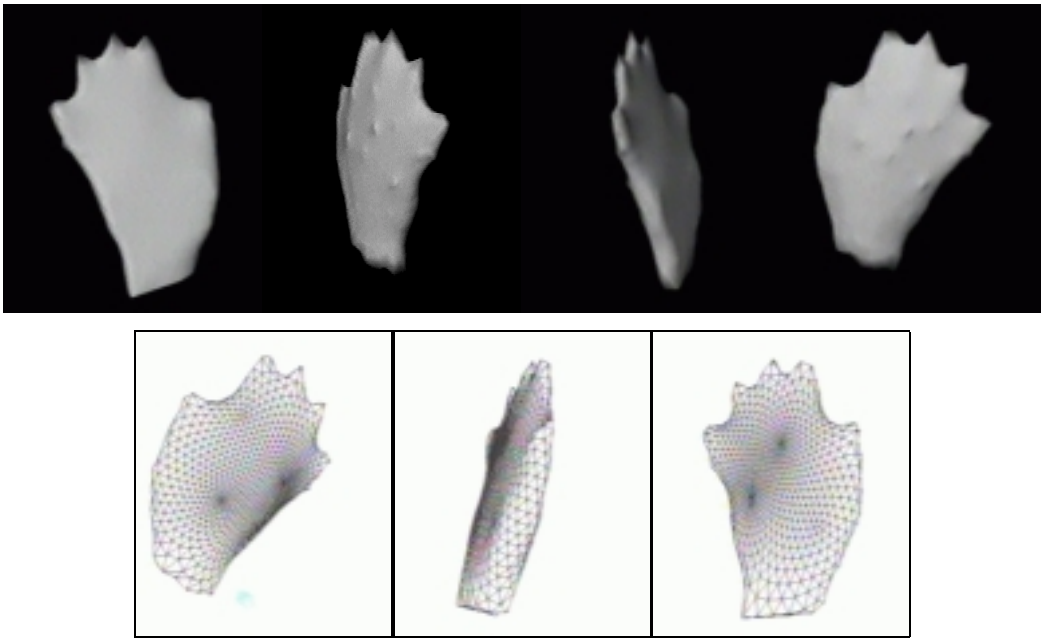


Figure 12.6.4 - 3D Surface Snake Applied to MRI Hand Dataset

Figure 12.6.4 shows the results of applying a 3D elastic surface to the dataset. This produces a poor segmentation for two reasons

1. The large amount of background noise in the volume means that the snake easily gets stuck as it shrinks to fit around the hand.

2. The long narrow features of the fingers make it difficult for the surface to succesfully segment their structure.

By increasing the data attraction force of the snake, the ability to locate and segment the fingers is increased. However, if this attraction force is increased the suseptibility to background noise is also increased and segmentation fails.

Figure 12.6.5 shows the development of the balloon mesh when applied to this dataset. Initially, a seed balloon is placed within the volumetric dataset.

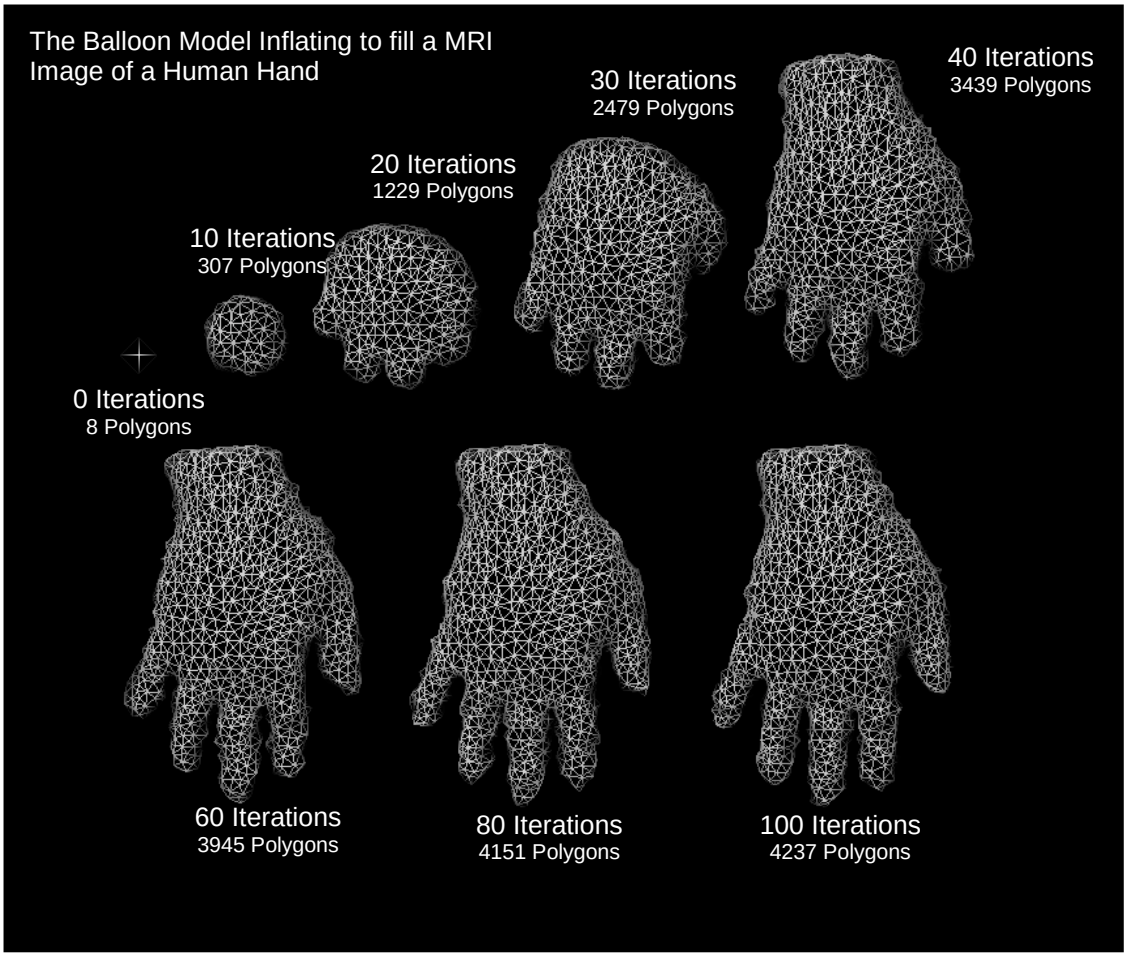


Figure 12.6.5 - Segmentation of an MRI dataset of the Human Hand

The seed consists of a simple diamond shape with 8 polygons and 6 vertices. Forces are applied to the model and after 10 iterations it has grown to 307 polygons. The almost spherical shape is due to the surface tension of the model. Its non-spherical symmetry shows that positive features have been detected early

on in the process and thus the inflation force has not been applied evenly. This demonstrates the algorithm's robustness to false boundaries and noise.

As the process iterates further the final shape very quickly starts to take form. Although mesh subdivision continues we can see that it is starting to decrease in rate considerably after 40 iterations. Figure 12.6.6 shows the rate of growth of the mesh.

Figure 12.6.6 - Graph Showing the Rate of Polygonal Increase.

Although the model will finally converge on a stable solution, it is sufficiently complete at around 70 iterations which takes approximately 35 seconds on a single MIPS R4400 200MHz processor, including render time. This is significantly faster than previous researchers' techniques, the most comparable being the work of Chen and Medioni [Chen 95], where a comparable complexity model takes approximately 30 mins to iterate on a SUN Sparc-10 machine. This can also be compared with a standard isosurface of the external hand boundary that generated a surface of 235000 polygons as compared to the balloon model of 4000 polygons.

The hand dataset is a good example of the effectiveness of the technique, demonstrating its ability to work with complex noisy images which contain an object with convex, concave and long narrow features.

12.7 Conclusions

This chapter has presented a surface segmentation method which uses a simulated inflating balloon model to estimate structure from volumetric data using a triangular mesh. The model uses simulated surface tension and an inflationary force to grow from within an object and find its boundary. Mechanisms have been described that allow either evenly spaced or minimal polygonal count surfaces to be generated. Unlike previous work by researchers, the technique uses no explicit attraction to data features and as such is less dependent on the initialisation of parameters and local minima. Instead, the model grows under its own forces, never anchored to boundaries but constrained to remain inside the desired object. Results have been presented that demonstrate the technique's ability and speed at the segmentation of a complex, concave object with narrow features, while keeping model complexity within acceptable limits.

12.8 Future Work

This work is ongoing, the primary rationale being the ability to produce low level polygonal surface approximations to allow 3D Point Distribution Models to be built for automatic recognition, segmentation and analysis of volumetric data. Work has also been done in the area of mesh self-intersection. A set of criteria have been developed which allow the detection of mesh self-intersection. Future work includes allowing this criteria to be used to detect intersections, and re-join the mesh at these points to allow more complex torus like shapes to be successfully extracted.

References

Ahmad, T., Taylor, C. J., Lantis, A., Cootes, T. F., Tracking and Recognising Hand Gestures Using Statistical Shape Models. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK: BMVA, 1995:403-412, 1995.

Azarbayejani, A. and Penland, A., Real-time self calibrating stereo person tracking using 3D shape estimation from blob features, ICPR'96, Vienna, Austria, 1996.

Ballard, H. D., Brown, C. M., Computer Vision, Prentice-Hall Inc London, 1992.

Barequet, G. and Sharir, M., Piecewise-Linear Interpolation between polygonal Slices. Computer Vision and Image Understanding, 63(2), 251-272, 1996.

Baumberg, A. and Hogg, D., An Adaptive Eigenshape Model. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK: BMVA, 1995:87-96.

Baumberg, A., and Hogg, D., Generating Spatiotemporal Models from Examples. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK: BMVA, 1995:413-422.

Beale, R. and Jackson, T., Pattern Recognition In: *Neural Computing an Introduction*, pp 15-37, IOP Publishing Ltd, 1990.

Bezdek, J. C., Ehrlich, R., Full, W., FCM: The Fuzzy cc-means Clustering Algorithm, Computers and Geosciences, 10(2-3), 1984, pp191-203.

Blake, A. and Isard, M., Active Contours, Springer Verlag, 1998.

Blake, A., Curwen, R, Zisserman, A., Framework for spatio-temporal control in the tracking of visual contours, Int J. Computer Vision, 11(2), 1993, pp127-145.

Bowden, R., Heap, A. J., and Hart, C., Virtual Datagloves: Interacting with Virtual Environments Through Computer Vision, in Proc. 3rd UK VR-Sig Conference, DeMontfort University, Leicester, UK, July 1996.

Bowden, R., Heap, A. J., and Hogg, D., C., Real Time Hand Tracking and Gesture Recognition as a 3D Input Device for Graphical Applications, Progress in Gestural Interaction, in: Harling P A, Edwards A D N Eds (Springer Verlag, London, 1997) 117-129.

Bowden, R., Mitchell, T. A., Sahardi, M., Cluster Based non-linear Principal Component Analysis, IEE Electronics Letters, 23rd Oct 1997, 33(22), pp1858-1858.

Bowden, R., Mitchell, T. A., Sahardi, M., Non-linear Statistical Models for the 3D Reconstruction of Human Pose and Motion from Monocular Image Sequences. To appear in Image and Vision Computing.

Bowden, R., Mitchell, T. A., Sahardi, M., Real-time Dynamic Deformable Meshes for Volumetric Segmentation and Visualisation, In Proc. BMVC, Adrian F. Clark Ed, Vol 1, pp 310-319, Essex, UK, Sept 1997.

Bowden, R., Mitchell, T. A., Sahardi, M., Reconstructing 3D Pose and Motion from a Single Camera View, In Proc. BMVC, John N. Carter and Mark S. Nixon Eds, Uni of Southampton, Vol 2, pp , Southampton, Sept 1998.

Bowden, R., Non-linear Point Distribution Models, In CVonline: On-Line Compendium of Computer Vision [Online]. R. Fisher (ed). Section 11.3.1.2 , Oct 98.

Bowden, R., Tools and Techniques for Three-Dimensional Computer Vision, a Literature Review, Technical Report, Department of Systems Engineering, Brunel University, May 1996.

Boyle, R., Lecture notes in, Pattern Recognition and Neural Networks, University of Leeds, 1995.

Bregler, C., and Omohundro, S., Surface Learning with Applications to Lip Reading, Cowan. J. D., Tesauro, G., and Alspector, J.(eds), Advances in Neural Information Processing Systems 6, San Francisco, CA: Morgan Kaufmann Publishers, 1994.

Carlbom, I., Terzopoulos, D., and Harris, K. M., Computer Assisted Registration, Segmentation, and 3D Reconstruction from Images of Neuronal Tissue Sections. IEEE Transactions on Medical Imaging, 13(2), 351-362, 1994.

Chen, Y., and Medioni, G., Description of Complex Objects from Multiple Range Images Using an Inflating Balloon Model., Computer Vision and Image Understanding, 61(3), 1995, 325-334.

Chiou, G. I., and Hwang, J. N., A Neural Network-Based Stocastic Active Contour Model (NNS-SNAKE) for Contour Finding of Distinct Features. IEEE Transactions on Image Processing, 4(10), 1407-1416, 1995.

Cipolla, R., and Blake, A. Surface Orientation and Time to Contact from Image Divergence and Deformation. Paper presented at the Second European Conference on Computer Vision - ECCV'92, Santa Marghertia Ligure, Italy, 1992.

Cline, H. E., Lorenson, W. E., Ludke, S., Crawford, C. R., and Teeter, B. C., Two Algorithms for the 3D Reconstruction of Tomograms. Medical Physics, 15(3), 320-327, 1988.

Cootes, T. F., and Taylor, C. J., A Mixture Model for Representing Shape Variation, in: Clark A F, ed., British Machine Vision Conference 1997, (BMVA, Essex, UK, 1997) 110-119.

Cootes, T. F., Di Mauro, E. C., Taylor, C. J., Lantis, A., Flexible 3D Models from Uncalibrated Cameras. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK:BMVA, 1995:147-156.

Cootes, T. F., Edwards, G. J. and Taylor, C. J., "Active Appearance Models", in Proc. European Conference on Computer Vision 1998 (H.Burkhardt and B. Neumann Ed.s). Vol. 2, pp. 484-498, Springer, 1998.

Cootes, T. F., Edwards, G. J. and Taylor, C. J., A Comparative Evaluation of Active Appearance Model Algorithms, BMVC98.

Cootes, T. F., Page, G. J., Jackson, C. B., Taylor, C. J., Statistical Grey-Level Models for Object Location and Identification. Image and Vision Computing. 14(8) Aug 1996, pp. 533-540.

Cootes, T. F., Taylor, C. J., Active Shape Model Search using Local Grey-Level Models: A Quantitative Evaluation, in Proc. British Machine Vision Conference, (Ed. J.Illingworth), BMVA Press, 1993, pp.639-648.

Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J., “Active Shape Models - Their Training and Application.”, Computer Vision and Image Understanding, 61(1), 1995, 38-59.

Cootes, T. F., Taylor, C. J., Cooper, D. H., Graham, J., Active Shape Models - Their Training and Application. Computer Vision and Image Understanding, 1995;61(1):38-59.

Cootes, T. F., Taylor, C. J., Lanitis, A., Cooper, D. H. and Graham, J., Building and Using Flexible Models Incorporating Grey-Level Information. Proc. Fourth

International Conference on Computer Vision, IEEE Computer Society Press, 1993, pp.242-246.

Delingette, H., Simplex Meshes: A General Representation for 3D Shape Reconstruction. Technical Report 2214, INRIA, 1994.

Elsayed, A., Reliability engineering, Addison Wesley Longman, pp8-9, 1996.

Etoh, M., Shirai, Y., and Asada, M., Contour Extraction by Mixture Density Description Obtained from Region Clustering. Paper presented at the Second European Conference on Computer Vision - ECCV'92, Santa Marghertia Ligure, Italy, 1992.

Fels, S. S., and Hinton, G. E., Building Adaptive Interfaces with Neural Networks: The Glove-Talk Pilot Study, *Human-Computer Interaction - INTERACT '90*, Diaper, D *et al* (eds), pp683-688, Elsevier Science Publishers B.V. (North-Holland), 1990.

Ferryman, J. M., Worrall, A. D., Sullivan, G. D., and Baker, K. D., A generic deformable model for vehicle recognition. Paper presented at the British Machine Vision Conference 1995, BMVC'95, University of Birmingham, Birmingham, UK, 1995.

Foley, J. D., Van Dam, A., Feiner, S. K., Hughes, J. F., "Achromatic and Coloured Light.", Chapter 13, Computer Graphics, Principals and Practice, pp563-604, Addison-Wesely 1990.

Fuchs, H., Kedem, Z. M., and Ulselton, S. P., Optimal Surface Reconstruction from Planar Contours. Commun. ACM, 20(10), 693-702, 1977.

GAlib, A C++ Library of Genetic Algorithm Components, http://lancet.mit.edu/ga/, MIT.

Goshtasby, A., Design and Recovery of 2D and 3D Shapes Using Rational

Gaussian Curves and Surfaces. International Journal on Computer Vision, 10(3), 233-256, 1993.

Hall, P., M., Marshall, D., Martin, R., R., Incremental Eigenanalysis for Classification, Research Report Series No:98001, Dept Computer Science, University of Wales, Cardif, 1998.

Handouyahia, M., Ziou, D., Wang, S., Sign Language Recognition using Moment-Based Size Functions, Vision Interface '99, Trois-Riveres, Canada, 19-21 May 99.

Heap, T. and Hogg D. C., Automated Pivot Location for the Cartesian-Polar Hybrid Point Distribution Model, in: Pycock D, ed., British Machine Vision Conference 1995, (BMVA, Birmingham, UK, 1995) 97-106.

Heap, T., and Hogg D. C., Improving Specificity in PDMs using a Hierarchical Approach, in: Clark A F, ed. British Machine Vision Conference 1997, (BMVA, Essex, UK, 1997) 80-89.

Heap, T., Hogg D. C., 3D Deformable Hand Models, Progress in Gestural Interaction, (Proceedings of Gesture Workshop, York, April, 1996)

Heap, T., Hogg D. C., Automated Pivot Location for the Cartesian-Polar Hybrid Point Distribution Model. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK: BMVA, 1995:97-106.

Herman, G. T., and Liu, H. K., Three-Dimensional Display of Human Organs from Computed Tomography. Computer Graphics and Image Processing, 9(1), 1-21, 1979.

Hill, A., Cootes, T. F., Taylor, C. J., A generic system for image interpretation using flexible templates, in British Machine Vision Conference, Springer Verlag, 1992.

Hill, A., Cootes, T. F., Taylor, C. J., Active Shape Models and the Shape Approximation Problem. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK: BMVA, 1995:157-166.

Hill, A., Taylor, C. J., Cootes, T., Object Recognition by Flexible Template Matching using Genetic Algorithms. In: Sandini G, ed. Second European Conference on Computer Vision - ECCV'92. Santa Marghertia Ligure, Italy: Springer-Verlag, 1992:852-856.

Hill, A., Taylor, C. J., Model based image interpretation using genetic algorithms, In Proceedings British Machine Vision Conference, Springer-Verlag, 1991, pp 266-274.

Hill, A., Taylor, C. J., Model based image interpretation using genetic algorithms, Image Vision Computing. 10, 1992, 295-300.

Hogg, D., C., Model-based vision: a program to see a walking person. Journal of Image and Vision Computing, 1(1), pp5-20.

Hunke, M., Waibel, A., Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, Monterey, California, Nov 1994.

Isard, M. and Blake, A., Condensation - conditional density propagation for visual tracking. International Journal of Computer Vision, 1998.

Ivins, J., and Porrill, J., Constrained Active Region Models for Fast Tracking in Colour Image Sequences. Computer Vision and Image Understanding, 1998, 72(1), pp54-71.

Karayiannis, N., B., Pai, P., Fuzzy Vector Quantization Algorithms and Their Application in Image Compression., IEEE Transactions on Image Processing, 4(9), 1995, pp1193-1201.

Kass, M., Withkin, A., and Terzopoulos, D., Snakes: Active Contour Models., International Journal of Computer Vision, 1988, 321-331.

Kass, M., Withkin, A., and Terzopoulos, D., Snakes: Active Contour Models. Paper presented at the Proceedings of the First International Conference on Computer Vision, London, 1987.

Kendall, M., Multivariate Analysis, Charles Griffin and Company Ltd, 1980.

Keppel, E., Approximating Complexe Surfaces by Triangulation of Contour Lines. IBM Journal of Research and Development, 19(1), 2-11, 1975.

Konheim, A., G., Cryptography: A Primer, John Wiley, New York, 1982.

Kotcheff, A. C. W., Taylor, C. J., Automatic Construction of Eigenshape Models by Gentic Algorithms. In: Proc. International Conference on Information Processing in Medical Imaging 1997, Lecturenotes in Computer Science Issue 1230, Springer Verlag, pp1-14, 1997.

Krishnapuram, R., Frigui, H., Nasraoui, O., Fuzzy and Possibilistic Shell Clustering Algorithms and Their Application to Boundary Detection and Surface Approximation - Part I. IEEE Trans on Fuzzy Systems, 3(1), pp29-43, 1995.

Krishnapuram, R., Frigui, H., Nasraoui, O., Fuzzy and Possibilistic Shell Clustering Algorithms and Their Application to Boundary Detection and Surface Approximation - Part II. IEEE Trans on Fuzzy Systems, 3(1), pp45-60, 1995.

Lantis, A., Taylor, C. J., Cootes, T. F., An Automatic Identification System Using Flexible Appearance Models. In: Hancock E, ed. British Machine Vision Conference 1994 - BMVC'94. Univeristy of York, York.: BMVA Press, pp65-74, 1994.

Lobregt, S., and Viergever, M. A., A Discrete Dynamic Contour Model. IEEE Transactions on Medical Imaging, 14(1), pp12-24, 1995.

Lorenson, W. E., and Cline, H. E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm. IEEE Trans. Nucl. Sci, Vol 38, pp748-754, 1991.

Magee, D., Boyle, R., Building Class Sensitive Models for Tracking Applications, In Proc. British Machine Vision Conference, BMVC'99, Nottingham University, pp 594-603.

Magee, D., Boyle, R., Building shape models from image sequences using piecewise linear approximation. In Proc. British machine Vision Conference, BMVC'98, Southampton University, pp 398-408, 1998.

McInery, T., and Terzopoulos, D., A finite element model for 3D shape reconstruction and nonrigid motion tracking, In Proc. International Conference on Computer Vision, Berlin, Germany, pp518-523, May 1993.

McKenna, S., Gong, G., and Raja, Y., Face Recognition in Dynamic Scenes, in: Clark A F, ed. British Machine Vision Conference 1997, (BMVA, Essex, UK), pp140-151, 1997.

Moshfeghi, M., Ranganath, S., and Nawyn, K., "Three-Dimensional Elastic Matching of Volumes. IEEE Transactions on Image Processing", 3(2), pp128-137, 1994.

Mullick, R., and Ezquerra, N. F., Automatic Determination of LV Orientation from SPECT Data. IEEE Transactions on Medical Imaging, 14(1), pp88-99, 1995.

O'Toole, A. J., Vetter, T., Troje, N., Bulthoff, H. H., IMAGING: Engendering Faces. Scientific American, April 1996.

Pal, N., R., Bezdek, J., C., On Cluster Validity for the Fuzzy c-Means Model, IEEE Trans on Fuzzy Systems, 3(3), 1995, pp370-379.

Parker, L., Utilising Human Audio Visual Response for Lip Synchronisation in VE, VRSIG97, Proceedings of the 4th UKVRSIG Conference, Richard Bowden Ed, Brunel University, UK, Nov 1997.

Pentland, A., P., Smart Rooms, Scientific American, pp54-62, April 1996.

Russ, J., C., The Image Processing Handbook, CRC Press, 1994.

Ross, S, M., Introduction to Probability and Statistics for Engineers and Scientists, John Wiley and Sons, 1987.

Rueckert, D., and Burger, P. Contour fitting using an adaptive spline model. Paper presented at the British Machine Vision Conference 1995, BMVC'95, University of Birmingham, Birmingham, UK, 1995.

Schnabel, J. A., and Arridge, S. R., Active Contour Models for Shape Description Using Multiscale Differential Invariants. Paper presented at the British Machine Vision Conference, BMVC'95, University of Birmingham, Birmingham, UK, 1995.

Shirley, P., and Tuckman, A., A Polygonal Approximation to Direct Scalar Volume Rendering. Computer Graphics, 24(5), pp51-58, 1990.

Soucy, M., and Laurendeau, D., Multiresolution Surface Modeling Based on Hierarchical Traingulation., Computer Vision and Image Understanding, 63(1), pp1-14, 1996.

Sozou, P. D., Cootes, T. F., Taylor, C. J., and Di-Mauro, E. C., Non-Linear Point Distribution Modelling using a Multi-Layer Perceptron, in: Pycock D, ed. British Machine Vision Conference 1995, (BMVA, Birmingham, UK), pp107-116, 1995.

Sozou, P. D., Cootes, T. F., Taylor, C. J., and Di-Mauro, E. C., A Non-linear Generalisation of PDMs using Polynomial Regression, in: Hancock E, ed. British Machine Vision Conference 1994, (BMVA, York) pp397-406, 1994.

Sozou, P. D., Cootes, T. F., Taylor, C. J., Di-Mauro, E. C., Non-Linear Point Distribution Modelling using a Multi-Layer Perceptron. In: Pycock D, ed. British Machine Vision Conference 1995, BMVC'95. University of Birmingham, Birmingham, UK: BMVA, pp107-116, 1995.

Stiefelhagen, R., Yang, J., Gaze Tracking for Multimodal Human-Computer Interaction, Proceedings of ICASSP'97.

Stiefelhagen, R., Yang, J., Waibel, A., Towards Tracking Interaction between People, Proceedings of the Intelligent Environments AAAI Spring Symposium, Stanford Uni, California, March 23-25, 1998.

Sumpter, N., Boyle, R. D., and Tillett, R., D., Modelling Collective Animal Behaviour using Extended Point Distribution Models, in: Clark A F, ed. British Machine Vision Conference 1997, (BMVA, Essex, UK), pp242-251, 1997.

Swets, D., Weng, J., Using Discriminant eigenfeatures for Image Retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 18, pp831-836, 1996.

Terzopoulos, D., and Vasilescu, M., Sampling and Reconstruction with Adaptive Meshes, in Proc. Conference on Computer Vision and Pattern Recognition, Maui, HI, pp70-75, June 1991.

Turk, M., Pentland, A., Eigenfaces for recognition, Journal of Cognitive Neroscience, 3(1), pp71-86, 1991.

Ueda, N., and Mase, K., Tracking Moving Contours Using Energy-Minimizing Elastic Contour Models. Paper presented at the Second European Conference on Computer Vision - ECCV'92, Santa Marghertia Ligure, Italy, 1992.

Waibel, A., Duchnowski, P., Connectionist Models in Multimodal Human-Computer Interaction, Proceedings of Goverment, Microcircuit Applications Conference GOMAC'94, SanDiego, Nov 1994.

Wall, M., B., A Genetic Algorithm for Resource-Constrained Scheduling, Doctoral Thesis, Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, June 1996. http://lancet.mit.edu/~mbwall/phd/

Walpole, R., E., Myers, R., H., Myers, S., L., Probability and Statistics for Engineers and Scientists, Prentice Hall International, Inc, pp108-110, 1998.

Waston, R., A Survey of Gesture Recognition Techniques. Dept of Comp. Sci., Trinity College, Dublin, Technical Report, pages TCD-CS-93-11, 1993.

Wolfson, H., J., Lamdan, T., Transformation invariant indexing. Geometric Invariance in Computer Vision, Mundy, J., L., Zisserman, A., (Eds), pp335-353. MIT Press 1992.

Yang, J., Lu, W., Waibel, A., Skin-Color Modeling and Adaptation, Proceedings of ACCV'98, vol. II. pp687-694 (Hong Kong), 1998.

Yang, J., Waibel, A., A real-time face tracker, Proceedings of the Third IEEE Workshop on Application of Computer Vision, Sarasota, Florida, pp142-147, 1998.

Zhou, P., and Pycock, D., Robust Model-Based Boundary Cue Generation for Cell Image Interpretation. Paper presented at the British Machine Vision Conference 1995, BMVC'95, University of Birmingham, Birmingham, UK, 1995.

Zhou, P., and Pycock, D., Robust Statistical Model-Based Cell Image Interpretation. Paper presented at the British Machine Vision Conference 1995, BMVC'95, University of Birmingham, Birmingham, UK, 1995.