Machine Vision Techniques for the Evaluation of Animal Behaviour

by

Dr Derek Robert Magee

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy

The University of Leeds
School of Computing
October 2000

Abstract

Over the last few decades the nature of farming has changed dramatically with small labour intensive farms being replaced by large, highly automated farms. With this change in the way things are done comes a new concern for animal welfare. Traditionally a skilled stock-man would look after a relatively small number of animals and have much direct contact with them on a daily basis. On the modern automated farm a few people will look after several hundred animals and as such there is much less direct contact.

Within this thesis a toolkit of methods is presented which is suitable for the building of automated monitoring systems within a farm environment. Animal shape is modelled by a two dimensional hierarchical contour model and various models for animal dynamics are evaluated. A framework is proposed that combines these spatial and temporal models as a combined tracker and behaviour analysis system. A method is also presented for the identification of individuals from their characteristic markings which utilises the result from the animal tracker method. The methods presented have wider application than simply the field of animal behaviour monitoring and are applicable to other problem areas within machine vision and beyond.

Acknowledgements

I would like to thank my colleagues in the Leeds Vision Group (past and present) for many stimulating and profitable discussions, Professor Mike Forbes from the school of biology for his help with data gathering and other animal related matters and my supervisor Roger Boyle.

Declarations

Some parts of the work presented in this thesis have been published in the following articles:

D. Magee and R. Boyle. "Building Shape Models from Image Sequences Using Piecewise Linear Approximation". In *Proceedings British Machine Vision Conference*, volume 2, pages 398–408, BMVA, September 1998.

D. Magee and R. Boyle. "Feature tracking in real world scenes (or how to track a cow).". In *Proceedings IEE Colloquium on Motion Analysis and Tracking*, pages 2/1–2/7, IEE, May 1999.

D. Magee and R. Boyle. "Building Class Sensitive Models for Tracking Applications". In *Proceedings British Machine Vision Conference*, volume 2, pages 594–603, BMVA, September 1999.

D. Magee and R. Boyle. "Spatio-temporal Modeling in the Farmyard Domain". In *Proceedings IAPR International Workshop on Articulated Motion and Deformable Objects (AMDO'2000)*, Palma de Mallorca, Spain, pages 83–95, Springer-verlag, September 2000.

D. Magee and R. Boyle. "Detecting Lameness in Livestock using Re-sampling Condensation and Multi-stream Cyclic Hidden Markov Models" In *Proceedings British Machine Vision Conference*, volume 1, pages 332–341, BMVA, September 2000[1,2].

[1]Winner of Best Industrial Paper Award

[2]Invited for submission to IVC journal special issue

Contents

List of Figures

Chapter 1

Introduction

The work described in this thesis was motivated by the desire of animal physiologists to study the ‘behaviour’ of Livestock, in particular dairy cows. In this thesis the word ‘behaviour’ is used to mean any sequence of movements that conveys information about the internal state or intentions of the animal being analysed. Of particular interest are behaviours that relate to an animal’s state of health (e.g. limping represents lameness) or an animal’s state of being (e.g. specific behaviours that relate to contentedness/uncontentedness or being ‘on heat’). Limping is used as an example of such a behaviour within this thesis as there is a clear and simple relationship between limping and lameness.

The title of this thesis ‘Machine Vision Techniques for the Evaluation of Animal Behaviour’ implies there is more presented here than simply the solution to a practical problem. The various novel techniques presented may be taken as a set, or may be seen as a toolkit of methods for the analysis of the various animal behaviours interesting to animal physiologists and the farming community. These behaviours may be characterised by their stochastic nature and the generality of descriptions required to describe them. Within this thesis existing, as well as novel, methods are presented. Quantitative comparisons are made between the existing and novel techniques. The techniques also have wider application within the machine vision field, particularly in the tracking and analysis of other deformable objects such as humans and in medical image analysis.

1.1 Example Problem Domain

As a motivation for the development of the methods developed as part of this thesis, and to provide a test bed for these methods, a primary application domain was selected. This is by no means the only application domain investigated within this work (or the only application domain this work is applicable to), however it is widely used throughout the thesis and serves as a focus for the evaluation of the methods. The scenario involves dairy cattle walking, in near profile, from the milking parlour to an area containing food. This is a common scenario on dairy farms and simplifies the problem domain by restricting the observed shape of the animals in the 2D field of view. Figure 1.1 illustrates a typical input sequence.

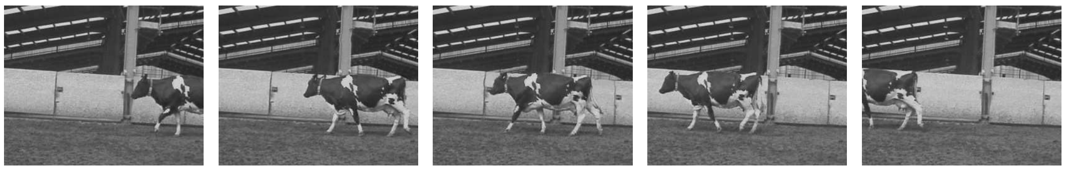


Figure 1.1: Typical Farmyard Scenario Used

A principal aim of the work carried out was to track the animal location and shape through a sequence such as the one illustrated in figure 1.1 with a precision such that meaningful information could be extracted by a higher level process about the animal's 'behaviour'. A secondary aim was to analyse this behaviour and be able to label or classify the behaviour. Lameness was chosen as an example behaviour that is easily classified by the limping motion of an animal.

1.2 Focus of this Work

The main focus of this work has been in automatically building models of an object's spatial (shape) and temporal (motion) characteristics. The purpose of such models is to encapsulate *a priori* information about these characteristics in such a way that this information may be used to extract these characteristics. At first glance this may seem like a 'chicken and egg' scenario in which it is required that a system knows what it is looking for before it can look for it, however this is exactly how human perception works under most circumstances. Humans find it harder to analyse and evaluate scenarios that have not been observed before and similarly it is much harder to design a system to evaluate observations with no *a priori* information. As such, this is the approach that is taken within this thesis. It is often impossible to make an *a priori* observation of every possible circumstance in 'real world' situations, and so object characteristic models must include the ability to generalise past observations to possible novel observations. When designing the models presented in this thesis a strong focus was put on including this power of generalisation. The other important criterion taken into account was the specificity of the models to the object class to be analysed. This is important as too general a model may be confused by noise, clutter or similar seeming objects or motions in a scene. There is always a trade off between the generality and specificity criteria within modelling schemes. The schemes presented within this thesis balance the necessary generality, in terms of representing 'real world' objects, with useful specificity, which allows object characteristics to be extracted easily and efficiently from highly complex data streams.

The spatial models presented within this thesis are based on a piecewise linear contour approximation. Contour models are a powerful abstraction of object shape as they are insensitive to lighting, colour and texture changes. Insensitivity to colour and texture changes is particularly important in the modelling of Livestock as markings differ from one individual to another. These models are also computationally efficient to work with in comparison with feature or texture based methods. The novel work presented within this thesis involves the development of a method for automatic collection of example data for model training (using a seeding process) and the development of a hierarchical 'learned' model scheme, which is an extension of point distribution model (PDM) [21] methods. The model seeding process involves manually or automatically determining the 'optimal' number of linear sections required to represent the perimeter of an object class or a subset of this class. In the context of this work an approximation is 'optimal' if the linear sections map to salient features of the object class or subclass. Once the seed or seeds are selected they are fitted to images by an energy minimisation process to build a data set for model training.

The hierarchical modelling scheme is based on a novel variation of the Point Distribution Model (PDM) [21] known as the 'Vector Distribution Model'. This models the variation of relative vectors between reference points (as shown in figure 1.2) rather than simply the positions of reference points. This scheme has advantages in terms of model compactness and the specificity of representation of rotational variation. This method is essentially a compromise between the original

PDM and the polar co-ordinates PDM of Heap and Hogg [44] in its ability to model rotational variation (modelled well by the polar co-ordinates PDM) and linear variation (modelled well by the original PDM).

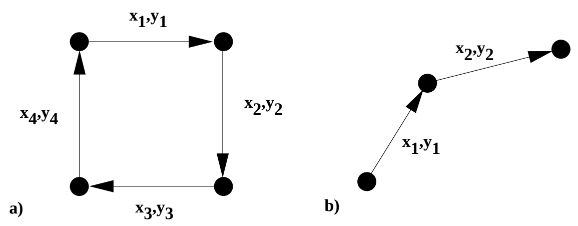


Figure 1.2: Relative Vectors Used in the Vector Distribution Model for Closed and Open Contours

The hierarchical model structure involves using multiple models, so object sub-classes with different numbers of salient features may be modelled. The variation within these models is separated into pseudo independent components (e.g. Front legs variation, Rear Legs Variation and Inter-animal variation), for each model, using a novel technique known as ‘Delta Analysis’ (a form of Linear Discriminant Analysis - See Appendix D). This allows the variation of the components to be modelled over time more easily. The model structure is illustrated in figure 1.3.

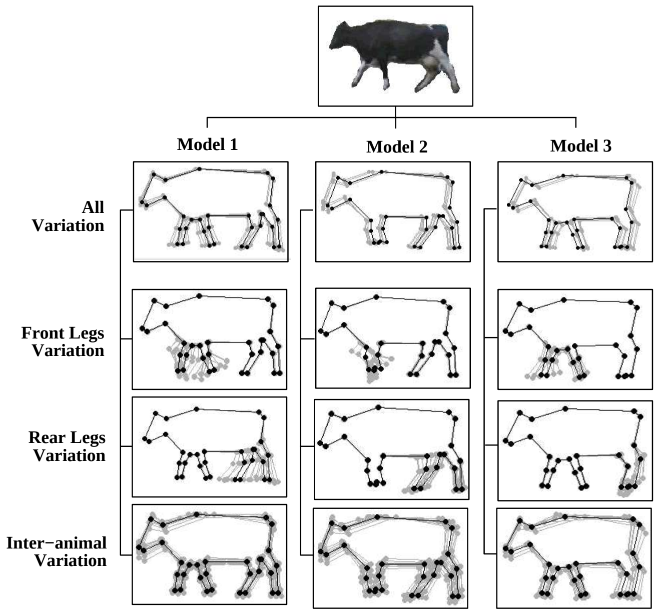


Figure 1.3: Hierarchy of Shape Models Used

This model structure, although apparently more complex than the standard PDM scheme, means the variation represented within each part of the scheme is simpler. This results in a highly specific model that can encapsulate the wide range of variation within the object class. Using this representation the information contained within the model about object pose is more easily available, which is important when building temporal models.

Two novel temporal models are presented within this thesis. The History Space Classifier (HSC) is an evolution of methods developed by Bulpitt *et al.* [13, 103]. These methods have three elements; a spatial classifier, a history function and a temporal classifier. The spatial classifier classifies the input at a given time instant, the results of which are fed into a history function that represents a complete history as a single point in a high dimensional space. The temporal classifier classifies this 'history space' representation to make a prediction of future system behaviour or a classification of past behaviour. The novel aspect of the HSC method is that it uses Principal Components Analysis (PCA) to reduce the dimensionality of the 'history space' allowing highly computationally efficient, low dimensional Gaussian mixture models to be used as the temporal classifier as opposed to the full dimensionality neural networks used by Bulpitt *et al.*. The saving in computational resources of this method over previous methods increases with the dimensionality of the history space, which is related to the complexity of the object modelling problem.

The second novel temporal model presented is the Multi-stream Cyclic Hidden Markov Model (M-SCHMM). This is a specific formulation of a Hidden Markov Model (see chapter 6 and appendix C) which has the cyclic hidden state architecture illustrated in figure 1.4 and supports multiple (but related) observation streams.

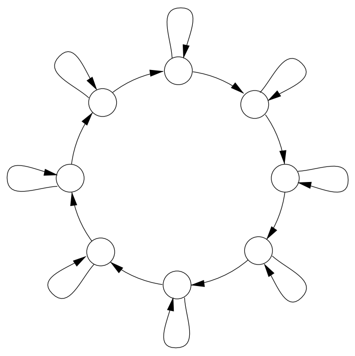


Figure 1.4: MSCHMM Hidden State Architecture

The MSCHMM is used in the Livestock analysis application to model the separate, but related, variation of the front and rear leg pairs. A novel method is presented for the selection of the number of hidden states and for the initialisation of transition and observation probabilities before the application of the Baum-Welch re-estimation procedure [85]. With the use of this training procedure these models are both highly specific to the class of temporal sequence modelled and general in their coverage of that class. The models are also highly computationally efficient, especially when predicting from a known hidden state, as demonstrated with their application within the Condensation / Re-sampling Condensation algorithms (see chapter 7).

To evaluate the object models developed within this thesis properly it was necessary to incorporate them into a practical framework that was able to extract information from a scene and classify this information in some way. The novel object tracking methods of 'Discrete Hypothesis Sampling' (DHS) and 'Re-sampling Condensation' (RC) were a result of this work. Both these methods are essentially stochastic search procedures that fit the spatial object models to an image sequence using the temporal models. The stochastic search procedures are initialised with a prior probability distribution for the spatial model parameters by predicting from past observations using the temporal models. A new image in the sequence is subsequently sampled (and re-sampled) based on this prior probability distribution and a posterior distribution for the spatial model parameters calculated. This process is illustrated in figure 1.5.

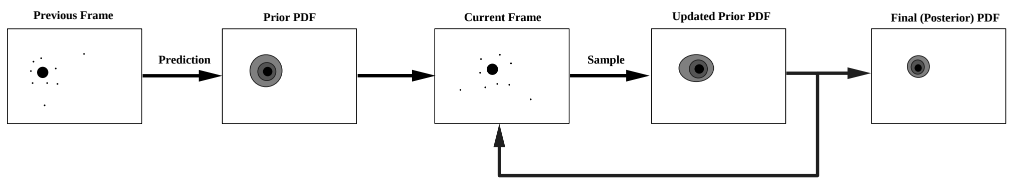


Figure 1.5: Sampling and Re-sampling an Image based on Prior PDFs

The differences between the methods are the model parameter probability distribution representation and the method of re-sampling the hypotheses and updating the probability distributions. DHS uses separate discrete probability distributions for shape parameters, scale and position, whereas RC uses a particle representation (as in the original Condensation algorithm [57]) which approximates to a joint probability distribution of these parameters. The probability update method for DHS involves iteratively re-sampling the image and updating the prior probability distributions, whereas RC involves only a single stage of re-sampling that combines the characteristics from successful samples in the initial sample set in a similar manner to a Genetic algorithm.

Multiple temporal models may be included in the RC scheme which allows this scheme to be used as a combined tracker and classifier meaning no higher level processing of the tracking results is required to perform analysis of tracking results. This scheme is highly efficient and suitable for ‘real time’ application.

The system architecture that ties together the different aspects of the research presented within this thesis is detailed in figure 1.6.

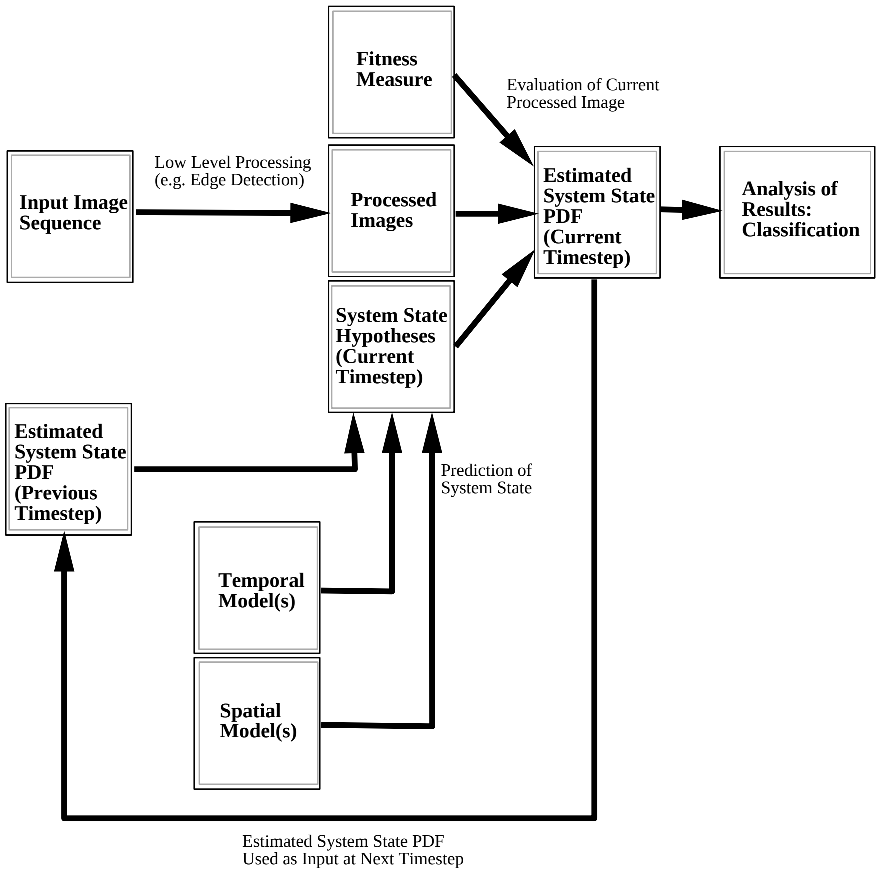

Fitness Measure
Evaluation of Current Processed Image
Input Image Sequence
Low Level Processing (e.g. Edge Detection)
Processed Images
Estimated System State PDF (Current Timestep)
Analysis of Results: Classification
System State Hypotheses (Current Timestep)
Estimated System State PDF (Previous Timestep)
Prediction of System State
Temporal Model(s)
Spatial Model(s)
Estimated System State PDF Used as Input at Next Timestep

Figure 1.6: System Architecture Used Within this Thesis

1.3 Overview of Thesis

In this introduction a general introduction and overview of the subjects covered within the thesis has been presented. Chapter 2 provides a review of previous work and background material. The remaining chapters describe the original work of the thesis and include the results of various evaluation experiments carried out. This work is organised as follows :-

- **Chapter 3**
  A novel method for data collection for contour models from image sequences is presented. This is evaluated on the contours of various real and synthetic objects.
- **Chapter 4**
  A novel variation of the Point Distribution Model (PDM) based on a relative vector between reference points contour representation is presented. This is evaluated against other PDM formulations.
- **Chapter 5**
  ‘Delta Analysis’, an alternative method to Linear Discriminant Analysis for the separation of inter- and intra-class variation within a multivariate data set, is presented. The application of this method to the separation of independent components in object models is discussed and a quantitative comparison of these two methods is made.
- **Chapter 6**
  Two temporal modelling schemes, ‘History Space Classifiers’ and ‘Multi-stream Cyclic Hidden Markov Models’, are presented. A quantative evaluation is given against the benchmark of the Markov Chain.
- **Chapter 7**
  Two object tracking schemes, ‘Discrete Hypothesis Sampling’ and ‘Re-sampling Condensation’ are presented. A quantative evaluation of these schemes against the Condensation algorithm [57] is presented. The idea of including multiple temporal models within the Re-sampling Condensation framework is presented and it is shown how this may be used as a combined tracker and classifier.

- **Chapter 8**

  A novel method is presented for the identification of individual animals from their markings. This utilises the results from the object tracking schemes described in chapter 7. The method is evaluated with a realistically sized data set.

Finally conclusions and future work are discussed in chapter 9.

Chapter 2

Modelling and Analysis of object motions and behaviours: A review

Early applications of machine vision were concerned with problems involving single images only such as character recognition, industrial inspection and face recognition. As computing power has increased applications involving streams of images have been developed. The objective of these methods is the analysis of objects in a scene or analysis of the structure of the complete scene. In the analysis of objects the aim is to extract information about object position (or relative position), pose and spatial state (for articulated or deformable objects). Examples of object analysis are human tracking and analysis [49, 2, 15, 18, 57, 78, 62, 118, 88, 52], animal tracking and analysis [103, 120, 73, 94], vehicle tracking and analysis [47, 34, 54] and the tracking and analysis of human internal anatomical structures [60, 55, 14]. Methods for the analysis of the complete scene are generally for autonomous vehicle [110] or robot [24] navigation and control or scene reconstruction (the so called ‘structure from motion’ problem) [53]. This thesis deals with the former problem of object analysis and, as such, the review in this chapter deals with methods related to solving such problems. The review is divided into three sections ‘Shape and Appearance Modelling’, ‘Temporal Models and Prediction’ and ‘Object Tracking’. This is a somewhat artificial separation as there are many methods that fall within more than one of these categories, however this separation serves to highlight the main problems that need to be solved in object analysis applications.

2.1 Shape and Appearance Modelling

The aim of modelling shape or appearance is to encapsulate *a priori* information about an object that may be present in a scene of interest. These 'spatial' models have a dual purpose; firstly to aid the location of the object in a scene and secondly to parameterise the various aspects of model variation (position, pose, orientation etc.). It is often possible to locate an object in a scene with a very low level model (e.g. pixel level models) or by modelling the scene rather than the object (e.g. background subtraction), however this section deals only with higher level models that relate more directly to the object to be located and analysed. Only such high level models offer useful parameterisations of complex model variation which may be used in object behaviour analysis.

2.1.1 Learned vs. Hand Crafted Models

One principal division in models of shape and appearance is the division into 'learned models' and 'hand crafted models'. Learned models are based on a training set which is usually formed from images of objects that have been pre-processed in some way. This pre-processing may take the form of a manual or automatic cropping (e.g. Eigenfaces [97, 109]) or a more abstract operation, for example to extract a contour representation (e.g. Bregler and Omohundro's lip models [11]) or a set of reference points in 2D or 3D (e.g. the point distribution model [21, 45]). Learned models usually consist of a statistical representation of this pre-processed data set which can classify the degree to which a novel example fits this set and may (in the case of 'generative models') be able to reproduce novel examples of the object class. One widely used statistical technique is Principal Components Analysis (PCA), also known as Eigen analysis or the Karhunen-Loeve Transform (see appendix D). This is a data dimensionality reduction method that makes the assumption that the data is a linear combination of orthogonal 'Eigenvectors'. The variance of the data set along each Eigenvector is maximised sequentially by rotating the reference co-ordinate system such that the variance along successive Eigenvectors is monotonically decreasing. In many cases the variance along the least significant Eigenvectors is negligible and these vectors may be discarded, reducing the dimensionality of the data set and increasing the specificity of the representation. PCA has been used to model pixel intensities over an image [97, 109], reference point locations [21, 45] and combined point and intensity data sets [18]. Cootes *et al.* [21] put an upper limit on the Mahalanobis distance (distance normalised by variance along each component) of a novel shape in the 'Eigenspace' spanned by the Eigenvectors for classification as a member of the class. This effectively defines a hyperellipsoid in Eigenspace that represents the set of valid class members. Bregler and Omohundro [11] take a slightly different approach and combine K-means clustering with local PCA to form a set of surfaces in the model parameter space which represent valid class members. Heap and Hogg [46] combine the methods of Cootes *et al.* [21] and Bregler and Omohundro [11] to form a set of surfaces in Eigenspace. PCA has been the dominant statistical method for learned models in machine vision over the last decade, however among the alternatives are simple statistics such as means and co-variances (e.g. [74]), Gaussian mixture models (e.g. [20]), Support Vector Machines (e.g. [80]), neural networks / self organising maps (e.g. [121]), Hidden Markov Models (e.g. [87]) and Linear Discriminant Analysis (e.g. [31]). Of these methods Linear Discriminant Analysis (see appendix D) deserves special mention as, in addition to producing a statistical representation of a data set, variation separation is performed. In the example of Active Appearance Models variation is separated into various components (pose, person ID, expression and illumination) by grouping the training data into classes. This separation of variation increases the ease of interpretation of the model parameters and has been shown to increase the robustness and efficiency of object location [31] in some cases.

2.1.2 Hand Crafted Models

Hand crafted models are generally based on simplifications of real world objects and are often constructed from simple primitives such as simple geometric shapes in 2D or 3D. Hogg [49] defines an edge based 3D model of a person from interconnected cylinders of various sizes and diameters. Sidenbladh *et al.* [95] use a similar model to Hogg, based on spheres and cylinders, which includes surface texture mapping to allow model fitting from image texture rather than simply edges. Ju *et al.* [64] model humans as 'cardboard people'. These are defined with interconnected planar patches. Terzopoulos and Metaxas [106] present a 3D deformable model, the 'Deformable Superquadratic', which incorporates the global shape parameters of a superellipsoid with the local degrees of freedom of a spline. DeCarlo and Metaxas [26] blend simple 3D shape primitives such as cylinders and spheres together to produce more complex deformable models than possible previous work (e.g. with Deformable Superquadratics [106]). Worrall *et al.* [116] define a rigid 3D model of a vehicle based on connected straight line sections.

The hand crafted models mentioned in this section represent useful abstractions of the true object to be analysed as primitives often map closely to actual physical features (e.g. arms, legs etc in the human example). This is useful in the analysis of these physical features. Some hand crafted models incorporate the potential to deform (e.g. Terzopoulos and Metaxas' deformable superquadratics [106]). Such models have the potential to map closely to real world objects at the expense of specificity. Non-deforming models have a high degree of specificity but can only map closely to real world objects if the topology of these objects is highly constrained.

2.1.3 Contour and Surface Models

Contour and surface models represent a useful abstraction of certain data classes in which intensity/texture information is unreliable or computationally expensive to work with. Contour models may be either learned or hand crafted. Perhaps the earliest contour model to make an impact (apart from the Hough transform) was Kass *et al.*'s active contour model or 'snake' [66]. This was based on a spline representation that was fitted to single images or successive images in a sequence. The only *a priori* information contained in this model was the information that the object to be located exhibits a continuous open or closed contour with a constrained maximum curvature. This model is highly susceptible to clutter and requires fairly precise initialisation. Yuille and Hallinan [122] introduced the idea of using parameterised contour templates and, as such, incorporating *a priori* information about a specific object to be located. The template used in this model formulation was fixed (the parameterisation being based on transforms of the template) and as such the variation in possible shape was limited. Blake *et al.* [8] incorporated templates into the active contour model algorithm to produce a snake that was drawn towards a particular template shape in the absence of strong image information to the contrary. For certain classes of object a single fixed template (as used by Yuille and Hallinan and Blake *et al.*) is insufficient to represent the complete range of variation within the object class. Cootes *et al.*'s Point Distribution Model (PDM) [21] and Bregler and Omohundro's surface learning [11] (both described previously in section 2.1.1) attempted to address this problem using a descriptive model of shape 'learned' from a set of examples using principal components analysis. Since the publication of these papers, there has been little of significance published on the fundamentals of contour modelling with the exception of a selection of papers on improving the specificity of the PDM (e.g. [98, 99, 46, 20]), although there has been significant advances in the fitting of these models to images (this is dealt with in section 2.3). This is fundamentally attributable to two reasons; firstly the current contour modelling schemes are adequate for current purposes (contour modelling algorithms are not generally considered to be the 'weak link' in tracking applications) and secondly the recent interest in non-contour based models (such as models of intensities) brought about by the increased computational power available in recent years. Despite the relative lack of development of contour models in recent years they are still widely used in object tracking and other machine vision applications due to their computational efficiency in evaluation and the robustness of fitting algorithms under many circumstances.

Surface models are essentially a 3D version of contour models, in fact some of the methods mentioned previously may be extended to 3D surface modelling. Sclaroff and Pentland [93] present a method of modelling surfaces using the finite element method. This is used in the analysis of range data of heads. Other surface modelling algorithms have been mentioned in section 2.1.2 such as Terzopoulos and Metaxas' deformable superquadratics [106] and DeCarlo and Metaxas's Blended deformable models [26].

2.1.4 Image and Eigenimage Based Methods

An alternative to modelling objects using contours is to model image intensities directly. Black and Yacoob [7] use sets of image regions which may undergo rigid or non rigid transformations from one frame to the next to track human faces. Principal Components Analysis (PCA) has been applied to greylevels in face and other images to obtain a compact parameterisation of these complex data sets [97, 109]. This is achieved by forming a (high dimensional) vector of all the pixel grey levels in each training image. Turk and Pentland [109] highlight the practical problem of finding the Eigenvectors of a high dimensional problem and present a method for reducing the problem to the order of the number of samples (which is usually much lower than the number of pixels per sample in these cases). In this method the number of Eigenvectors found is equal to the number of data items rather than the dimensionality, however this is not a problem as the training data set can be described exactly using these Eigenvectors. If the number of data items is large this may be reduced using clustering algorithms. Black and Jepson [5] have used PCA based Eigenimage/Eigenface models in object tracking applications. Standard Linear Discriminant Analysis (see appendix D) does not apply well to raw image data as the intra-class covariance matrix is often singular due to the sample size being much smaller than the dimensionality. Belhumeur *et al.* [4] perform PCA followed by Linear Discriminant Analysis (See appendix D) in a method known as 'Fisherfaces' to include class separation in their Eigenface representation.

2.1.5 Appearance Models

The previous two sections have dealt with models based on shape (contours) and colour or intensity. Objects in the real world have properties relating to both shape and colouring (there is in fact a high degree of independence between these properties in many objects). In recent times there have been several models proposed that combine both these properties which have proved useful in machine vision applications. Cootes *et al.*'s Active Appearance Model [18] combines 2D shape (based on a triangulation) and intensity information in a single PCA based model. Intensity information is represented as an Eigenimage (see section 2.1.4) in a normalised co-ordinate frame. The normalised intensity image is mapped into the real image frame by warping the normalised image by the triangulated shape representation. The variation in the shape representation is also modelled using PCA and the co-variance of the two data types modelled by a further application of PCA. La Cascia *et al.* [16] present a similar method in which a normalised intensity model is warped to a 3D surface (a cylinder). Basu *et al.* [1] model the shape of human lips using an elastic mesh based on the finite element method. The possible deformations of this mesh are learned using PCA and the model is fitted to data using colour information using prior probabilities from Gaussian mixture models of lip colour and non-lip colour. Graf *et al.* [40] present a model of the face that is based on 3D planar patches and a set of example bitmaps for these surfaces, however this model is only used for sequence generation rather than information encapsulation for monitoring novel sequences. Sidenbladh *et al.* [95] map texture onto a 3D model of a person constructed from cylinders from one frame to the next, however no *a priori* model of texture or intensity is used in this scheme. The approach taken by Sidenbladh *et al.* is sensible as the intensity (or colour or texture) of point on a 3D object is subject to change with position due to illumination (and other factors such as a human changing their clothes). Changes may be small between frames and, as such, this method does not need to deal with these changes. The methods of Cootes *et al.* and La Cascia *et al.* deal with this problem by modelling the range of possible intensities using PCA. Appearance based methods are currently relatively computationally expensive to use in comparison to alternative methods, however with the advent of more powerful computers and special hardware (e.g for texture mapping) these methods provide a good solution in some application domains. The primary application domain of such models is in modelling human faces as both intensity and shape information are complex and hard to extract on their own. Other application areas include medical imaging [17] and people modelling [95], however face modelling remains the primary application domain of such techniques.

2.1.6 Feature Based Models

An alternative to using methods that directly model object attributes such as shape or appearance is to model the results of 'bottom up' (model-less) data processing operations on particular features of interest or on an entire image area. Wren *et al.* [117] represent a human body as a set of 'blobs' with coherent colour properties. Rigoll *et al.* [87] model a set of features produced using a discrete cosine transform (DCT - as used in JPEG image compression) using a pseudo-2D Hidden Markov Model. Wavelet transforms have become a popular form of feature extractor (especially for images of human faces). Wu and Toyama [119] model the results of Gabor wavelet transforms on faces with a single Gaussian for each point at a series of poses to detect facial pose. Lyons *et al.* [71] also use Gabor wavelets, however only the results of transforms at the nodes of an elastic grid are modelled (using Linear Discriminant Analysis). This system is used for face classification.

2.2 Temporal Models and Prediction

There is much interest within the field of machine vision in the modelling and prediction of time dependent (often probabalistic) processes. Examples of such processes are human activity, animal activity, vehicle activity (including traffic flow) and the activity of complex or unknown machines or robots (or the interaction of any of these things). The aim of such modelling is to encapsulate enough information about a system to make either a classification of observed behaviour or a prediction about future behaviour. Simple models, such as the Kalman filter [65], may be based on assumptions about system behaviour (the system may be approximated by a linear stochastic difference equation in the case of the Kalman filter, see section 2.3.1), however these assumptions are often over simplistic and, when broken, lead to inaccurate classification or prediction. If the temporal properties of a system are known these may be encapsulated as a set of rules, however this is often not the case and the dynamics must be ‘learned’ by observing the behaviour of the system. The rest of this section is devoted to methods that have been employed in modelling such systems.

2.2.1 Markov Chains

Markov chains (a form of directed graph) are widely used in many areas of computer science and mathematics as a temporal model or predictor. They are used to model probabalistic processes (often machines) where the state of the process may be described by one of a fixed number of tokens or labels. The widespread use of Markov chains is generally due to their simplicity and thus speed of evaluation. Markov chains are based on a set of discrete states. A matrix of transition probabilities is used to represent the probability of a system being in a particular state at the next time step given a knowledge of the state of the system at a finite number of time steps before this. For a ‘first order’ Markov chain the transition probabilities depend only on the previous state as illustrated in figure 2.1.

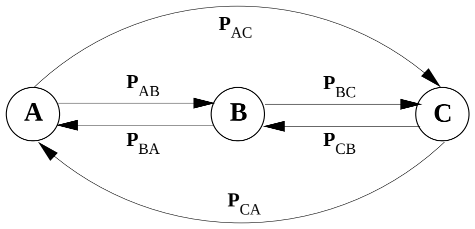


Figure 2.1: A Simple ‘First Order’ Markov Chain with 3 States

2.2.2 Variable Length Markov Models

Variable length Markov models (VLMMs) are a form of Markov chain (see previous section) where the order (length of history used) of the prediction is dependent on the state to be predicted from. The order at each node is determined from the training data using a cross-entropy measure (the Kullback-Leibler divergence [90]). Essentially this means that if prediction using a longer history produces a significantly different result than using a shorter history (on average) the longer history is used. Theoretically this results in an optimally compact model that is efficient to evaluate. VLMMs can as such model both short and long term activity, although in practice an upper limit is put on the order for memory and computational efficiency reasons. These models have been used extensively in speech and language applications (e.g. [42, 50]) but have only recently been introduced into the machine vision field. Galata *et al.* [35] use VLMMs to model short and medium term human activity (such as aerobics) using multiple VLMMs to model atomic actions. An additional VLMM is used to model the transition between atomic actions in this scheme.

2.2.3 Hidden Markov Models

Hidden Markov Models (HMMs) extend the concept of the Markov chain by assuming that observations result from some underlying process. In many cases the exact nature of this process is not observable (or 'Hidden'), for example speech, however the resulting observations (sound in the case of speech) are. HMMs have been widely used in the fields of speech recognition and language modelling for some time and recently have become popular tools within the field of machine vision. American Sign Language recognition using HMMs has been performed by Metaxas [76] using 3D deformable models and by Starner and Pentland [100] using information from a simple 2D hand tracker. Kettnaker and Brand [67] model Traffic flow using optical flow and HMMs and Gong [37] uses HMMs to predict the movements of airport service vehicles for use in an adaptive attention tracking system. Brand [9] uses HMMs to model the relationship between a sequence of 2D silhouettes and the 3D pose of a person

HMMs model an underlying process using a first order Markov chain and the relationship between the process and the observations by a probability distribution (either discrete or continuous). Figure 2.2 illustrates this by modelling the structure of the word 'hello' using one state for each syllable.

This model of the word 'hello' illustrates two important features of the HMM. Firstly, although the Markov chain is only first order, a temporal history is encoded by limiting the state transitions possible. In this example state A will always occur before state B, thus encoding the temporal ordering of the two syllables in the word. Secondly, if the four possible permutations of syllables

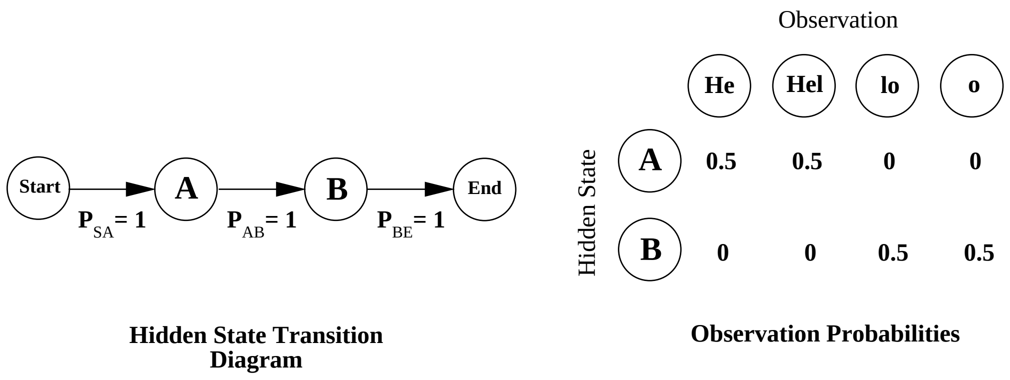


Figure 2.2: Modelling the word 'Hello' using a Hidden Markov Model

are examined ('He-lo','He-o','Hel-lo' and 'Hel-o'), it can be seen that an invalid combination is possible ('He-o'). If the HMM were to be used for recognition this would not be a problem as there is no other word in the English language that consists of these syllables, however if the HMM were to be used for speech generation this would present a problem. HMMs intended for sequence generation or prediction in general require more complex state transition architectures than HMMs intended for recognition only. Figure 2.3 illustrates a typical 'left-right' architecture used in real speech recognition applications.

$$\begin{tikzcd}
\text{Start} \arrow[r] & A \arrow[r] \arrow[rr, bend left] & B \arrow[r] \arrow[rr, bend right] & C \arrow[r] & D \arrow[r] & \text{End}
\end{tikzcd}$$

Figure 2.3: A 'Left-right' HMM Architecture as used in many Speech Applications

Rabiner in his excellent introduction to HMMs [85] lists three problems that need to be solved in order to use HMMs;

1. Problem 1 is the evaluation problem, or how the probability that an observed sequence was produced by a given HMM may be calculated.
2. Problem 2 is the recovery of a 'correct' sequence of hidden states for an observation sequence.

3. Problem 3 is the training problem, or how to optimise a set of HMM parameters so as to best describe how an observation sequence (or sequences) come about.

In addition to these there is a fourth problem; prediction. The problem is, given some knowledge of the system in the past / present, how can we predict the future behaviour of the system. These problems are addressed in appendix C.

2.2.4 History Representation Classification Schemes

Markov based methods (see previous sections) use a finite (and usually fixed length) temporal history to predict future observations with each observation having an equal effect on the prediction. An alternative to this is history representation classification schemes that represent a complete history as a fixed size vector that can be thought of as a point in a high dimensional space. A generic schematic of such methods is given in figure 2.4.

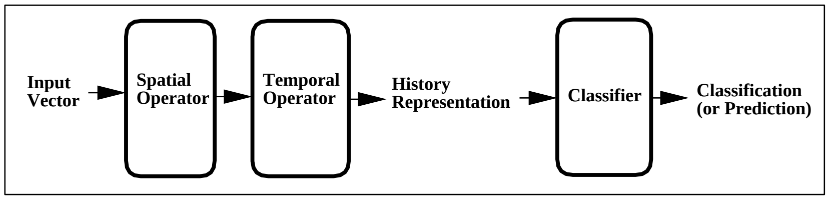


Figure 2.4: A Generic History Representation Classification Scheme

The spatial operator works only on the input at the current timestep. This may take the form of a spatial classification or simply a pre-processing step (or a combination of these). The purpose of this stage is to transform the input vector (which may be an image or set of model parameters) into a form that may be used by the temporal operator. The temporal operator encodes the output of the spatial operator over time to produce a ‘history representation’. The fixed size of the history representation vector means the information stored from each timestep cannot be equal. The solution to this is to give the most weight to the most recent observations with increasingly less weight to past observations. An example of such an operator is a set of leaky integrators as illustrated in figure 2.5.

One of the earliest implementations of such a scheme was by Bulpitt and Allinson [13]. This

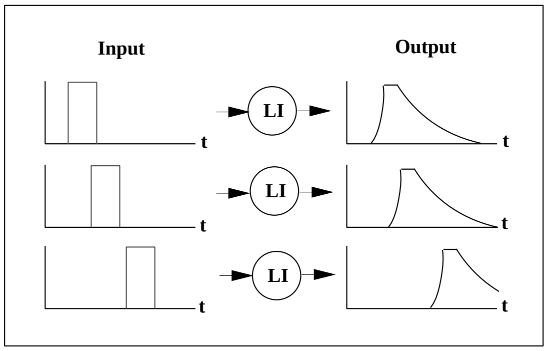


Figure 2.5: Encoding History Using Leaky Integrators

used adaptive resonance theory (ART) neural networks as the spatial operator (used as a spatial classifier) and as the history representation (temporal) classifier. The parameters of these networks are learnt as a pair in an unsupervised manner (there is feedback between the two networks to allow this). The temporal operator is essentially a set of leaky integrators as described previously. The input to the temporal operator is only activated for one node at a time (this is known as the ‘Winner takes all’ approach). This scheme was applied to the recognition of human motion using the location of moving light displays as an input. A simplified version of this scheme using separately trained spatial and temporal neural networks, presented by Sumpter and Bulpitt [103], was used in the modelling and prediction of the behaviour of a flock of ducks when confronted with a robotic sheepdog.

Davis and Bobick [23] apply a simpler technique to raw image sequences of human activity. A ‘Motion History Image’ is formed that has intensity related to how recently motion occurred at that pixel. This is constructed using a linear time decay operator on pixel intensities which is set to maximum intensity if motion occurs at the current timestep. These images (and ‘Motion Energy Images’ that represent where the motion has occurred) are matched with stored templates to classify the motion. The spatial operations are limited to pre-processing operations (such as sub-sampling) in this scheme and provide no classification. The scheme is illustrated in figure 2.6.

Johnson and Hogg [63] use a vector quantisation process similar to a self organising map [68] to produce a set of prototypes for use with the spatial operator. A vector of distances from these prototypes is formed as the output of the spatial operator. The temporal operator limits the decay rate of the output of the spatial operator so as to encode a history. It should be noted that this

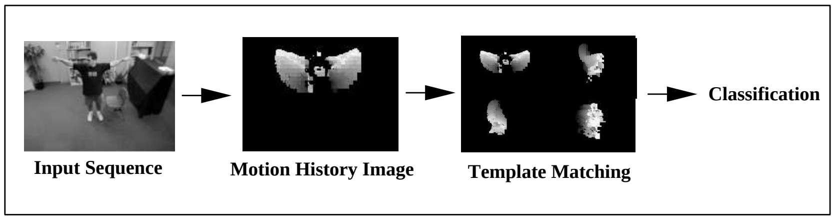


Figure 2.6: Motion Classification Using Motion History Images

works on all components of the vector (i.e. it does not take the ‘Winner takes all’ approach used by Bulpitt *et al.* [13, 103]). This approach is less prone to errors caused by erroneous spatial classification than the ‘Winner takes all’ approach. The temporal classifier for this scheme is again based on a vector quantisation (of the history representation space). A first order Markov chain is built which models transitions between prototypes within the history representation space.

2.2.5 Trajectory Modelling Schemes

Various approaches to temporal modelling have employed forms of trajectory modelling in abstract configuration spaces. These spaces may represent points in space, model parameters, image intensities (or the results of spatio-temporal processing on these) among others. In some cases time is included as a dimension of this space, resulting in a time dependent model, and in other cases time is omitted, resulting in a completely time independent model. In most cases it is desirable to include some representation of time in the model, even if this is only preserving temporal coherence, however an element of time independence is also required to incorporate the natural time variation in many interesting temporal actions (e.g. human gait, sign language, lip movements etc.). Some interesting solutions have been suggested to the problem of incorporating time into trajectory models.

Campbell and Bobick [15] relate pairs of model parameters representing the pose of a ballet dancer using a cubic polynomial. These ‘pair predictors’, which are a binary function of time (based on a distance threshold), are logical ANDed together to give a yes/no classification for a particular temporal action. Niyogi and Adelson [79] fit ‘snakes’ (cubic B-splines) to a 2D (X position vs. time) space to classify individual gait. A nearest neighbour recognition scheme is used to classify a novel sequence as belonging to a particular person. Wilson and Bobick [115] fit a ‘principal curve’ in a configuration vs. time space which is collapsed to simply a configuration space for operation. The curve is quantised and graph (Markov) based methods and dynamic programming used to compute ‘average combined membership’ for a gesture over a novel sampled gesture. Murase and Sakai [77] compute the difference between an input and reference vector sequence in a truncated Eigenspace based on background differenced images. This difference is minimised over a range of linear time stretches and shifts to incorporate temporal variation. Huang *et al.* [51] extend the method of Murase and Sakai using a combined Eigenspace and Canonical (Linear Discriminant Analysis) transform to improve class separation. In this scheme classification may be performed using accumulated distance from the centroid of the training data for a particular class (effectively ignoring temporal information) for a limited data set due to the wide separation of data sets in this representation space. Later work includes optical flow images and as such incorporates a temporal aspect into the model [52]. This combined model is observed to outperform the model based on intensities alone. Sidenbladh *et al.* [95] define a normalised (in time) walking cycle over an arbitrary (fixed) time period. The variation of the joint angles of a person is learned from data set of these normalised cycles using multivariate principal components analysis.

2.3 Object Tracking

Object tracking is essentially extracting information (such as position, shape or pose) about an object from an image sequence. ‘Tracking’ implies prior information (from previous frames) is used to extract information from subsequent frames (if this is not the case the task is simply a set of individual object extraction problems). Prior information may take the form of a single hypothesis about object state (e.g. position, size, shape etc.) that is propagated from one frame to the next (e.g. in the Active Shape Model [19] or Active Appearance Model [18]) or a probability density function (PDF) of possible states (e.g. Kalman filters [65] or the Condensation algorithm [57]).

A priori information about object characteristics may be used in object tracking to improve robustness and efficiency. Such schemes are referred to as ‘Model Based’ tracking schemes. In such schemes mathematical models are built of an object’s spatial (size, shape etc.) and temporal (typical motion) characteristics. Tracking using a model based scheme is a matter of fitting the model to the image sequence and extracting the model parameters (the so called 'top down' approach). Perhaps the first example of such a scheme is Hoggs 'Walker' model [49] in which a 3D model of a person constructed from a set of cylinders is fitted to an image sequence using edge information and a exhaustive search procedure. Recently Sidenbladh *et al.* [95] extended this method to work with image appearance (colour and texture) rather than edge information and employed a more efficient search strategy based on the Condensation algorithm [57]. Examples of 2D models such as Kass' 'Snakes' [66], Cootes and Taylors 'Point Distribution Models' and Cootes *et al.*'s 'Active Appearance Models (among others) have been described previously in section 2.1.

An alternative to model based methods are methods that use spatial or temporal image processing functions or statistical methods to extract information about movement (e.g. optical flow), colour or texture (or changes in these over time) from the image sequence at the pixel level. Areas of the image can then be grouped and associated with objects from frame to frame using correlation methods based on spatial or temporal coherence (this is referred to as the 'bottom up' approach). McKenna *et al.* [75] use colour and intensity gradient information to separate moving objects from a complex background. Colour histograms and spatial position are used to identify separate objects. Polana and Nelson [83] use optical flow with a constant velocity assumption to help cluster related areas of motion. Davies *et al.* [22] use a temporal wavelet filter to extract moving objects and a Kalman filter to aid correlation. Other applications of low level approaches in combination with Kalman filtering are Rosales and Sclaroff [91] who perform background differencing based on the mean and variance of a 3D colour space and Stauffer and Grimson [101] who model the background pixels as a set of Gaussian mixture models in RGB colour space, classifying any pixel that does not fit the background model as part of a moving object. Raja *et al.* [86] also use Gaussian mixture models (in Hue/Saturation space) but model both the foreground and the background for tracking skin coloured objects such as hands and faces. Wren *et al.* [117] use a single Gaussian to model the background and group foreground pixels as 'blobs' using colour statistics.

A large number of methods cannot be exactly classified as 'top down' or 'bottom up' as they are hybrids between these two approaches. Such methods extract 'medium level' features (such as blobs) using pixel level methods and fit a high level model to these features. Ju *et al.* [64] use optical flow to fit a 2D 'cardboard person' to an image sequence. Haritaoglui *et al.* [43] use a similar model but extract pixel regions using background intensity statistics and image processing to perform background subtraction. Intille *et al.* [56] extract blobs by background subtraction and use template matching to extract higher level information in a constrained ('Closed-World') environment. Brand [9] also uses background subtraction after which a 3D model based on a Hidden Markov Model is fitted. Bregler [10] proposes a hierarchical probabalistic framework that links pixels, blobs and linear dynamics using the Expectation Maximisation (EM) algorithm.

It was stated previously that object tracking involves propagating prior information about objects states (position, shape, scale etc.) from past frames to subsequent frames. Object tracking methods may be divided into two groups; 'Local Search Methods' where a single hypothesis is propogated and 'Global Search Methods' where a probability density function (PDF) of hypotheses is propogated. Local search methods iteratively refine the initial guess propogated from the previous frame to produce a single hypothesis about object state in the current frame. Global search methods sample the space of possible solutions stochastically using the 'prior' PDF propogated from previous frames. This sampling may be iterative as in the Markov Chain Monte Carlo method [25] or as a single sample set as in the Condensation algorithm [57]. Some examples of Local and Global search methods are presented in the following sections.

2.3.1 The Kalman Filter and the Extended Kalman Filter

The Kalman filter [65] is has been widely used in machine vision applications for some time (e.g. [33, 105, 8, 2, 101]). Kalman filters provide a 'recursive solution to the least-squares method' [114] which may be used to make a probabalistic prediction about the future state of a linear system described by the linear stochastic difference equation:

$$x_{k+1} = A_k x_k + B u_k + w_k \tag{2.1}$$

with a measurement process modelled by:

$$z_k = H_k x_k + v_k \tag{2.2}$$

where:

x_k	$=$	The state of the system at time k
u_k	$=$	The control input

w_k	$=$	The process noise
v_k	$=$	The measurement noise
A_k	$=$	Matrix relating state at time step k to state at step k+1 in the absence of either a driving function or process noise
B	$=$	Matrix relating the control input to the system state
H_k	$=$	Matrix relating the measurement to the system state

The Kalman filtering method produces a prediction of the actual state of the system based on the noisy measurement (z_k) and a 'prior' estimate of the system state. This takes the form of a single state prediction and an estimate of the error covariance. An exact formulation of the Kalman filtering method is presented elsewhere (e.g. [114, 65]) however the basic principal is presented in figure 2.7.

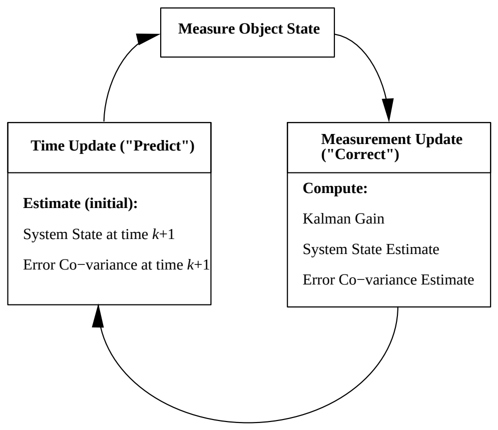


Figure 2.7: The Kalman Filter

The 'Kalman Gain', computed in the 'Measurement Update' part of the cycle in figure 2.7, is calculated from the initial measurement error co-variance estimate (calculated in the 'Time Update' part of the cycle) and is used to weight the relative contributions of the prediction and the measurement in the final estimate of the system state. Effectively this means that if the predicted measurement error co-variance is low, the measurement (z_k) is 'trusted' more. An estimate of the measurement error co-variance and the process noise co-variance must be provided for the Kalman filter to operate. These are usually estimated 'offline' from a set of training data, although this is not an entirely deterministic procedure.

Many systems are not well approximated by the linear stochastic difference equation 2.1 which has led to the development of the Extended Kalman Filter (EKF). The EKF ‘linearises about the current mean and co-variance’ [114] and represents a non-linear process as a piecewise linear approximation.

Whether the Kalman filter (or EKF) is a local or global search method (or neither) depends on the measurement process. In its simplest form the measurement process may be completely independent of the estimated system state. In this case the Kalman filter is simply a noise reducing post-processing stage to another method and is not an integral part of the tracking algorithm. Alternatively the estimated system state may be used as an initial estimate for a local search scheme (e.g. Baumberg and Hogg’s Active Shape Model scheme [2]). The estimated system state and error co-variance could also be used as a prior PDF for a global search scheme, although no example of this has been found by the author.

2.3.2 The Active Shape and Appearance Models

The active shape model (ASM) [19] and active appearance model (AAM) tracking schemes [18] are both local search methods, however they differ in the fact that ASM is local in the image domain where as the AAM is local in the model parameter domain. The ASM fits a point distribution model (PDM : see section 2.1.3) by searching (with limited range) for intensity changes (edges) along normals to an initial estimate as illustrated in figure 2.8.

The search procedure illustrated in figure 2.8 produces a candidate solution. This candidate solution is projected into the PDM parameter domain (Eigenspace) and the Mahalanobis distance of the shape from the mean is calculated to determine if it is a ‘valid’ shape. If this distance is greater than a specified threshold the shape parameters are scaled linearly by a single factor to ensure the candidate shape is valid. The new approximation is then calculated as a weighted average of the previous estimate and the normalised candidate solution. This process is repeated iteratively until convergence. The weighted sum ensures only small changes in shape occur in each iteration which allows reference points to move in directions not normal to the initial contour estimate as the gradient of the contour changes over the iterations. The algorithm has a certain resistance to noise and clutter as possible shapes are constrained, however in extreme situations convergence to

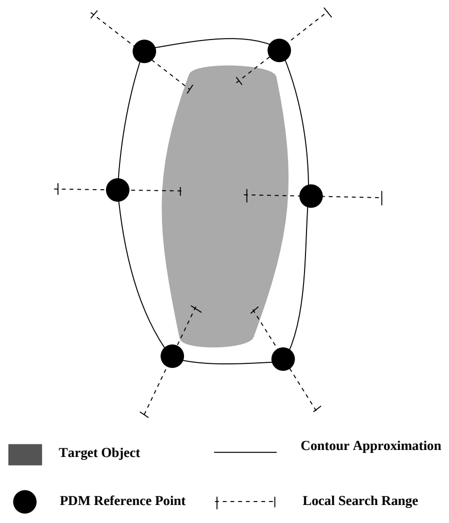


Figure 2.8: The Active Shape Model Local Search Procedure

an incorrect locally optimal solution is possible.

The AAM fits a model based on shape and intensity (described in section 2.3.2) to an image using pixel intensity information. For each iteration of the local search a complete residual error image is computed (in the shape normalised image domain) as the difference between an image reconstructed from the model and the actual image. The relationship between the residual error image and the model parameter changes required to make the model fit the image is assumed to be linear and is learned offline by linear regression from a set of known model parameter displacements. The model fitting algorithm is then simply a matter of calculating the residual image and performing a matrix multiplication as given in equation 2.3.

$$\delta c = \mathbf{A}\delta\mathbf{I} \tag{2.3}$$

Where:

$$\begin{aligned} \delta c &= \text{The error in the model parameters} \\ \mathbf{A} &= \text{Matrix learned by linear regression} \\ \delta\mathbf{I} &= \text{The residual error image vector} \end{aligned}$$

As with the ASM the actual update is weighted as in equation 2.4.

$$c_{n+1} = c_n - k\delta c \tag{2.4}$$

Various values of k are tried until the magnitude of the residual error is decreased. This process is repeated iteratively until no reduction in the magnitude of the residual error is possible. A refinement to this algorithm is to search with models at multiple scales from course to fine, projecting from one model to the next. This can increase the robustness and efficiency of the algorithm as the courser models are less susceptible to noise and finding locally optimal solutions.

2.3.3 Particle Filtering Algorithms

Particle filtering algorithms propagate a prior PDF of predicted object state from frame to frame which is based on a particle (sample) representation. The particle representation is essentially a fixed number of samples with a location (in object configuration space) and a weight related to the probability at the sample location. This representation is only an approximation to the true PDF, however it is computationally efficient to stochastically sample from this probability distribution which is the reason for the power of particle filtering algorithms. Particle filters work in a similar way to the Kalman filter described previously (see section 2.3.1) in that a prior PDF is formed by prediction from previous timesteps, data at the current timestep is evaluated and a posterior PDF is calculated (including the data from the current timestep). The differences between the two filters are that particle filters can represent a non Gaussian prior or posterior PDF (and as such multiple hypothesis) and any suitable predictor may be used with the filter (particle filters are not limited to linear prediction as with the Kalman filter).

Perhaps the first application of a particle filter in machine vision was by Gordon *et al.* [38], however results were only presented for a synthetic scenario. Later Isard and Blake formalised the use of particle filtering techniques in their Condensation (CONditional DENSity propogATION) algorithm [57] and presented results from 'real' tracking scenarios such as head and hand tracking. The Condensation algorithm operates as follows:

1. For the first frame generate a particle PDF either at random or using a process that models the initial state of the system (skip steps 2 and 3). For subsequent frames use the posterior

PDF from the previous frame.

2. Stochastically **select** a sample from the particle PDF described in 1) based on sample weights.

3. **Predict** (using any suitable temporal model) the state of the sample selected in 2) at the current timestep.

4. **Evaluate** the weight of the sample at the location determined in 3) using a suitable fitness measure. Normalise the weights such that the sum of the weights for all samples is unity.

If a single result is required at each timestep (rather than a PDF) the sample with the largest weight may be selected or alternatively a weighted mean may be taken. The latter method should be used with caution as the mean of multi-modal distributions may lie at a point of low probability density.

2.3.4 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) [36] is a class of methods in which the underlying probability distribution of a random process (for example a sampling process) is estimated iteratively using a first order Markov Chain defined by a transition kernel. These methods work by assuming the random process results from a particular distribution type (e.g. a Poisson distribution), taking an initial estimate to the parameters of this distribution and updating this estimate iteratively by sampling the process using the Markov Chain. The transition kernel of the Markov chain is based on the current estimate of the underlying distribution such that if the estimated distribution becomes equal to the actual underlying distribution the process will converge and the transition kernel will remain constant for all subsequent iterations. In practice there is a large number of methods for which convergence will take place, however proof of convergence is difficult for all but relatively simple methods. A generalised MCMC process is illustrated in figure 2.9.

For complex processes, such as those found in machine vision applications, a probability representation based on particles may be used. This representation is used in particle filtering algorithms and is described in section 2.3.3. MCMC methods and particle filtering algorithms have much in common, both being Bayesian sampling/estimation techniques, with the principal difference being the dynamic nature of the problem in most particle filtering implementations. Particle filtering

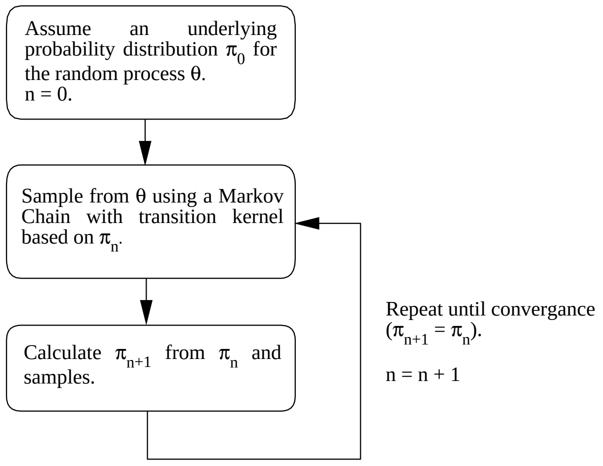

Assume an underlying probability distribution π_0 for the random process θ.
n = 0.
Sample from θ using a Markov Chain with transition kernel based on π_n.
Calculate π_{n+1} from π_n and samples.
Repeat until convergance ($\pi_{n+1} = \pi_n$).
n = n + 1

Figure 2.9: An Overview of Markov Chain Monte Carlo Methods

algorithms are more widely used in machine vision applications than MCMC methods due to their computational efficiency (convergance is not required), however de Souza *et al.*s [25] use of an MCMC based method for fitting point distribution models of fish to static images could be applied to streams of images given sufficient computational power.

2.4 Summary

In this chapter various methods have been presented for the spatial object modelling, temporal object modelling and object tracking. These methods become powerful tools for the analysis of image sequences when combined as a machine vision application. It has been described how object characteristics (both spatial and temporal) may be modelled at various levels of abstraction with the power of the model not necessarily being dependant on the closeness of the model to the object being modelled. Hybrid methods combining both low level and high level abstractions of object characteristics (for example optical flow and 3D textured models [95]) are particularly powerful as are medium level abstractions such as contour representations (e.g. [21]). Methods such as those described in this chapter have been applied in many application domains such as the analysis of human, animal and vehicle motions as well as in the field of medical image analysis, however there is still scope for improvement in terms of the accuracy and robustness of these methods. It is the aim of this thesis to tackle a small part of this remaining problem within the scope of a livestock analysis scenario.

Chapter 3

Automatic Data Collection for Contour Based Models Using Piecewise Linear Approximation

With the advent of statistical model based techniques for object tracking such as the Point Distribution Model (PDM) [21] the problem of data collection has arisen. For these models to accurately model variation within an object class, a large, representative, set of training examples is required. In many cases this is obtained by hand fitting a set of 'landmark points' to a set of training shapes. This is a very time consuming operation and ultimately limits the size of the training set available to build the model.

Baumberg and Hogg [3] present a simple method for automatic data gathering from image sequences. This involves using a least squares minimisation technique to fit piecewise cubic B-Splines to binary object outlines which are extracted from the sequence using background subtraction. This technique produces spline control points at approximately uniformly spaced intervals along the spline. The piecewise B-Spline representation has an arbitrary number of sections with the differential at each side of spline intersections constrained to be equal, to enforce smoothness. Statistical models are built from the spline control points rather than 'landmark points' as in the conventional PDM. This technique is illustrated in Figure 3.1.

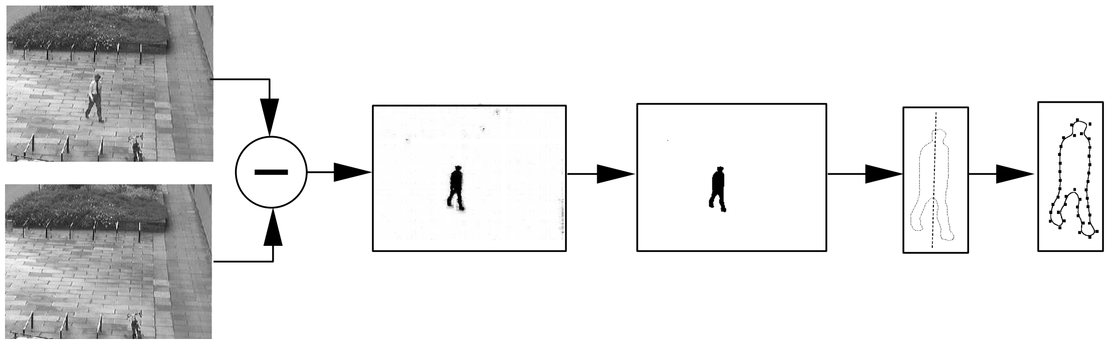


Figure 3.1: Baumberg and Hogg's Method of Generating Landmarks for Individual Shapes

Hill and Taylor [48] present a more complex algorithm based on finding pixel-to-pixel pairwise outline correspondences (for example using curvature) and taking mean shapes of similar pairs in a binary tree until a single mean outline is obtained as in Figure 3.2. A set of landmarks is then fitted to this mean shape. To quote: "Any sensible algorithm can be used for this purpose e.g. a set of landmarks positioned according to equal path-length along the boundary". These landmarks are then projected back down the binary tree using the pixel-to-pixel pairwise correspondences to give a set of landmarks for each training shape from which a PDM is built.

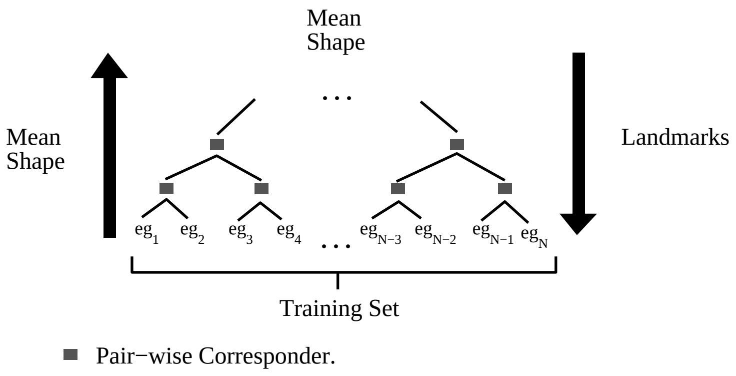

Mean Shape
Mean Shape
Landmarks
eg_1 eg_2 eg_3 eg_4 ... eg_{N-3} eg_{N-2} eg_{N-1} eg_N
Training Set
Pair−wise Corresponder.

Figure 3.2: Hill and Taylor's Method of Generating a Mean Shape and Initial Landmarks

These initial landmark fits are refined to give a more compact representation by minimising the formula given in equation 3.1. This formula has two terms, Mahalanobis distance and Representation Error. Mahalanobis distance is the distance in shape space of the shape from the mean shape of the PDM, weighted by component variance and Representation Error is "the discrepancy between a desired set of landmarks and the ability to represent the suggested landmarks on a given boundary". Thus the optimisation is a compromise between an efficient (compact) representation and an accurate representation. A PDM is constructed from these new landmark points and the process repeated iteratively until convergence.

$$f(\theta, s, \mathbf{b}) = D^2 + a(E^2 - E_0^2) \tag{3.1}$$

Where:

θ = Angle of rotation of PDM

s = Scale of PDM

$\mathbf{b}$ = Shape parameter description vector of PDM

$D = \sum_{i=1}^{N_c} \frac{b_i^2}{\lambda_i}$ (The Mahalanobis distance of data item from the mean in 'shape space')

b_i = Eigen component number i of approximation

λ_i = Eigenvalue number i of approximation

N_c = The number of components used in the approximation

$E^2 = \sum_{i=1}^{N_p} (x_i - x_i^{'})^2 + (y_i - y_i^{'})^2$

x_i, y_i = Approximation to boundary point

$x_i^{'}, y_i^{'}$ = Nearest point on object boundary to x_i, y_i

N_p = No of points in approximation

$a = \frac{2}{\lambda_t}$

$\lambda_t = \sum_{i=1}^{N_c} \lambda_i$

$E_0^2 = E^2$ measured for the mean shape

The two methods mentioned so far have been successfully used in many applications, however in neither case are the 'landmark points' produced salient features over the entire data set. It is desirable that landmark points represent corresponding features over the data set. If this is not the case, the model will include variation due to landmark placement as well as 'real' shape variation. This leads to a non-optimally compact model which will be inefficient in tracking applications. The Baumberg and Hogg method uses an entirely arbitrary number of 'control points' and the positioning of these is at approximately uniform intervals around the perimeter. It is thus conceivable that the position of corresponding control points for similar shapes could map to different physical features, especially if the number of control points selected is non-optimal (see figure 3.3 for examples of this). Within this thesis this phenomenon will be referred to as ‘control point drift’. The Hill and Taylor method (as described in [48]) uses landmarks positioned according to equal path-length along the boundary of a mean shape. This will not produce a set of features which are salient over the entire data set, however if an alternative method is used, for example selecting points of high curvature, these points may map to corresponding points of high curvature in the training set. There is however no guarantee that this will be the case and there is no guarantee that the mean will contain all the features of interest observed over the entire data set.

In practice both these methods compensate for point fitting errors by using more landmarks than is optimal which can reduce the magnitude of errors but makes the model less efficient in tracking applications.

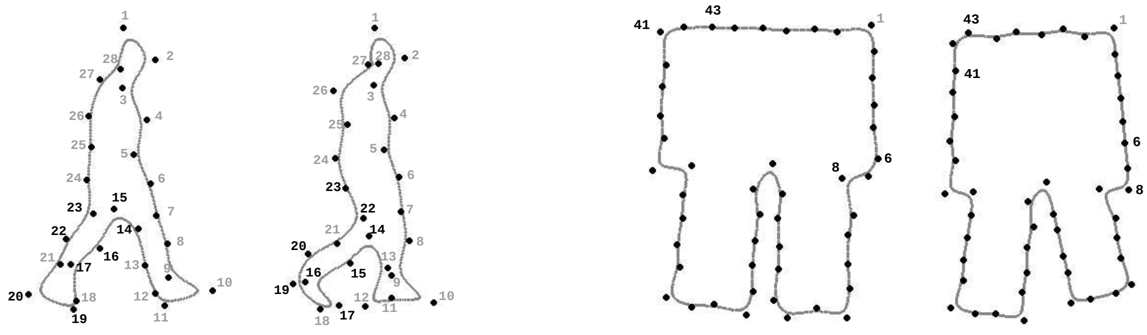


Figure 3.3: Examples of ‘control point drift’ in the Baumberg and Hogg Spline fitting method

An alternative approach to the methods described is presented in the remainder of this chapter. This novel method attempts to extract features which are salient over the entire data set (or a subset of this) for use in shape modeling applications.

3.1 Extracting Salient Features Using Straight Line approximation

Previously it has been stated that the most efficient way of representing the variance within a set of shapes is to model the variation of salient features over this set. This assumes that a two way mapping between a shape boundary and these features is available. An example of such a feature is a straight line approximation. If a set of shapes can be approximated using a piecewise linear model it is simply a matter of extracting these linear features from each member of the data set and building a model. In practice however these salient features are corrupted by noise and by the fact that real objects cannot be exactly approximated by a set of straight lines. The result of this is that, using standard methods of straight line approximation, different numbers of features will be extracted for roughly similar shapes (this is discussed more fully in section 3.1.2 later). A method is presented that addresses this problem using ‘approximation seeding’ and ‘shape simplification’. In some data sets some features are occluded, for example when taking different 2D views of a 3D object or when features self occlude. If models are to be built with landmarks that relate to ‘real’ features, seperate models for each subset of the data set that has the same features visible are required. This is performed automatically in the model building process.

3.1.1 Initial Video Processing

For model building purposes the raw video input is pre-processed in two stages to produce a list of adjacent contour pixel points describing the outline.

Firstly ‘background subtraction’ is carried out to produce a binary silhouette. A background image containing no moving objects is taken as a reference. Each colour component (red, green and blue) of this image is subtracted in a pixel-wise fashion from frames containing moving objects, the absolute value taken and the three colour difference components summed. The result of this is a difference image with only one colour component. Various image processing functions (blurring, median filtering, dilation and erosion etc.) may be applied to this image if required to clean up and smooth the image before it is thresholded to give a binary difference image as shown in Figure 3.4. (see [3]).

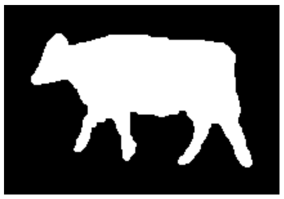


Figure 3.4: Background Subtracted Binary Image

3.1.2 Fitting Lines to Shape Outlines

There has been much written on the approximation of continuous functions by straight lines [30, 82, 108, 113]. The general conclusion is that there is no ‘correct’ solution to such problems. Methods that have been employed to perform such tasks generally fall into two categories (i) Points of maximum curvature or (ii) Iterative end-point fits.

In (i) the continuous function is segmented by picking points of maximum curvature for the continuous function. Curvature is calculated as the differential of tangential angle [81] or less often as arc-chord distance [82]. This method works well for functions that are mainly made up of straight lines but fails when the function has long arcs of constant curvature. Hill and Taylor [48] use a system based on this method to fit points to the outlines of human hands and hearts, but then go on to approximate the outline by a spline.

In method (ii) the continuous function is segmented using an iterative process. The function is first approximated by a single straight line with end points chosen arbitrarily. If the continuous function deviates from the approximation by more than a specified threshold, the line is divided into two lines at the point of maximum deviation. This process continues until no point of interest on the continuous function deviates from the straight line approximation by more than the chosen threshold. This method works reasonably well for functions containing straight and curved segments but is sensitive to the choice of initial points. It is also sensitive to ‘wild’ points and any noise in the continuous function may have a large effect on the final result.

The proposed method is a combination of the two methods with an additional iteration that results in a more robust solution to the problem. Initially the tangential angle of each point on a shape outline is calculated and from these angles curvature at each point is calculated by differentiation. The selection of this method is purely arbitrary and curvature may equally well be calculated as arc-chord distance. Maxima of curvature are found and lines between successive maxima points used as a first estimate of the straight line approximation. It may be observed in Figure 3.5 that, due to perimeter roughness, there are many more of these points than is necessary to approximate accurately the perimeter and there are areas where there few points due to constant curvature. These problems are resolved by the procedure that follows.

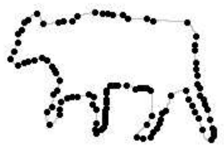


Figure 3.5: Maxima of Curvature

3.1.2.1 Adding Extra Points

Extra points are added to the straight line approximation formed from the maxima of curvature (see figure 3.5) to solve the problem of straight line segments that do not approximate well their corresponding object contour section. This is achieved by calculating the maximum perpendicular deviation of the contour from the straight line approximation for each straight line section (see figure 3.6). If the largest maximum deviation is greater than a specified threshold a point is added on the contour at the point of maximum deviation. This is repeated until the maximum perpendicular deviation of the contour from the straight line approximation is less than this threshold.

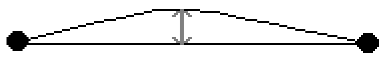


Figure 3.6: Maximum Perpendicular Deviation From Straight Line Approximation

3.1.2.2 Removing and Adjusting Points

As stated previously the straight line approximation based on maxima of curvature has more points that necessary to accurately approximate an object contour due to surface roughness of the contour. This is dealt with by iteratively removing points and adjusting the remaining points to a locally optimal fit. Points are removed one at a time by considering each reference point and the straight line between the two adjacent reference points. The perpendicular distance between this line and the point is considered as in figure 3.7a. If this distance is less than a specified threshold the point is removed, with the point with the smallest distance being removed first.

After each point is removed the remaining points are adjusted locally by considering the movement

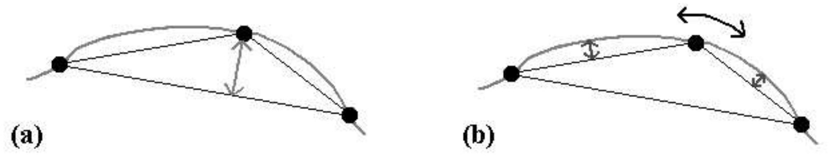


Figure 3.7: (a) Removing Unnecessary Reference Points, (b) Adjusting the Position of Reference Points

of each reference point to each of the two contour points adjacent to these points, as shown in figure 3.7b. The point move that results in the largest reduction in average perpendicular deviation of the object contour from the two straight line approximation sections is performed. This process is repeated until no such move exists and a locally optimal fit is found. It has been observed that excluding moves that result in an increase in maximum perpendicular distance of the contour from any straight line approximation can result in a better final fit, however the effect of the entire adjusting process is marginal and satisfactory results may be obtained by excluding this stage entirely if speed is an issue.

This method is robust in the fact that it produces approximations with similar numbers of lines for similar shapes (see figure 3.8). It is not prone to erroneous results caused by the fit being optimised locally as reported about methods described previously [30].

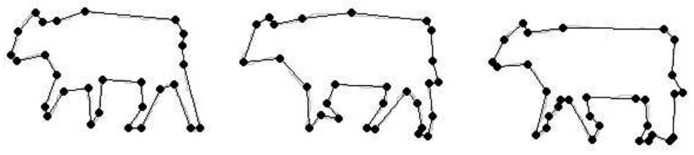


Figure 3.8: Straight Line Approximations to Outlines

3.2 Building Models from Outline Data Using Seeds

The method described in the previous section produces some aesthetically pleasing cow outlines but the number of lines required varies from shape to shape. Even fairly similar shapes may be represented by slightly different numbers of lines.This is a disadvantage when trying to build a concise model of shape using mathematical methods such as PDM's ([21] and [3]). These methods enforce a pre-determined number of points upon the shape regardless of the shape to be modelled. Using common sense a human subject will approximate groups of similar shapes with the same number of lines and different groups with different numbers of lines where these lines will correspond to actual physical features. This approximation is not only based on the outline of a particular shape but on prior knowledge of the object. An extension to the line fitting method proposed in the previous section includes prior knowledge about the shape we are trying to extract using a 'seed' to start the line fitting process. A seed is created by hand placing reference points around the perimeter of a single shape belonging to the class to be modelled and converting these points into a series of vectors with an angle component and a magnitude component that is a proportion of the total distance around the perimeter of the shape. This is then used to fit a straight line approximation to each perimeter in the training set.

The process of fitting the 'seed' shape(s) to the training set aims to minimise the error between the straight line approximation and the actual perimeter while maintaining similarity between the straight line approximation and the seed (in terms of angles and distances). To this end a 'Fit Score' is defined which is minimised to obtain the best compromise between these two criteria.

3.2.1 Choice of Fit Score

The 'fit score' must consist of an 'error metric' which measures the difference between the approximation and the outline and a 'difference metric' which measures the difference between the seed and the approximation. The values of these for each section in the approximation must be combined in a sensible manner.

3.2.1.1 Choice of Error Metric

There are three choices for an error metric; integral difference, maximum deviation or average deviation. Maximum deviation (the maximum perpendicular deviation of the contour from the approximation) is unsuitable as it is sensitive to noise in the contour caused by surface roughness. Average (perpendicular) deviation and integral difference are more suitable as they are less effected by ‘noise spikes’ in the contour. Average perpendicular deviation and integral error are in fact equivalent measures so the choice between these is arbitrary.

3.2.1.2 Choice of Difference Metric

The difference metric models the relative angle and distance differences between the approximation and the seed shape. This may be done directly, however the angle and distance difference metrics must then be combined. This could be performed as a weighted sum or as a multiple. A multiple is unsuitable as if either component was zero the difference metric would be erroneously zero. A weighted sum is an option, however the choice of weights is not obvious. Methods such as entropy maximisation [102] (for a particular data set) could be used, however this would require a data set to base this on, defeating the object of this method. An alternative approach is to use the magnitude of the vector difference as this is related to both the angle and distance differences and requires no weights. The approximation vectors must be normalised by the total perimeter to ensure the two vectors are comparable. This is illustrated in figure 3.9.

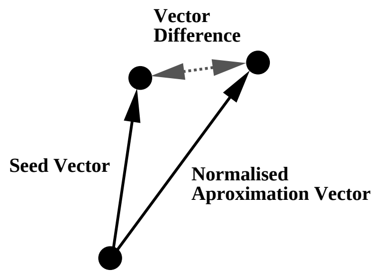


Figure 3.9: The vector difference metric

3.2.1.3 Combining the Error Metric and Difference Metric

This is again the choice of a multiple or a weighted sum. The multiple approach is not suitable as the fit score would be erroneously zero if either metric were zero. This leaves only the option of a weighted sum and the problem of determining suitable weights. As in the previous section an entropy maximisation may be used, however the simpler approach of normalising each component is used. The error metric (integral difference) is normalised by the integral of the area within the seed shape and the difference metric is simply calculated with respect to the seed shape contour resulting in the fit function given in equation 3.2.

$$Fit\,Score = \frac{1}{A}\sum_{n=1}^{N} Integral\,Difference_n + \frac{1}{N}\sum_{n=1}^{N} Vector\,Magnitude\,Difference_n \qquad (3.2)$$

Where:

$A =$ The area within the seed shape

$N =$ The number of linear sections of the seed shape

3.2.2 Minimisation of the Fit Score

The ‘Fit Score’ for a particular perimeter-seed combination could be minimised globally by performing an exhaustive search or using a global search method such as a genetic algorithm. These methods are however very slow and inefficient for the problem of shape matching as a local search with good initialisation is possible in many scenarios. Initialisation of a local search is performed by dividing the perimeter of the training shape up into sections with length proportional to the magnitude components of the seed vectors. The positioning of these sections is determined by moving the approximation with respect to the contour in an exhaustive one dimensional search so as to minimise the fit score. This is illustrated in figure 3.10.

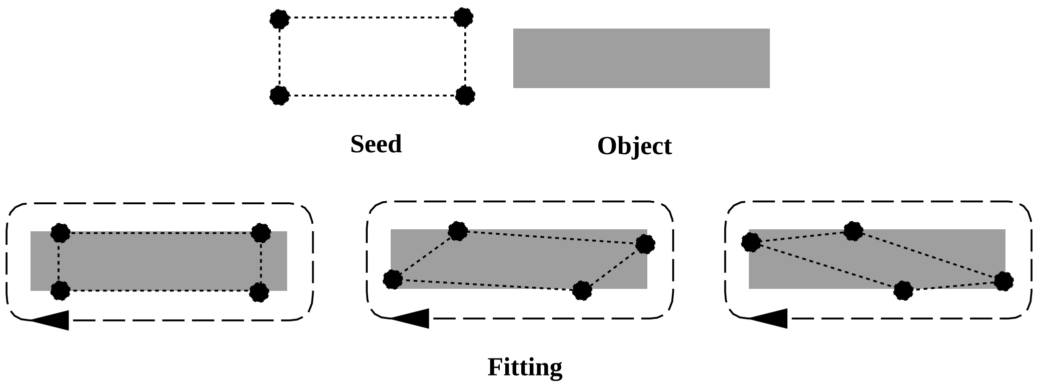


Figure 3.10: Initial seed fitting by cycling the seed round the object contour

The initial guess is improved using a local search method similar to the point adjusting method described in section 3.1.2.2. Point moves are considered which minimise the fit score and points adjusted accordingly, the move resulting in the largest reduction in fit score being performed first. In addition to moves moving landmark points to adjacent perimeter points (as in the point adjusting method), three other possible moves are considered. The locations of these three possible moves are defined as the points of maximum perpendicular deviation from the three lines connecting the current reference point location and the two adjacent reference point locations. This is illustrated in figure 3.11.

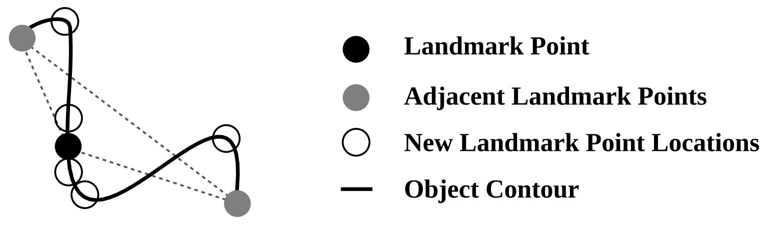


Figure 3.11: The five landmark point positions considered for each landmark point

3.2.3 Using Multiple Seeds

For the main data set used (sequences of cows walking) there are shape discontinuities related to the fact that when a cow's legs come together the number of lines required to represent the shape accurately falls. This is also the case for many other object classes. For this reason multiple seeds are used. For the cow example the seeds are; one with all the legs apart, one with the front legs together and one with the rear legs together, as illustrated in figure 3.12.

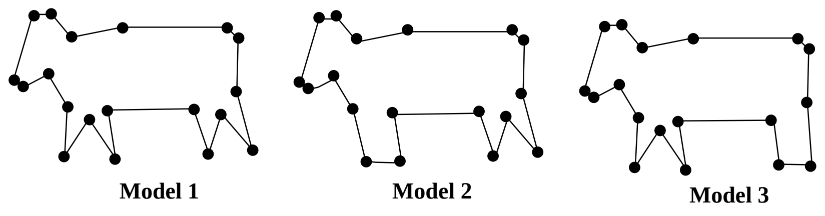


Figure 3.12: The Three Seeds Used For the Livestock Example

The training data is thus partitioned into groups depending on which seed results in the lowest fit score for a particular frame. Frames that have a particularly high fit score (greater than a pre-determined threshold) are discarded. The grouped frame fits are converted into vectors and averaged to produce a new set of seeds. These seeds are applied to the entire data set in the same way as before. This process is repeated until the difference between the averaged output is not significantly different from the seed input. This method does not always converge in the mathematical sense, however after a number of iterations the difference in output observed between iterations stabilises to a sufficiently low level as to be considered stable. Figure 3.13 shows how the frames were grouped in the final iteration for the cow data set. This shows clearly the cyclic nature of cow walking behaviour and how the models may be used in tracking and behavioural studies. In the frame where no model fits satisfactorily the cow is in a transitory state (front legs crossed). This suggests a fourth seed may be required.

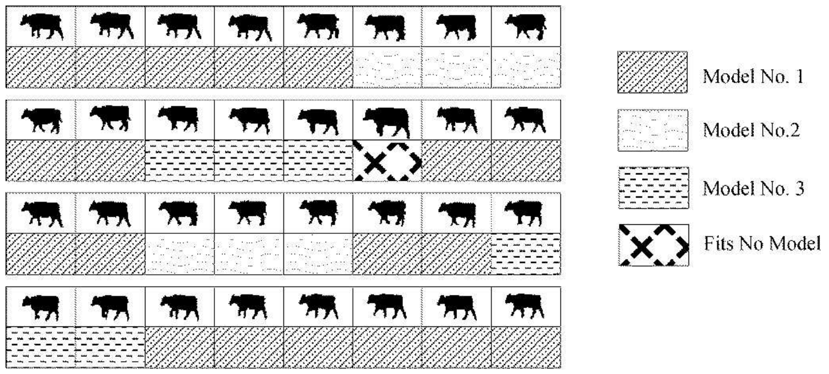


Figure 3.13: Grouping of Successive frames to Form Shape Models

3.3 Automatically Extracting Seeds from Outline Data

The results from the ‘hand seeded’ method described previously are reasonably good but rely on human perception of the number of lines required to represent a given class of shapes and thus the method is not repeatable for an arbitrary shape class. Automation of the seeding process is also desirable if a large number of shape classes are to be extracted from a data set. The method devised uses straight line approximations as extracted in section 3.1.2 as initial seeds and a simplification stage to reduce the number of points necessary to describe a shape class. The straight line approximations used are calculated using the ‘fit score’ described in the previous section using the following method:

1. A similarity matrix is calculated by using each straight line approximation as a seed, fitting it to each other frame and calculating the 'fit score' (see equation 3.2).

2. The matrix is made symmetric by multiplying by its transpose to give a single similarity metric between pairs of frames.

3. Each row is summed to give a similarity score for each frame. The frame with the lowest score (i.e. the frame most similar to all other frames) is selected as the first seed frame.

4. The frame that is most different from the frame selected in 3) (i.e. the frame with the highest 'fit score' in the first seed frame row of the similarity matrix) is selected as the next seed frame.

5. Successive seed frames are selected as the frame that is most different from any previously selected frame. This is done by taking the rows in the similarity matrix for all previously selected frames and finding the lowest element in each column. The frame corresponding to the highest of these values is chosen as the next seed frame.

Figure 3.14 illustrates the results of the calculation of the similarity matrices for a reduced size data set consisting of 4 outlines relating to each of the cow prototype shapes given in figure 3.12.

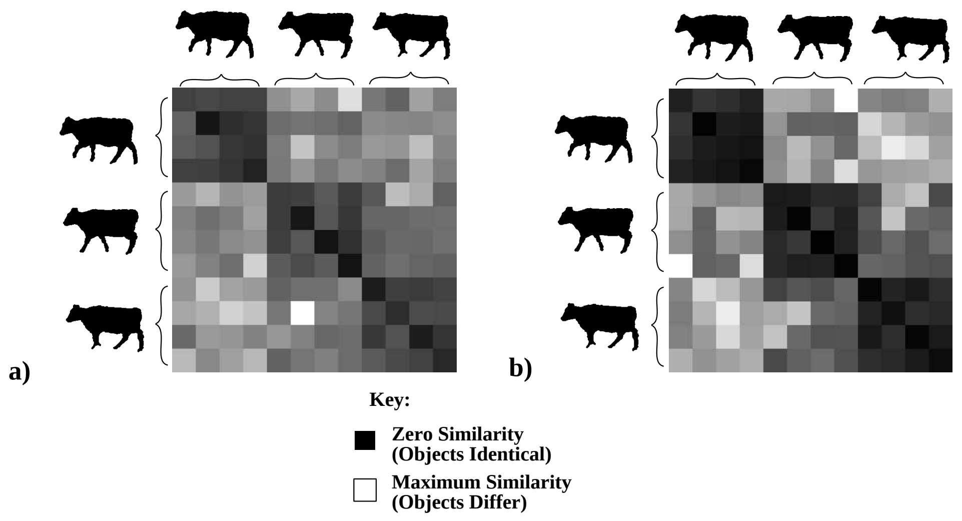


Figure 3.14: (a) Similarity Matrix (b) Symmetric Similarity Matrix

The number of seed frames used may be selected directly or on goodness of fit criteria. Goodness of fit criteria such as putting a lower limit on the similarity value at stage 5) are useful but should be used with caution as criteria may change from one shape class to another.

Once the seed frames are selected they are fitted to the data and grouped as in section 3.2. After each iteration the groups are converted into vectors, averaged and then converted back to points using the average perimeter as a scaling factor. The number of these points is reduced using the ‘remove points’ algorithm described in section 3.1.2.2 in which points which are close to the straight line between the adjacent two lines are removed. This simplified average is converted back into vector form and used to seed the next iteration. This algorithm can lead to the shape being over simplified as illustrated in figure 3.15.

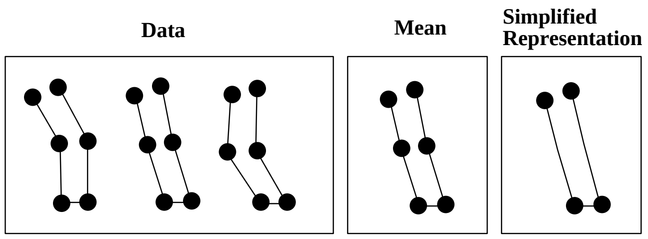


Figure 3.15: An Illustration of the Over Simplification Possible Using the Method

A modification to the algorithm may be incorporated to solve this problem whereby the ‘remove points’ algorithm is no longer performed at the point where the seed ceases to fit the data to a satisfactory degree as defined by the ‘fit score’. The unmodified version resulting from the previous iteration is then used as the seed for the next stage. From then on the ‘remove points’ algorithm is no longer applied and the iteration works exactly as in section 3.2.

This method produces aesthetically pleasing results, as can be seen in the models produced (see Figure 3.16), but there is room for refinement in two areas. Firstly fitting each frame to each other frame is computationally expensive. If speed were an issue it is obvious to see how this stage could be refined to use fewer comparisons by identifying similar shapes from the first few frame fits. Other refinements to this method could be implemented in the dynamics of the iteration. Over-simplification is observed to be a problem and the solution provided, although reasonably robust, is fairly basic. There is much scope for subtle refinement of this iteration, however this has not been investigated as the presented method was observed to perform satisfactorily for the purposes intended.

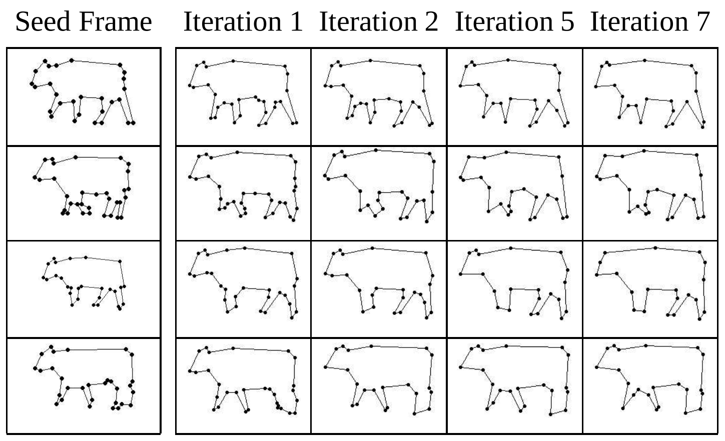


Figure 3.16: Mean Model Over Several Iterations of the Model Building Process

3.4 Evaluation

3.4.1 Application to Synthetic Data

As discussed previously the contours of real objects are not necessarily made up of precise straight line sections. This makes the quality of any straight line approximation subjective. For a quantative evaluation of this method a synthetic data set of card animals and people was created as illustrated in figure 3.17.

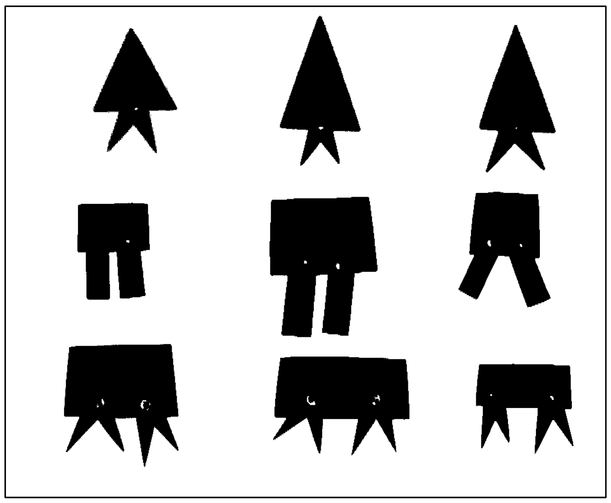


Figure 3.17: Synthetic Paper Animals and People Used in the Evaluation

Twenty examples each of three individuals for each object class (triangle people, square people, square pets) were created using a low quality camera connected to a computer (180 examples). The leg position and pose of individuals were altered from one example to the next. These images were then thresholded to give images such as those shown in figure 3.17. This process created silhouettes with comparable perimeter roughness to images of real objects, but with clearly defined straight line approximations.

The method was applied to this synthetic data set with various numbers of models and various ‘fit thresholds’. Using three models and a suitably high fit threshold so as to include 100% of silhouettes, three models were built corresponding to the three object classes used (triangle people, square people, square pets). All silhouettes were correctly classified by model and the salient straight line features of each object class were successfully extracted. Example results are illustrated in figure 3.18.

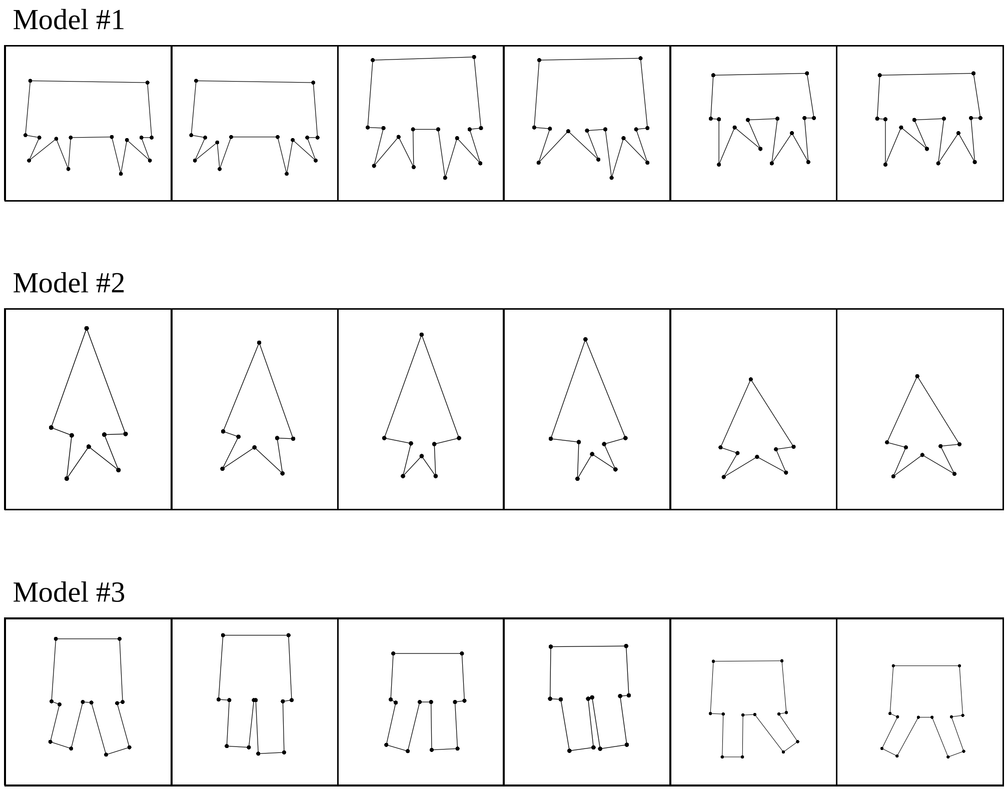


Figure 3.18: Selected results from applying method to synthetic data

Not all fits were good using a high fit threshold. Figure 3.19 illustrates some of the poorer fits.

If the method is applied to the data set with different numbers of models and the average fit score

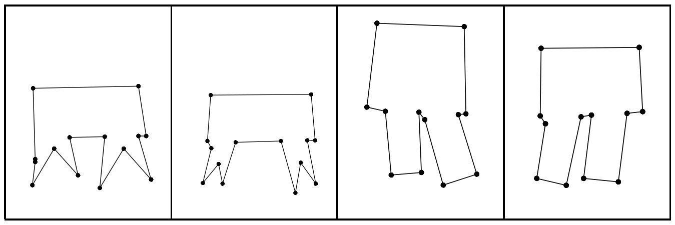


Figure 3.19: Erroneous Approximations at High Fit Threshold

examined there is a clear drop in average fit score between 2 and 3 models. This is shown in figure 3.20. Such a graph could be used in the selection of the number of models if the number of object classes were unknown.

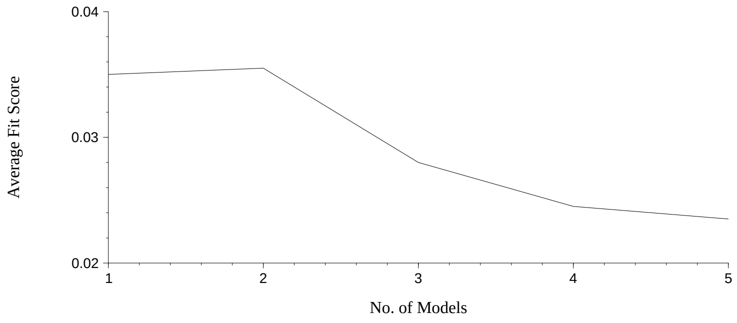
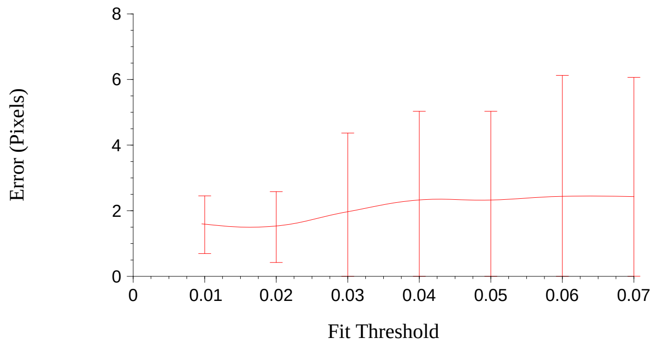
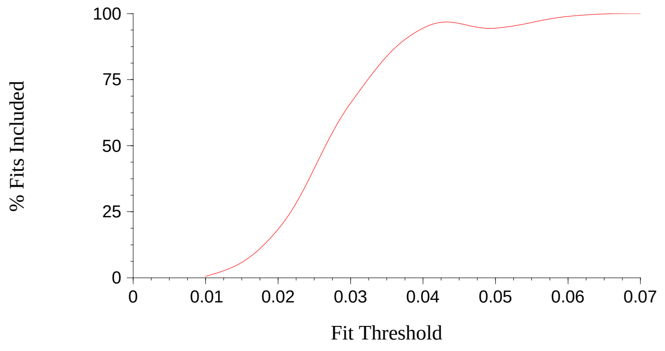


Figure 3.20: Graph of Average Fit Score vs. Number of Models for Synthetic Data

For a quantative measure of the quality of the approximations the results were compared to a hand fitted ‘ground truth’ measurements. The results are dependent on the threshold used on the fit score to reject erroneous fits. The lower the threshold is set the better the fits, but the fewer silhouettes used. Lowering the fit threshold improves good fits over the iterations as well as rejecting erroneous fits as erroneous fits are not included in the average used to seed the next iteration. Figure 3.21 illustrates the trade off between quality and quantity of data.



a)



b)

Figure 3.21: Fit threshold vs. a) Ground Truth Error b) Percentage of data included in model

For comparison the robustness of hand fitted data was evaluated by performing multiple hand fits and examining the distance from the mean hand fitted points. The mean and standard deviation of these distances are presented in figure 3.22.

Mean	Standard Deviation
1.22	0.89

Figure 3.22: Mean and standard deviation difference from the mean for hand fitted shapes (pixels)

The results in figures 3.21 and 3.22 show that, although the automatically fitted approximations are not as accurate as the hand fitted approximations, the results produced are of the same order as the hand fitted approximations and are as such perfectly acceptable. The size of the errors for the automatic approximations is approximately 1-2% of the average object width or height (around 200 pixels) whereas the variation in the hand fitted data is around 0.5%. It should be noted that these figures are not directly comparable as the variation of the hand fitted data and the automatic approximation are both based around the mean of the hand fitted data (there is no absolute ground truth). Hand fitted data has however been considered acceptable in past applications (e.g. [19]) and is a good basis for comparison.

3.4.2 Application to Real Data

The evaluation presented in the previous section demonstrates the effectiveness of the algorithm on objects that are made up of easily definable straight line sections. In this section qualitative results are presented for the algorithm when applied to a selection of real life objects. The first 'real' data set used was a set of outlines of hands. 20 examples of left hands and 20 examples of right hands (in different poses) were scanned in using a similar method to the synthetic data. The graph of average fit score versus the number of models used is given in figure 3.23.

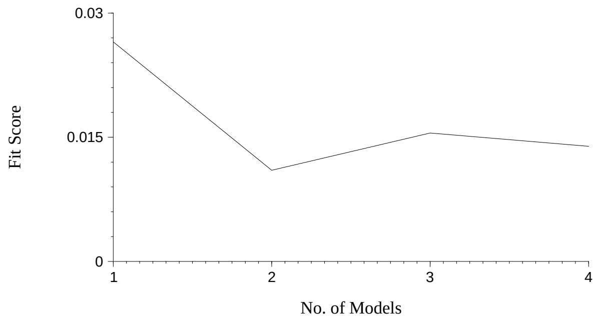


Figure 3.23: Graph of Average Fit Score vs. Number of Models for Hand Data

Figure 3.23 shows a significant fall in average fit score between 1 and 2 models. The slight increase in fit score between 2 and 3 models is due to the effects of at least one model having insufficient data. This can lead to over-simplification of the model and the final model having a poor fit. Using two models the silhouettes were successfully grouped with one model based only on left hand silhouettes and one model based only on right hand silhouettes. All silhouettes were included in the models and there were no obviously erroneous approximations. The complete set of approximations is given in figure 3.24.

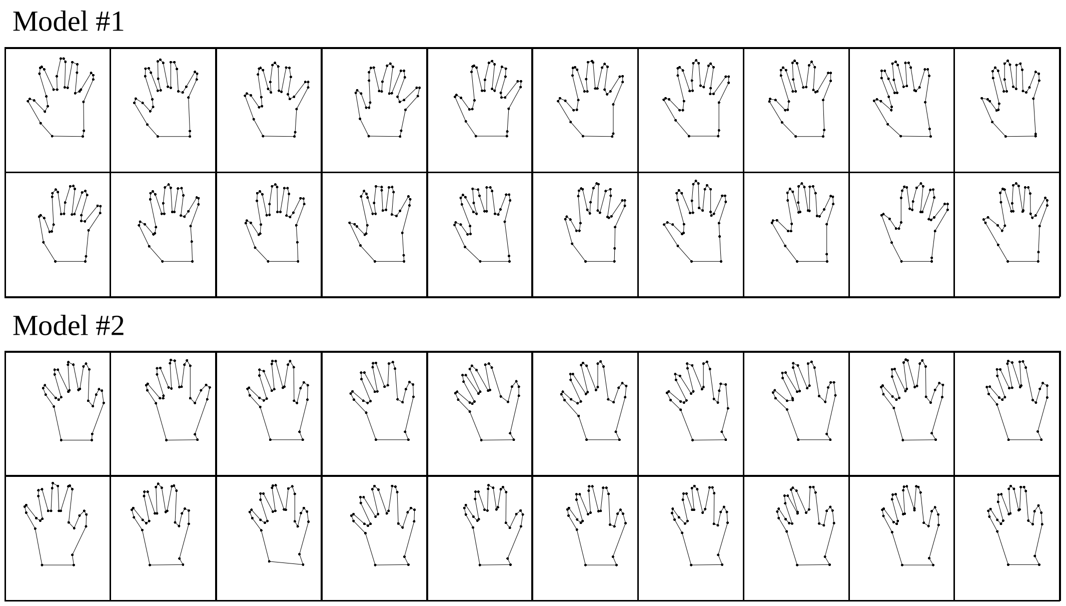


Figure 3.24: Mean Model Over Several Iterations of the Model Building Process

The second set of ‘real’ data was 60 outlines of cows taken from three seperate animals. Silhouettes were extracted using background subtraction as described in section 3.1.1. The graph of average fit score versus the number of models used is given in figure 3.25.

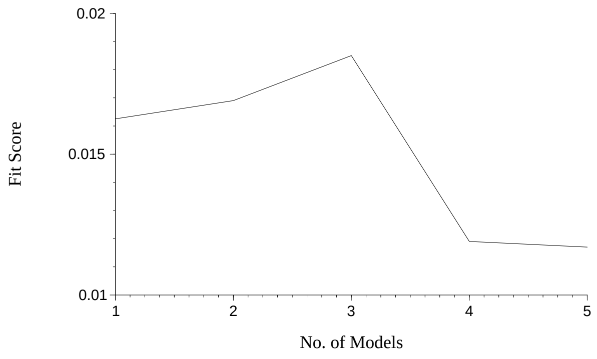


Figure 3.25: Graph of Average Fit Score vs. Number of Models for Cow Data

The graph in figure 3.25 shows a significant drop in average fit score between 3 and 4 models. This suggests that using 4 models is significantly better than using 3 models (as used in the hand seed example previously). It can be seen from the fits illustrated in figure 3.26 that models 2 and 4 are the ‘front legs together’ and ‘rear legs together’ models used in the hand seeded example, however there are two models in which all four legs are visible (models 1 and 3). These two models represent two quite different poses (in terms of leg joint angles) of the animals, although the salient features (straight line sections) present in the two models are similar (in terms of topology). In terms of the definition of a salient feature (similar in distance and angle) the algorithm has made the correct choice of number of models, even though this conflicts (slightly) with initial human intuition. The final decision as to whether 3 or 4 models should be used must rest on other considerations, such as the computational cost and robustness of applications using these models, in addition to the information supplied by the average fit score. The fit score can only be taken as a guide in such cases.

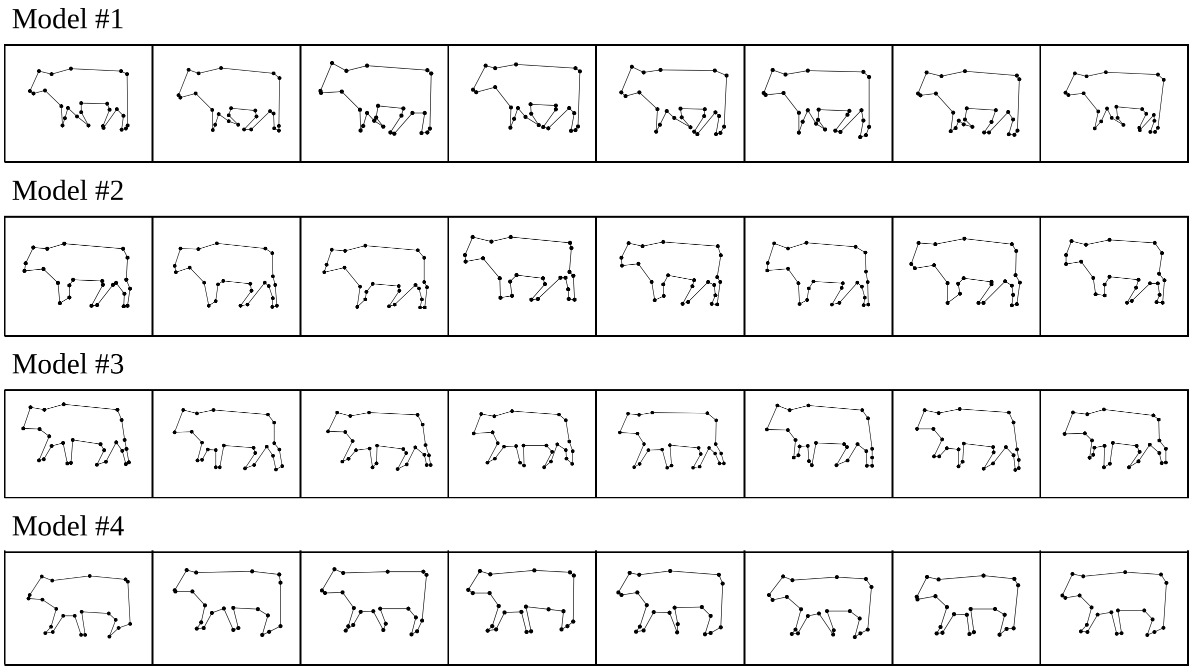


Figure 3.26: Sample Results of Algorithm Applied to Cow Outlines

3.5 Discussion

The method described in this chapter is a good way of producing a useful, and reasonably concise, description of a contour data set based on a particular set of salient features (straight line sections). This method differs from previous methods ([3, 48]) in that multiple models are used to restrict the choice of ‘landmarks’ to features that are salient over a sub-set of the data when there is only partial salience over the complete set. This increases the specificity of the representation at the cost of a degree of compactness and simplicity. The number of models is variable and determined semi-automatically, such that if the data set may be represented by a single model it will be.

The ‘fit seeding’ procedure used within this method allows salient features to be determined either manually or automatically. Automatic seeding can, when features are not clear, lead to an over-simplified representation of the data. This is the principal limitation of the method. Over simplification may be tackled by making the shape simplification criteria tighter, however this can lead to erroneous landmark placement if features are not clear. The magnitude of the error is of course limited by the constraints imposed by the seed, however these errors can be significant. This is tackled, to a certain extent, by rejecting fits with a high fit score from the final model. Fit score appears to correlate reasonably well with the quality of the fit (for particularly bad fits) and is a fairly robust metric for rejecting poor fits. In a perfect system there would be no poor fits, however the proportion of poor fits in this method is generally low ($<$10%) and grossly erroneous fits are detected and rejected. Manual seeding suffers many of the problems of automatic seeding and care should be taken with the selection of features.

Chapter 4

A Vector Based Contour Model for Object Tracking

In chapter two a number of models which are widely used in object tracking applications were described. These may be divided broadly into two categories; appearance based models and contour based models. The principal application of this thesis is the monitoring of Dairy Cow behaviour. These animals exhibit modelable variation in shape from one animal to another, however their appearance (in terms of local colouring) varies from one individual to another in an effectively random way. It is believed by farmers that no two individuals have the same local colouring, in fact local colouring is even used in the identification of individual animals (see chapter 8). For this reason it is difficult to build a generic appearance type model for dairy cows that retains any specificity with regard to the characteristic markings exhibited by these animals. It is possible to build appearance based models for individual cows or generic models that may be initialised for a particular individual, however the problem of object tracker initialisation is complex if such models are used and this prohibits their practical use in 'real time' monitoring systems for the present.

Contour based models were chosen for the prototype system implemented primarily because of the problems with appearance based methods discussed previously. Contour based models are not without their problems, primarily the lack of a direct mapping between the model and image intensities, however they are computationally efficient to evaluate and a substantial body of research exists on them [66, 8, 19, 3, 57].

4.1 Choice of a Contour Model

A simple contour model, known as the ‘Active Contour Model’ or ‘Snake’, based on a B-Spline representation was presented by Kass [66]. This was fitted to an image by minimising an ‘Energy function’ based on edge gradient information (from an edge detector) and smoothness constraints using a local gradient descent method. This method fails badly in the presence of clutter or if initialisation is poor. This is principally due to the fact that no *a priori* information about the object to be found is included in the model. *A priori* information may be included in a model by manually defining a model with properties similar to the object (e.g Hogg’s ‘Walker’ model [49] or the simpler model of Nash *et. al.* [78]) or by using a generic model where parameters are ‘learned’ from a training set of objects belonging to a given class (e.g. The Point Distribution Model of Cootes *et. al.* [21]). The desired properties of such a model are Generality (i.e. the model will represent all members of a given class of object) and Specificity (i.e. the model will only represent objects of the specified class). There is always a trade off between generality and specificity in an object modelling scheme. Hand specified models are generally simplifications of the object class they are trying to represent and as such tend to be overly specific, not representing the entire range of variation within the object class. Learned models can represent the entire range of variation within the training set. They may not be specific enough however, and can represent objects that do not belong to the class being modelled. There is also the problem of data collection for learned models which can be very time consuming.

In chapter 3 it was shown how it is possible to automatically generate the training data required for learned models such as the Point Distribution Model (PDM) which makes these models an attractive proposition. Hand defined models are also difficult to specify and may not model the complete range of variation within the object class. For these reasons learned models are used in the example application presented within this thesis.

4.2 Existing Point Distribution Model Schemes

Since the emergence of the original Point Distribution Model (PDM) [21] (which is described in detail in chapter 2) there have been a number of variations on the original idea. The original scheme modelled the variation of a fixed number of points (normalised by translation, scale and rotation) using Principal Components Analysis (PCA). This was initially applied to the modelling of the shapes of electrical components but has since been applied to numerous applications (primarily in medical imaging). Baumberg and Hogg [3] took this idea and applied it to the control points of a closed B-spline representation for human motion tracking. This simplified the training data acquisition problem and meant the 'landmark points' did not have to represent real salient features where these were unavailable (as in the case of 2D views of humans). The disadvantage of this scheme is that it adds an additional layer of complexity which can result in a more complex model than one constructed from true landmark points in cases where linear motions in the object domain map to non-linear motions in the spline point domain. This is illustrated by the phenomenon of 'control point drift' around the contour as discussed in chapter 3.

Heap and Hogg [44] tackle the problem of modelling object classes where rotational variation is possible using a polar co-ordinate PDM. Rotation is not modelled well by the conventional PDM as rotation must be approximated by combinations of linear eigenvectors. This leads to a model with poor specificity as the model contains more degrees of freedom than the object class. Within a suitable polar co-ordinate system rotational variation is linear and can be modelled by a single eigenvector, thus limiting the number of degrees of freedom and increasing the specificity of the model. The disadvantage of this scheme is that linear variation is not modelled efficiently leading to the use of hybrid cartesian-polar PDMs with different elements of an object being represented in different co-ordinate systems. This method was applied to he modelling of an angle poise lamp and human hands. Sumpter and Boyle [102] show the need for component scaling when components are measured in different units (e.g. radians and centimetres). They present a method for determining the scaling factors using eigen entropy maximisation for a particular example data set, however these factors will change from set to set and as such are not a general method for determining the scaling factors required for a polar PDM or hybrid cartesian-polar PDM. Heap and Hogg use the simpler method of multiplying the angle components by the average radius over the data set based on the fact that angle multiplied by radius gives arc displacement in the same units as the radius (assuming the angle is measured in radians). Tipping and Bishop [107] assume that all components may be approximated by a single Gaussian with unit standard deviation. This assumption may be used to scale components by dividing by the actual standard deviation of each component. This method is obviously not suitable if the Gaussian assumption is violated. The choice of pivot points and polar co-ordinate systems is critical to the success of the polar co-ordinate PDM, however this is not necessarily a straight forward task. Heap and Hogg [44] present a method for automatically determining pivot points, however the polar co-ordinate system for each point is determined arbitrarily by taking angles with respect to the previous section of a 'pivot tree' as shown in figure 4.1.

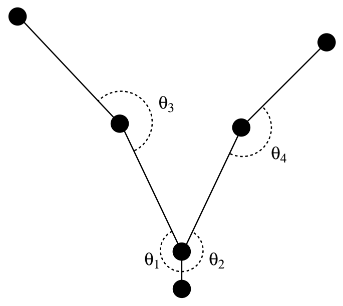


Figure 4.1: An example 'Pivot Tree' used in the polar co-ordinates PDM scheme

All polar co-ordinate systems have an inherent discontinuity (e.g. at the 2π to 0 transition or π to $-\pi$ transition) which must be avoided by choice of a suitable co-ordinate system for each part of the system. There is no completely general automatic way of performing this task. In addition, using the average radius as a scaling factor for the angle components is only valid in cases where the radius does not change significantly (as arc length is only equal to radius times angle if the radius is constant). These constraints limit the application of the polar co-ordinate PDM and go some way to explain its less than widespread use in comparison to the conventional PDM.

The three PDM schema are illustrated in figure 4.2.

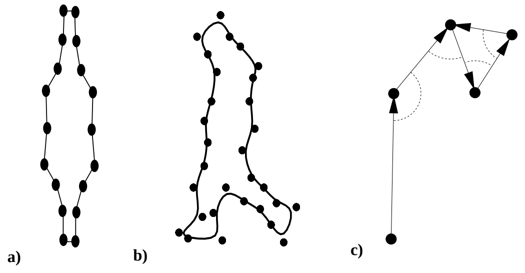


Figure 4.2: PDM Schema; a) Original Scheme, b) B-Spline based PDM, c) Polar Co-ordinates PDM

4.3 The Vector Distribution Model : A Novel PDM Scheme

In the previous section it was discussed how the polar co-ordinates PDM offers a more specific representation of cyclic movements than the conventional PDM without sacrificing the generality inherent in the PDM scheme. The disadvantages of this scheme are the poor representation of linear variation, manual choice of polar co-ordinate systems and the difficulty in determining suitable scaling factors for the components measured in different units. A novel 'Vector Distribution Model' is presented that uses the vector principal of the polar co-ordinates PDM, but uses cartesian vector representations rather than polar representations. This is illustrated in figure 4.3.

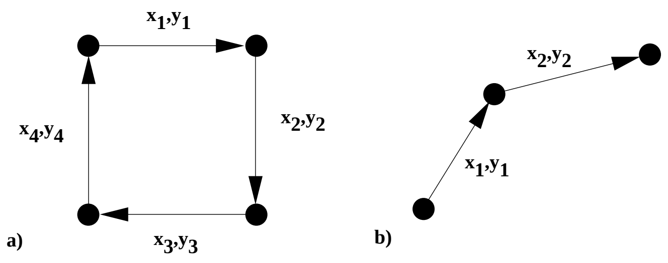


Figure 4.3: The Novel 'Vector Distribution Model' Representation For a Closed and an Open Perimeter

The vectors (x_n, y_n) are normalised by the total perimeter calculated as in equation 4.1.

$$\text{Total Perimeter} = \sum_{n=1}^{N} \sqrt{x_n^2 + y_n^2} \tag{4.1}$$

Where:

$N =$ The number of points in the Vector Distribution Model

It can be seen in figure 4.3b that the dimensionality of the data is reduced by 2 for representations where the ends are not closed. It is interesting to note that this is also effectively the case for representations where the ends do join up as the last vector adds no information to the representation. This is evaluated in section 4.5.

There is no need for iterative normalisation of translation and scale (Procrustes alignment) in this scheme, as in the conventional PDM scheme (described in appendix A), as individual examples are normalised by the vectorisation process. Rotation is not normalised, however the problem of rotational normalisation is much simplified by this approach as it becomes a one dimensional problem (optimise over rotation) rather than four dimensional problem (optimise over x translation, y translation, scale and rotation) as in the original scheme. In many circumstances the advantage of rotational normalisation is not clear. Rotational normalisation will produce a more compact model, however useful rotational information is lost if this is performed. Using the example of human tracking, it can be seen in figure 4.4 how different poses with similar shapes may be normalised to an un-realistic arbitrary rotation by the iterative process used in the conventional PDM.

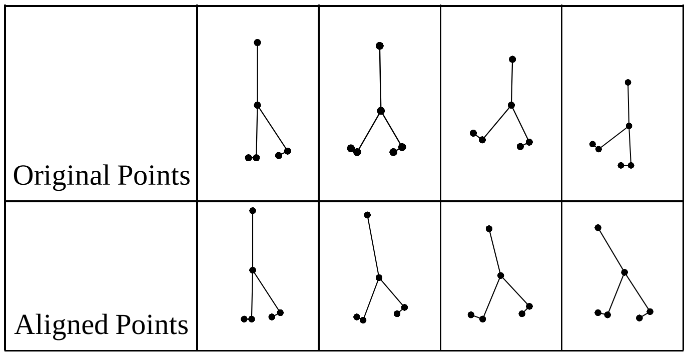


Figure 4.4: The Effect of Procrustes Alignment on a set of synthetic 'Stick People'

The effect of the loss of rotational information is application specific. In the Active Shape Model [19] a shape is compared to the PDM to check for shape validity. In this situation there is no problem with rotational normalisation as the algorithm is only interested in valid shapes regardless of their translation, rotation and scale. An alternative approach to model fitting is sampling, as in the Condensation algorithm [57]. In this algorithm loss of rotational information is important as sampling must be performed over a wide range of rotations which exceeds the actual range of rotations exhibited by an object under some circumstances. This leads to sub-optimal sampling and possible erroneous solutions. In practice rotation could be modelled by a separate model, however this would not be necessary if it were not normalised out by the Procrustes alignment.

The Vector Distribution Model (VDM) models well systems which include small rotational variations, as rotation may be modelled by a linear transform for small angles. This is also true for the conventional PDM, however the advantage of the VDM is in systems with multiple connected rotations such as the system illustrated in figure 4.5 where a pivot point of rotational variation is not a fixed point. In such systems the variation of individual points is complex, however the variation of individual vectors is simple thus favouring the VDM and the polar co-ordinate PDM over the original PDM. Human and animal limbs are examples of such a system.

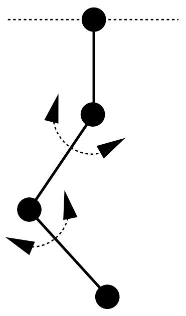


Figure 4.5: A System Well Suited to the VDM

4.4 Evaluation of PDM Schemes

PDM schemes must be compared on the basis of their relative specificity and generality in addition to the practical considerations discussed previously. Three PDM schemes have been evaluated; the original scheme, the polar co-ordinate PDM and the new VDM. The B-spline based PDM is not evaluated as it is simply a specific implementation of the original PDM scheme. Two different polar co-ordinate PDM (PPDM) schemes based on relative angles (as used by Heap and Hogg [44]) and absolute angles (i.e. with respect to the vertical axes) were evaluated to demonstrate the effect of using different co-ordinate systems. Synthetic data sets were constructed that exhibit rotational variation about a non-fixed point to demonstrate the drawbacks of the original PDM scheme. These data sets also exhibit general linear variation (i.e. linear variation not parallel to any vector) to demonstrate the drawbacks of the polar co-ordinate PDM. These properties are exhibited by many ‘real life’ objects, in particular humans and animals. The synthetic data sets were built by modelling a three pendulum system (see figure 4.6) with non-multiple frequencies and variable maximum angular variation magnitude (22.5, 45 and 90 degrees). Additive Gaussian noise was added to the data set members to simulate an observation process (e.g. manual or automatic point fitting). A synthetic data set was used as opposed to a set based on ‘real life objects’ as variation is more readily controllable and ground truth is easily available.

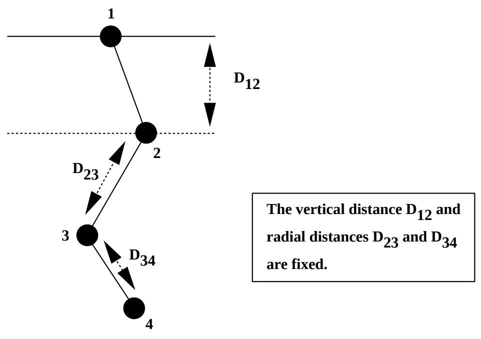

1
2
3
4
D_{12}
D_{23}
D_{34}
The vertical distance D_{12} and radial distances D_{23} and D_{34} are fixed.

Figure 4.6: The synthetic pendulum system used in the evaluation of PDM schemes

To measure the trade off between specificity and generality the relationship between reconstruction error and model compactness (the number and dimensionality of eigenvectors) is investigated for the different PDM schemes. The reasoning behind this is that, if all eigenvectors are used in a reconstruction, a PDM scheme is completely general (i.e. will model anything). As the number of eigenvectors in the reconstruction is decreased the model becomes more specific and less general, modelling the data set more and more poorly. An ideal PDM scheme will not loose its power to model the particular object class it is built to model as the number of eigenvectors is decreased until this falls below the number of degrees of freedom of the object class. The test object has three degrees of freedom.

For a fair comparison of methods rotation is not normalised in any scheme, although translation and scale are normalised using the standard technique for each method (Procrustes alignment or vectorisation). If rotational variation is included in the data set, the true ability of a scheme to model rotational variation may be determined. This ability is important in complex systems where rotational alignment may give erroneous results or remove important rotational information from the data set (see figure 4.4).

4.4.1 Evaluation Results

Note: Results presented are mean results over the data set. Standard Deviations are given in square brackets.

PDM Scheme	No. of Components Used					
	1	2	3	4	5	6
PDM	5.17 [2.12]	3.03 [1.68]	1.23 [0.62]	0.69 [0.50]	0.39 [0.31]	0.00 [0.00]
VDM	7.35 [3.06]	4.60 [2.16]	1.23 [0.63]	0.72 [0.53]	0.04 [0.04]	0.00 [0.00]
PPDM (Relative Angles)	8.60 [4.02]	6.91 [4.50]	1.27 [0.67]	0.73 [0.56]	0.05 [0.04]	0.00 [0.00]
PPDM (Absolute Angles)	7.93 [3.46]	5.15 [2.45]	1.26 [0.67]	0.72 [0.56]	0.05 [0.04]	0.00 [0.00]

Figure 4.7: Average RMS point reconstruction error (pixels) for different PDM schemes for data set with 22.5 degrees maximum angular displacement

PDM Scheme	No. of Components Used					
	1	2	3	4	5	6
PDM	9.99 [4.13]	5.52 [3.29]	1.62 [0.79]	1.43 [0.70]	0.76 [0.56]	0.00 [0.00]
VDM	13.92 [5.71]	8.61 [4.07]	1.50 [0.69]	0.92 [0.58]	0.37 [0.28]	0.00 [0.00]
PPDM (Relative Angles)	17.77 [8.55]	13.94 [9.47]	1.46 [0.79]	1.37 [0.71]	0.81 [0.59]	0.00 [0.00]
PPDM (Absolute Angles)	16.24 [7.35]	10.97 [5.27]	1.45 [0.79]	1.36 [0.71]	0.81 [0.59]	0.00 [0.00]

Figure 4.8: Average RMS point reconstruction error (pixels) for different PDM schemes for data set with 45 degrees maximum angular displacement

PDM Scheme	No. of Components Used					
	1	2	3	4	5	6
PDM	21.40 [7.53]	12.40 [6.05]	5.41 [3.89]	3.35 [1.47]	1.52 [1.04]	0.00 [0.00]
VDM	28.64 [13.79]	19.66 [8.31]	4.24 [1.51]	2.81 [1.30]	1.04 [0.69]	0.00 [0.00]
PPDM (Relative Angles)	35.92 [17.41]	26.45 [18.08]	3.74 [1.91]	3.68 [1.96]	0.92 [0.70]	0.00 [0.00]
PPDM (Absolute Angles)	33.68 [15.33]	21.91 [9.93]	3.69 [1.93]	3.68 [1.96]	0.92 [0.70]	0.00 [0.00]

Figure 4.9: Average RMS point reconstruction error (pixels) for different PDM schemes for data set with 90 degrees maximum angular displacement

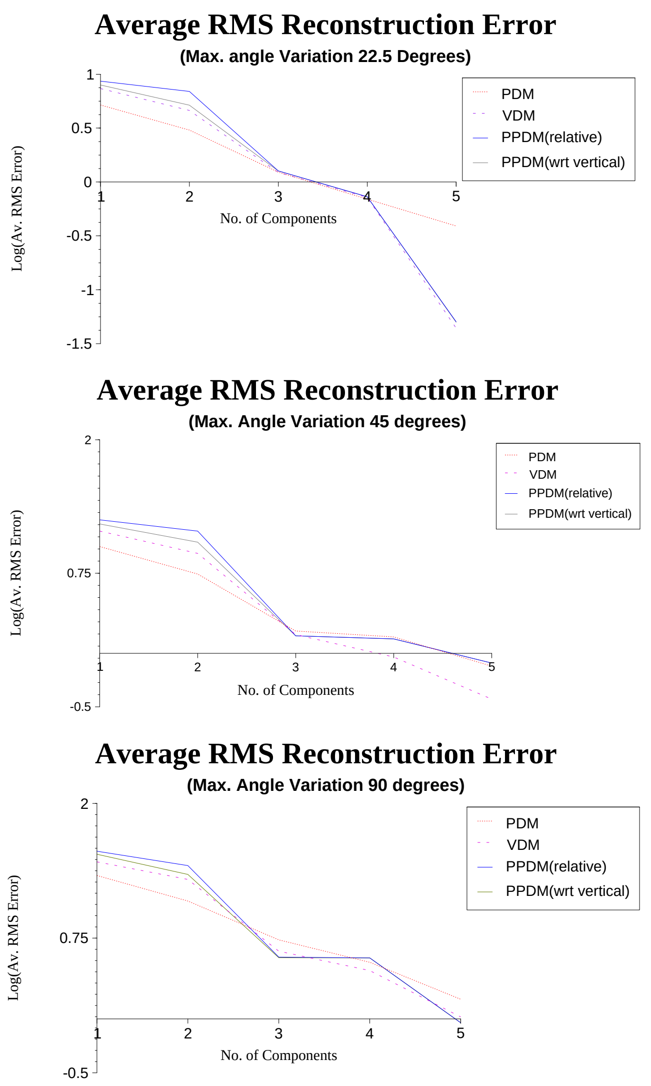


Figure 4.10: Average RMS point reconstruction error for different PDM schemes for data sets with varying maximum angular displacement

The results given in figures 4.7 to 4.10 show clearly the differences between the different PDM schemes. For reconstructions with fewer eigenvectors than the number of degrees of freedom of the system (3) the conventional PDM scheme outperforms the other schemes. This is not however a sensible operating point for any scheme as the reconstruction errors are high. When reconstructing with an equal or greater number of eigenvectors than the number of degrees of freedom of the system the Vector Distribution Model (VDM) and Polar Co-ordinates PDM (PPDM) schemes outperform the conventional PDM as their eigenvectors more closely map to the degrees of freedom of the system. In this operating region the performance of the VDM scheme and the two PPDM schemes is similar for the small angle variation example (22.5 degrees maximum variation) and the large angle variation example (90 degrees maximum variation), however the VDM scheme outperforms all other schemes for the intermediate angle variation example (45 degrees maximum variation). The PPDM scheme based on absolute angles marginally outperforms the PPDM scheme based on relative angles between sections in all cases.

The reconstruction results presented may be explained by considering the architectures of the different schemes. The conventional PDM scheme does not include the concept of rotation and, as such, the different rotational variations are non-orthogonal (as well as being non-linear). The scheme maximises the variance along each orthogonal eigenvector in turn which approximates to minimising the reconstruction error for the training set for successive numbers of eigenvectors. After the first eigenvector the remaining eigenvectors are constrained to be orthogonal to all previous eigenvectors, which is a very restrictive constraint for systems where the true set of variations (or the best linear approximation to these) are non-orthogonal. This explains the decreasing performance of the conventional PDM with increasing numbers of eigenvectors when compared to the other schemes.

When comparing the VDM scheme to the PPDM schemes the different representational powers of the two co-ordinate systems must be examined. The polar co-ordinates system models rotation about a fixed point in a perfectly linear fashion (i.e. as a single component), however a linear movement non-parallel to the line between a point and its pivot is highly non-linear in this co-ordinate system. Conversely the vector co-ordinate system used in the VDM represents any variation which is linear in the point domain as linear, however rotational variation is non-linear and can only be approximated by a linear variation in the vector co-ordinate system. This approximation is better the smaller the angle of rotation as shown in figure 4.11.

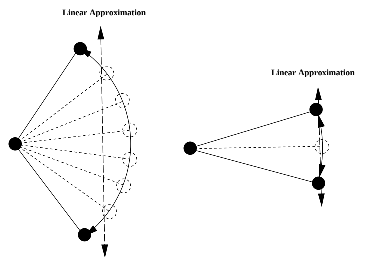
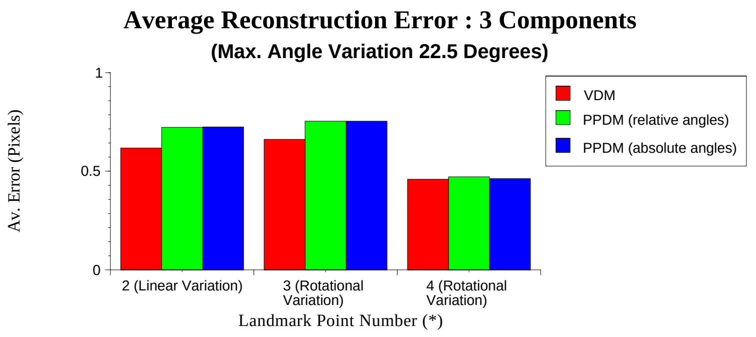
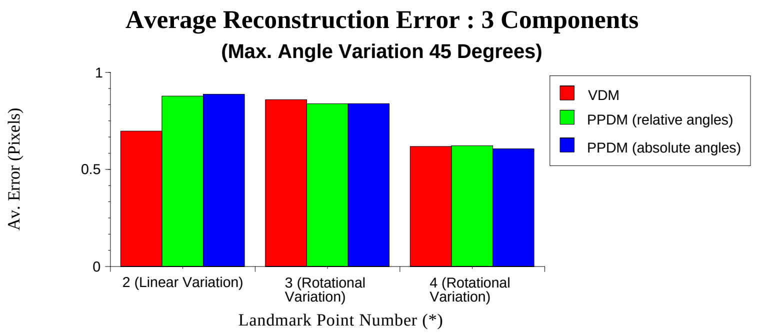
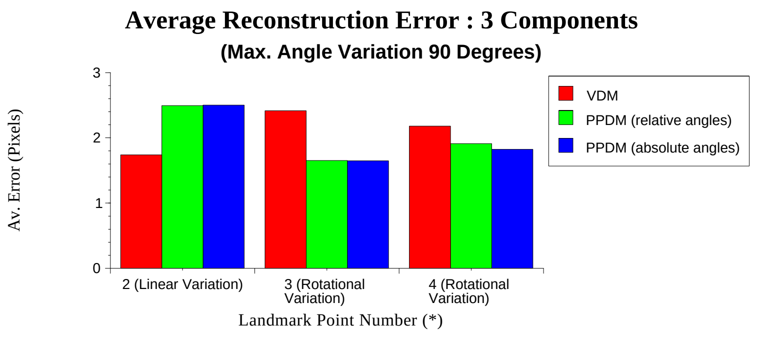


Figure 4.11: Linear approximations to a large angular variation and a small angular variation

In the synthetic system used in this evaluation there is both linear and rotational variation. The magnitude of both the linear and the rotational variation increases with the angle variation and, as such, the ability of the different schemes to represent different parts of the system changes. This is demonstrated by plotting the average errors for individual points as in figure 4.12.







$*$ See figure 4.6

Figure 4.12: Average point reconstruction error using 3 components for different PDM schemes for data sets with varying maximum angular displacement

Figure 4.12 shows that, for large variations, the VDM approximates linear variation better and the PPDM approximates rotational variation better. As the magnitude of the variation is decreased the differences between the schemes is less marked, however the rate of increase in the ability of the VDM to model rotation is greater than the rate of increase in the ability of the PPDM to model linear variation, for this system. This results in better performance of the VDM scheme at intermediate variation magnitudes. For the very small angular variation example (22.5 degrees maximum variation) the VDM even performs marginally better for the rotationally varying points than the PPDM schemes, however this is due to the fact that errors across different points are not completely independent due to the nature of the system, and the rotational results are effected by the inability of the PPDM schemes to model the linear variation.

To determine why the two PPDM schemes produce different results the eigenvectors and eigenvalues must be examined. Figure 4.13 shows the eigenvalues for all schemes and figures 4.14 and 4.15 show as an example the first few eigenvectors for the two PPDM schemes for the 90 degrees maximum variation data set.

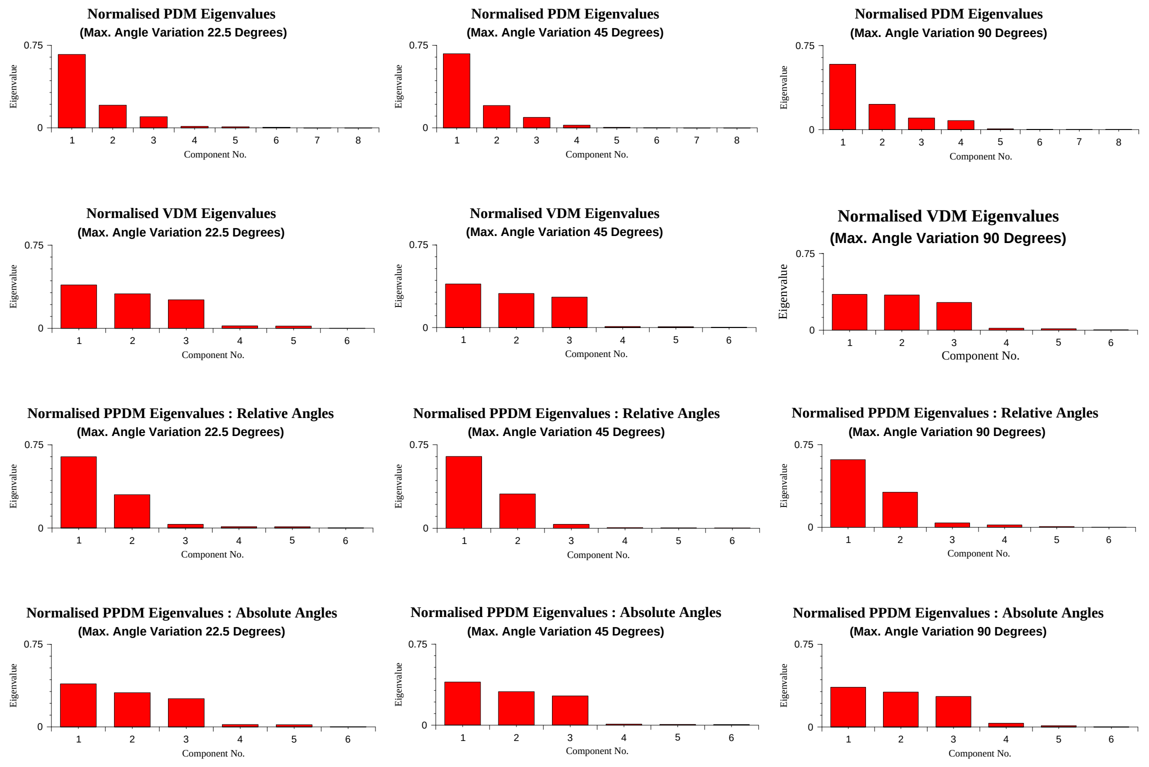


Figure 4.13: Eigenvectors for different PDM Schemes for data sets with varying maximum angular displacement

It can be seen from figure 4.15 that the PPDM based on absolute angles models fairly well the

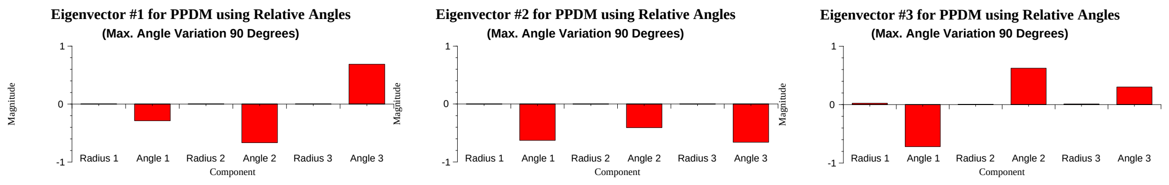


Figure 4.14: The first 3 eigenvectors of the PPDM using relative angles for the 90 degrees maximum variation data set

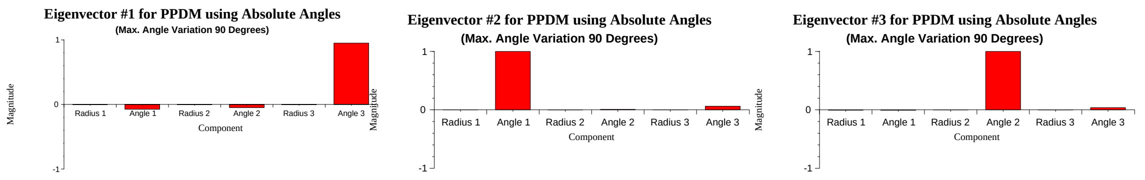


Figure 4.15: The first 3 eigenvectors of the PPDM using absolute angles for the 90 degrees maximum variation data set

three degrees of freedom of the system as three separate eigenvectors (although the second eigenvector only approximates the linear variation in the first section). This results in the eigenvalue distribution given in figure 4.13 where the variance is fairly evenly distributed in the first three components and is very low in the remaining components. The PPDM based on relative angles between sections does not model the three degrees of freedom as seperate eigenvectors, although radial variation is minimised. This is due to the fact that the true variation is with respect to the global co-ordinate system and the mapping between this system and the relative angle co-ordinate systems introduces co-variances between the angles. Although the effect of the choice of co-ordinate system on the representational power of the PPDM is marginal (in this case), this illustrates the importance of appropriate choice of co-ordinate system as well as selection of pivot points.

4.5 Dimensionality Reduction in Closed Contour Vector Representations

In section 4.3 it was shown how the dimensionality of an open contour system is reduced by two by vectorisation. This is also the case for closed contour systems if the final vector adds no new information (i.e. the ends join up in all cases). This is the case if the sum of all x and y vector co-ordinates is zero for the mean vector and all eigenvectors. If all training examples have a closed contour this is approximately the case. To illustrate, this calculation was determined for some of the data sets used in chapter 3. The results are given in figures 4.16 and 4.17.

Vector/Mean	$\sum X$ (Pixels)	$\sum Y$ (Pixels)	Eigenvalue
Mean	0	0	N/A
EV 1	0.000022	0.000004	1.69×10^{-03}
EV 2	0.000017	0.000018	1.08×10^{-03}
EV 3	-0.000043	0.000062	3.51×10^{-04}
.			.
.			.
EV 14	0.000144	-0.001407	1.15×10^{-06}
EV 15	-1176.095359	1055.754149	1.46×10^{-16}
EV 16	1055.754157	1176.095342	5.27×10^{-18}

*Figure 4.16: Sum of X and Y components for mean and eigenvectors for VDM built with synthetic 'Triangle People' outline data (Dimensionality 2*8)*

It can be seen from figures 4.16 and 4.17 that the magnitude of these sums is orders of magnitude lower than the variation within the data sets. It is in fact several orders of magnitude lower than the pixel accuracy used. For the final two eigenvectors this is not true, however there is effectively zero variation in these components so there is no effect on reconstructions within the training data set. The final two components allow the eigensystem to model any system of vectors (not simply the ones in the training set) and thus model variations in the final vector that mean it does not close the contour. This is to be expected as a complete eigensystem can model any system with the same dimensionality as the training data as it is simply a rotation of the raw vector space.

Vector/Mean	$\sum X$ (Pixels)	$\sum Y$ (Pixels)	Eigenvalue
Mean	0	0	N/A
EV 1	0.000019	0.000005	4.34×10^{-04}
EV 2	-0.000008	0.000024	2.32×10^{-04}
EV 3	-0.000013	-0.000011	1.84×10^{-04}
.		.	
.		.	
EV 62	-0.000390	0.000119	2.84×10^{-07}
EV 63	218.691786	-4156.211159	3.49×10^{-17}
EV 64	-4156.211203	-218.691785	4.37×10^{-19}

*Figure 4.17: Sum of X and Y components for mean and eigenvectors for VDM built with Cow outline data (Dimensionality 2*32)*

In practice the results presented in this section mean that the last two components of the mean and each eigenvector need not be used in the projection from the eigenspace to a point representation. These components are also not required in the projection from the point representation to the eigenspace as they only effect the last two components that have no variance within the training set (and should have little variance for similar object shapes). Effectively the dimensionality of the problem is reduced by two, increasing the operating speed of associated algorithms. It should be noted that the dimensionality of the vectors should not be reduced before the Principal Components Analysis is performed as this produces a different set of eigenvectors and eigenvalues due to the variation in the final vector not being included.

4.6 Discussion

The results in this chapter have shown that PDMs based on different co-ordinate systems are suitable for modeling different types of variation. The case of rotational variation is interesting as it is poorly modeled by the conventional PDM scheme. This is a problem as many ‘real life’ objects, in particular biological objects, exhibit rotational variation. Using polar co-ordinate based PDMs (PPDMs) it is possible to model rotational variation very specifically, however these schemes cannot model general linear variation well and the choice of pivot points, axes reference and scaling factors is not necessarily straight forward. The novel Vector Distribution Model (VDM) presented in this chapter was designed as a compromise between the conventional PDM scheme and polar co-ordinates based methods. Linear variation is modeled perfectly by this scheme and rotational variation is modeled well by a linear approximation assuming the magnitude of the rotational variation is not too large (magnitudes up to 45 degrees are modelled with usable accuracy in the scope of this thesis). This scheme is particularly applicable to modeling object classes that include both linear and limited rotational variation, for example Livestock and Humans. An alternative approach would be to use a hybrid modeling scheme using cartesian, cartesian vector or polar co-ordinates for different parts of an object where most applicable (as demonstrated by Heap and Hogg [44]), however the advantage of the VDM scheme is the completely automatic nature of the model building process which requires no manual (or complex automatic) choice of pivot points, angle reference or scaling factors.

The principal application of this thesis is livestock tracking and analysis. It is discussed in later chapters how a quantised shape model is used in this process. For a good quantisation the modeling scheme must provide high generality at low dimensionality for the data class to be modeled (i.e. good reconstructions at low dimensionality are required). Specificity is not so much of an issue as this is provided by the vector quantisation process, however specificity and generality are inherently linked. The VDM was chosen as the modeling scheme for the example implementation (a Livestock tracker and lameness detector) as the object class to be modeled (Livestock) exhibit both linear variation (e.g. in the different builds of individuals) and rotational variation (e.g. in the leg movements). The VDM provided a much simpler, automatic way of modeling this variation than a hybrid cartesian-polar scheme but with similar representational powers (as the rotational variation is small).

Chapter 5

Separation of Variation into Meaningful Components Using ‘Delta Analysis’

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) have been introduced in chapter 2 and are explained in more detail in appendix D. These are methods which may be used to parameterise the variation within a data set in such a way that the dimensionality is reduced or the discriminating power of the representation is increased (or both). They have been used to build models for tracking such as the Point Distribution Model (PDM) [21] and the Active Appearance Model [18].

Successive applications of LDA may be used in the Active Appearance Model to separate out different types of variation such as the differences between individuals, lighting conditions, facial orientation and facial expression when modelling faces. Edwards *et. al* [31] separate out one type of variation by grouping the data into classes and then construct a new data set from the difference between the original data set and the least squares approximation in the truncated ‘discriminant space’. The process is repeated on the new data set using a different set of classes until all the different types of variation have been separated out. A ‘residual space’ may be added to the set of ‘discriminant spaces’ by performing standard PCA on the remaining difference data set. This method produces a set of orthogonal spaces, the eigenvectors of which may be combined (as in equation 5.1) to form a single representaion for use in tracking applications.

$$x = \bar{x} + \sum_{i=1}^{N_1} b_{1,i}E_{1,i} + \sum_{i=1}^{N_2} b_{2,i}E_{2,i} + \sum_{i=1}^{N_3} b_{3,i}E_{3,i} + \ldots. \tag{5.1}$$

Where:

x	$=$	Reconstruction from model
$\bar{x}$	$=$	Model mean
N_n	$=$	The number of linear components of eigenspace n used
$b_{n,i}$	$=$	The model control parameter for component i of eigenspace n
$E_{n,i}$	$=$	The eigenvector for component i of eigenspace n

In this chapter the problems of LDA, when used on certain data sets, are highlighted and an alternative approach to component separation and model building known as ‘Delta Analysis’ is presented. This approach uses the same model formulation used by Edwards *et al* [31] (described by equation 5.1), although the eigenvectors are formed in a different way. This new method is evaluated against the conventional LDA approach.

5.1 The Problem with LDA

Mathematically LDA works by forming two covariance matrices of the data around class means (intra-class covariance) and of the class means around the global mean (inter-class covariance). The ratio of inter-class variation over intra-class variation is maximised by solving a generalised eigen equation involving these two covariance matrices (see appendix D). This is fine for data sets where class means have similar intra-class characteristics, however this may not always be the case due to the number of samples being finite or due to characteristics in the sampling method. Under such circumstances intra-class variance is simultaneously minimised and maximised and the end result is a compromise, including an element of intra-class variation in the discriminant space which should be optimised for inter-class discrimination. An example of this problem is given in figure 5.1 where leg position is obviously an intra-class property, however leg position differs between the two class means due to the nature of our sample selection.

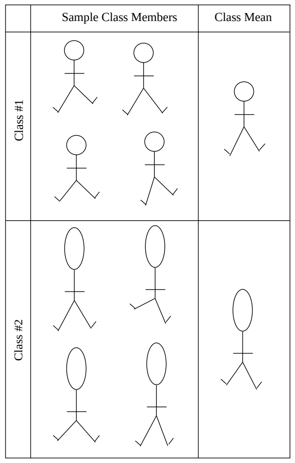


Figure 5.1: Example showing class means with differing intra-class properties

5.2 Creating Complementary Eigen-spaces Using Delta Analysis

Delta analysis is an alternative to LDA that does not use class mean information, instead modelling within class and between class variation explicitly using deltas between data items.

5.2.1 Creating an Intra-class Optimised Eigen-space

An intra-class optimised eigen-space is created by forming a covariance matrix (S_w) from the deltas between each pair of data items as in equation 5.2.

$$S_w = \frac{1}{n_t} \sum_{i=1}^{n_c} \sum_{j=1}^{n_i} \sum_{k=1,k\neq j}^{n_i} (\mathbf{y_{i,j}} - \mathbf{y_{i,k}})(\mathbf{y_{i,j}} - \mathbf{y_{i,k}})^T \tag{5.2}$$

Where:

$\mathbf{y_{x,y}}$ = Member y of class x	n_c = The total number of classes
n_t = The total number of data items	n_i = The number of members in class i

PCA is performed on this matrix to create a 'delta eigen-space'. For each pair of data items there are two deltas ($\overline{AB}$ and $\overline{BA}$ for pair A and B) which are equal and opposite to each other. In this way the mean of the deltas is zero and the relationship between the data and its projection in eigen-space simplifies to that given in equation 5.3.

$$\mathbf{d} = \mathbf{y}E \tag{5.3}$$

Where:

$\mathbf{d}$ = A delta vector	E = A matrix of Eigenvectors
$\mathbf{y}$ = Vector in ‘Delta Eigen-space’	(One Eigenvector per row)

It would not be correct to project raw data vectors into this ‘Delta Eigen-space’, however; as the deltas are relative and have zero mean they may be considered with respect to any point in data space. A new ‘Data Projection space’ with identical axes to the ‘Delta Eigen-space’, but with an origin representing the data mean (rather than the mean of the deltas), may thus be defined and this will be optimised for within-class variation. The variances of the components in this ‘Data Projection space’ are not equal to the eigenvalues produced by PCA but may be calculated by projecting the data into this space and calculating the variance for each component in the standard way.

5.2.2 Creating an Inter-class Optimised Eigen-space

Creating an optimal inter-class space is harder than creating an optimal intra-class space since intra-class variation is included in the inter-class deltas. One approach to isolating inter-class information is to form the product of an inter-class covariance matrix and the inverse of an intra-class covariance matrix. This is similar to LDA described in appendix D, but this method does not always produce real eigenvalues (this can be a problem with LDA also [4]).

Instead the variance of the intra-class deltas is used to normalise the inter-class deltas by dividing each inter-class delta component by the equivalent intra-class variance. This approach does not include intra-class covariance information and thus relies on inter-class modes of variation being fairly well de-coupled from within class modes of variation but will always produce a real solution. Equation 5.4 describes the exact formulation of the covariance matrix (S_b).

$$S_b = \frac{1}{n_t}\sum_{i=1}^{n_c}\sum_{j=1,j\neq i}^{n_c}\sum_{k=1}^{n_i}\sum_{l=1}^{n_j}\delta_{i,j,k,l}\delta_{i,j,k,l}^T \tag{5.4}$$

Where:

$\delta_{i,j,k,l} = (\mathbf{y}_{i,j} - \mathbf{y}_{k,l})\mathbf{v}_w$	$n_t =$ Total number of data items
$\mathbf{y}_{x,y} =$ Data Vector y of class x	$n_c =$ Number of classes
$\mathbf{v}_w = (\frac{1}{v_1}, \frac{1}{v_2}, \ldots, \frac{1}{v_n})$	$n_i =$ Number of members in class i
(The Inverse Intra-class Variance Vector)	$n_j =$ Number of members in class j

A between class ‘Data Projection Space’ is formed from the eigenvectors of this covariance matrix in exactly the same way as the within class space (see section 5.2.1).

5.2.3 Combining Intra-Class and Inter-Class Spaces in Practical Applications

If the example variance graphs in figure 5.2, which are taken from the cow outline example in section 5.3, are examined, we see at first glance that the space produced using conventional PCA appears much better optimised for dimensionality reduction than the two spaces produced by delta analysis. However, if the first few vectors of these eigen-spaces are projected into PCA eigen-space (see figure 5.3) it is interesting to note that, for this data set, most of the energy in these vectors lies in the first few components of this eigen-space and all of the first few eigen components have power in at least one of the delta space vectors.

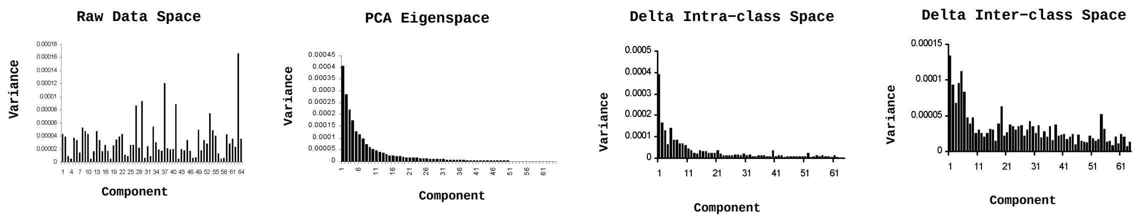


Figure 5.2: Variances of Components in Various Spaces

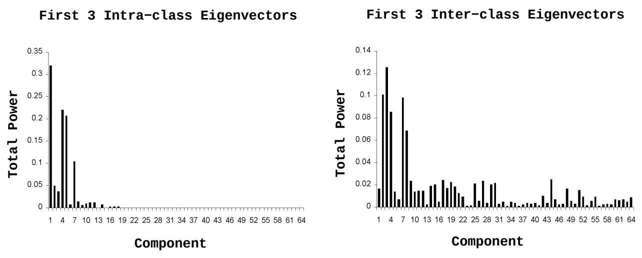


Figure 5.3: Power of Inter and Intra-class Eigenvectors Projected into PCA Eigen-space

These results show that the first few eigenvectors in PCA eigen-space may be approximated by the first few eigenvectors in the within and between class 'Data Projection Spaces' and as such the entire data set way be approximated using these vectors as in equation 5.5.

$$\mathbf{x} \approx \bar{\mathbf{x}} + \sum_{n=1}^{n_{wc}} w_n \mathbf{e}_{wc_n} + \sum_{n=1}^{n_{bc}} b_n \mathbf{e}_{bc_n} \tag{5.5}$$

Where:

$\mathbf{x}$ = Data vector to be approximated	n_{wc} = Dimensionality of Truncated Within Class Space
$\bar{\mathbf{x}}$ = The mean vector	n_{bc} = Dimensionality of Truncated Between Class Space
$\mathbf{e}_{wc_n}$ = Within Class Eigenvector n	w_n = Component Value for Within Class Eigenvector n
$\mathbf{e}_{bc_n}$ = Between Class Eigenvector n	b_n = Component Value for Between Class Eigenvector n

The values of n_{wc} and n_{bc} will obviously depend on the particular data set but this example suggests significant dimensionality reduction may be possible. The disadvantage of this scheme is that within class eigenvectors are not orthogonal to between class eigenvectors and thus there is not necessarily a unique solution when X is known and w_n and b_n need to be calculated. A least squares minimisation method may be used to find an optimal solution, however it may be computationally more efficient to find firstly an optimal solution for inter-class parameters ($b_{1\to n}$) only and then complete the solution by finding the optimal intra-class parameters ($w_{1\to n}$). This scheme applies well to object tracking where inter-class parameters are constant over time and need only be found once. An evaluation of the approximation accuracy of this scheme compared to PCA is presented in section 5.3.3.

5.3 Application to Data

Delta analysis has been tried on two sets of point data, a synthetic set (Triangle people), and a set derived from live data (cow outlines) and a set of intensity images (AT&T Database of faces [92] - 10 examples of 40 people). Triangle people are very simple figures made from card triangles and a drawing pin to pivot the 'legs'. 20 examples of 10 individuals were created using an 8 point approximation (see Figure 5.4.a). 118 cow outlines were obtained by associating points with landmarks on cows from video sequences (see figure 5.4.b). Data sets were divided into equal sized training and test sets for classification experiments. Vector distribution models (see chapter 4) were built from the point data sets and delta analysis performed on the raw vector data. Delta analysis was applied to the face data using a simple nearest neighbour clustering algorithm on the grey-level intensity deltas and the PCA method for large dimensionality data described in chapter 2 (reducing covariance dimensionality from 10304 to 200).

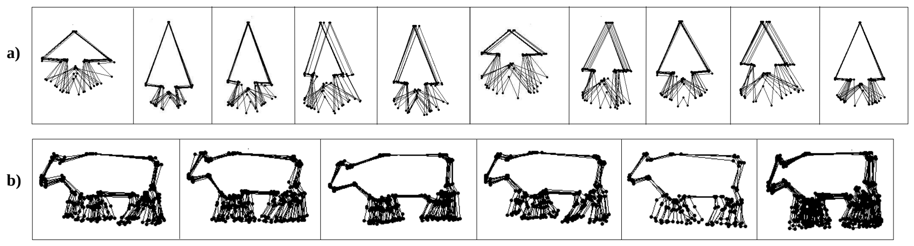


Figure 5.4: (a) 'Triangle people' data (b) Cow data

5.3.1 Evaluation of Inter-class Separation

Delta analysis was performed on the training data sets and the resultant inter-class data projection spaces used as a nearest neighbour classifier. These were compared to nearest neighbour classifiers in canonical space, combined eigen-canonical space, PCA eigen-space and raw vector space. Delta analysis was also performed on PCA data in a similar way to the combined eigen-canonical transform [104]. It was initially unclear whether Euclidean or Mahalanobis distance (a distance metric normalised by variance) should be used in spaces where the variance of components is known, so both were tried where possible for evaluation. Figures 5.5 to 5.9 show the results of these tests. The final row in the tables, 'Information content', gives the sum of the PCA eigenvalues for components used divided by the sum of all eigenvalues to give an idea of the information content of the reduced space.

Space Used	No. Components Used							
	1	2	3	4	...	8	...	16
BC Delta Space (Mahalanobis)	88.5	97	99	96	...	85.5	...	85.5
BC Delta Space (Euclidean)	88.5	99.5	99.5	99.5	...	90	...	94.5
PCA (Mahalanobis)	86.5	52	78	79	...	95	...	98
PCA (Euclidean)	86.5	81	96	95	...	95.5	...	95.5
Eigen-Canonical (Max. Dimensionality)	27.5	57	60.5	73.5	...	42.5	...	44.5
Eigen-Canonical (Half Dimensionality)	46	62	71	76.5	...	33.5	...	-
Regular Canonical	71.5	93	91	89	....	40	...	24.5
PCA+Delta BC Space (Mahalanobis)	28.5	66.5	86.5	92	...	95.5	...	95.5
PCA+Delta BC Space (Euclidean)	28.5	65	87	92	...	98	...	98
Raw Vector	-	-	-	-	...	-	...	94.5
Information content	0.59	0.77	0.89	0.98	...	0.99	...	1

Figure 5.5: Nearest Neighbour Classification Accuracy (%) For Various Spaces (Triangle People)

Space Used	No. Components Used							
	1	2	3	4	...	32	...	64
BC Delta Space (Mahalanobis)	30.3	53.5	58.6	59.6	...	72.7	...	73.7
BC Delta Space (Euclidean)	30.3	53.5	60.6	60.6	...	73.7	...	73.7
PCA (Mahalanobis)	11.1	25.3	41.4	43.4	....	85.86	...	97.8
PCA (Euclidean)	10.1	24.2	36.4	40.4	...	47.5	...	47.5
Eigen-Canonical (Max. Dimensionality)	26.3	43.4	59.6	35.4	...	66.7	...	96.0
Eigen-Canonical (Half Dimensionality)	28.3	41.4	50.5	45.5	...	55.6	...	-
Regular Canonical	23.2	36.4	47.5	49.5	...	22.2	...	34.3
PCA+Delta BC Space (Mahalanobis)	10.1	10.1	10.1	11.1	...	46.4	...	57.5
PCA+Delta BC Space (Euclidean)	11.1	13.1	14.1	13.1	...	53.5	...	60.6
Raw Vector	-	-	-	-	...	-	...	73.7
Information content	0.20	0.33	0.44	0.52	...	0.97	...	1

Figure 5.6: Nearest Neighbour Classification Accuracy (%) For Various Spaces (Cows)

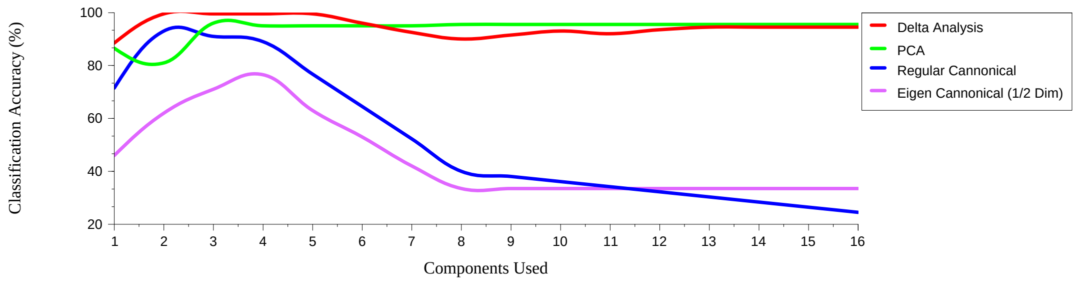


Figure 5.7: Nearest Neighbour Classification Accuracy (%) For Various Spaces (Triangle People)

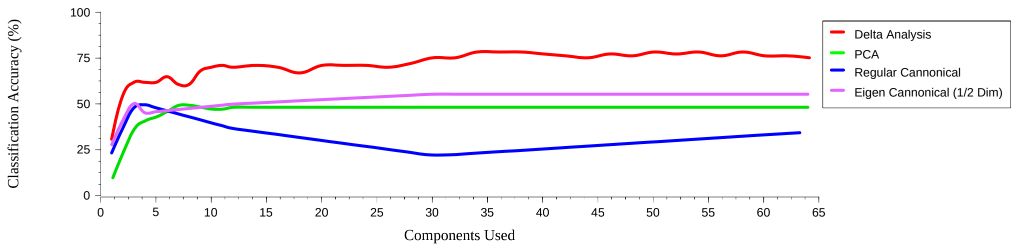


Figure 5.8: Nearest Neighbour Classification Accuracy Using Euclidean Distance (%) For Various Spaces (Cow Outlines)

Space Used	No. Components Used				Max.
	1	2	3	4	accuracy
Approx. Delta BC Space (Mahalanobis)	21	42	52.5	60.5	92.5[115]
PCA (Mahalanobis)	15	36.75	49.75	57	94 [79]
PCA (Euclidean)	15	36.3	51.8	58.5	90.3 [71]
Eigen-Canonical (Dimensionality = 100)	11	29	38.8	50	74.8 [20]
PCA+Delta BC Space (Mahalanobis)	2.3	1.5	3	4	93 [197]
PCA+BC Delta Space (Euclidean)	2.3	1.5	2.8	3.5	91.3 [188]
Raw Data	-	-	-	-	90.3 [10304]

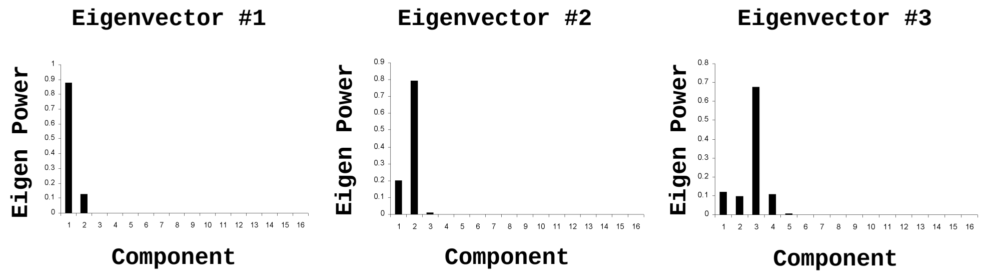
Note: Value in square brackets is no. of components used.

Figure 5.9: Nearest Neighbour Classification Accuracy (%) For Various Spaces (Face Data)

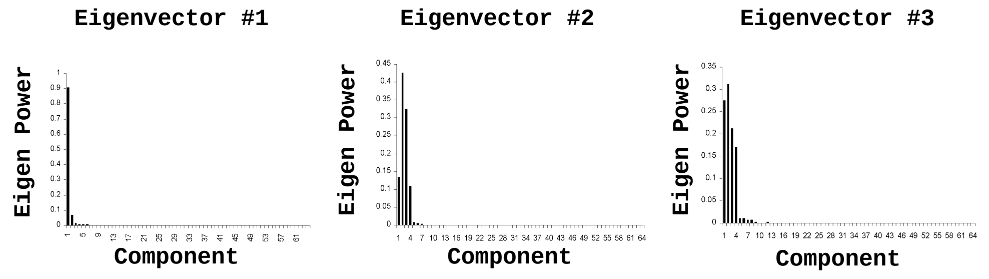
Figures 5.5 to 5.9 show that, when using the first few components only, the between class separation optimised spaces produced using Delta Analysis perform better than all other methods at classifying the data for all the data sets used. This indicates there is better inter-class separation in these spaces. When increasing numbers of components are used there is generally an increase in the performance of the PCA spaces and a decrease in the performance of the Canonical analysis based spaces. This is due to the fact that adding components to the PCA space generally increases the intra-class information contained in the space (this information is spread through all components) whereas adding components to Canonical analysis based spaces may add little intra-class information (as this is mostly concentrated in the first few components). This argument would lead us to expect a drop off in the performance of the Delta Analysis based spaces, however the drop off observed is only marginal. This leads us to conclude that either; i) the separation in the first few components is so good it swamps the contribution of later components to the classification or ii) some inter-class variation is included in the later components due to similarity between elements of inter- and intra-class variation. In reality this effect is probably a combination of these two factors and it should be noted that Delta Analysis works best when inter- and intra-class variation have a high degree of spatial separation in terms of the ‘real world’ components they effect.

5.3.2 Evaluation of Intra-class Separation

The intra-class space produced by delta analysis was compared to spaces produced using PCA on single class data. The ‘modes of variation’ in these spaces were observed to be qualitatively similar. To evaluate this the first three eigenvectors of the intra-class space were projected into these single class PCA eigen-spaces. The results, shown in figure 5.10, demonstrate that these eigenvectors lie almost exclusively within the subspace defined by the first four PCA eigenvectors and (for the triangle people especially) are composed mostly of a single PCA eigenvector.



a)



b)

Figure 5.10: Mean of Projections of Eigenvectors into Single Class Spaces a) Triangle People b) Cows

5.3.3 Evaluation of Combined Representation

Both real and synthetic data were approximated using truncated delta and standard PCA spaces to evaluate the trade off between data compression and accuracy in these two schemes. The mean vector error was calculated and normalised by the average vector size over all data sets. Delta space approximations were performed with equal numbers of inter and intra class components (i.e. 2 components = 1 intra and 1 inter class component).

Tables 5.11.a-c show that the approximation accuracy for a delta space approximation is comparable but marginally worse than the equivalent eigen-space approximation with the same number of components. This is as would be expected as the eigen-space representation is optimised for dimensionality reduction. This reduction in dimensionality reduction must however be traded against the advantages of class separation.

a)

Components Used	2	4	6	8	16
PCA Eigen-space	11.4[11.3]	3.8[3.6]	2.4[2.2]	1.5[1.2]	0[0]
Delta Eigen-spaces	14.9[13.0]	10.3[10.2]	8.3[9.3]	7.8[9.1]	0[0]

b)

Components Used	2	4	6	8	16	32	64
PCA Eigen-space	18.3[16.7]	15.4[14.2]	13.9[11.8]	12.4[10.2]	8.6[6.9]	4.4[3.6]	0[0]
Delta Eigen-spaces	19.0[18.1]	17.5[16.4]	16.4[14.9]	15.6[14.2]	12.5[11.0]	8.5[7.4]	0[0]

c)

Components Used	10	20	50	100	200
PCA Eigen-space	16.6 [14.4]	13.9 [12.5]	10.1 [9.3]	6.7 [6.1]	0 [0]
Delta Eigen-spaces	18.9 [15.9]	16.5 [14.2]	13.1 [11.5]	9.7 [8.8]	5.8 [5.6]

Figure 5.11: Mean and Standard Deviation of Error Rates (%) For Different Approximations a) Triangle People b) Cows c) Faces

5.4 Separation and Quantisation of Variation in Livestock

The principal application of this thesis is the analysis of livestock. In the previous chapter it was described how 2D contour models of cow outlines were built. These model the entire range of variation within the class of objects as a single system encompassing inter-animal and intra-animal variation. It is desirable to separate variation into separate (meaningful) components for two reasons; for the increased efficiency of model fitting searches (in tracking applications) and for the increase in semantic information contained in the representation (for temporal modelling applications). The model fitting searches are more efficient because inter-animal characteristics are static over a sequence of a single animal and because the size of the search spaces is reduced (this is discussed in chapter 7). The semantic information contained in the representation is increased as components of the parameterisation map more closely to the degrees of freedom on the real system and because independent components are decoupled (and may be modelled separately over time).

In the example application implemented the variation is separated into three components; inter-animal, front legs and rear legs. This is achieved by applying Delta Analysis twice. The first application separates variation into inter- and intra-animal components grouping the data into classes by individual. The second application is based on data that is constructed from the intra-animal variation only. This is formed by projecting the data into the combined eigenspace representation (using a least squares optimisation technique) and projecting back into the vector domain using only the intra-animal components. This data set is grouped (by hand) into classes representing similar front leg positions. The front legs are chosen as there is more semantic information contained in them and, as such, they are easier to group. Delta analysis was applied which separates the intra-animal variation into front legs and rear legs components. It is interesting to note that this method defines head position as part of inter-animal variation as this does not alter significantly in the sequences used. There is an argument for separating out head position as a separate form of variation in a more general system (as it is in fact an intra-animal characteristic), however this was not necessary in the system implemented for evaluation purposes due to the data sequences used.

The methods described in chapters 6 and 7 require a quantised model. It is possible to construct such a quantised model from the model described in this chapter by performing Vector Quantisation (VQ) on each of the eigenspaces of the model. Various methods for performing VQ are discussed in appendix E.

5.5 Discussion and Conclusions

It has been shown that delta analysis can outperform PCA, canonical analysis and combined PCA and canonical analysis (CST) in simultaneously producing optimal inter-class separation and dimensionality reduction. The intra-class optimised spaces produced by delta analysis are qualitatively similar to spaces produced by performing PCA on single class data sets assuming that intra-class variation is comparable across data sets. It has also been shown that subsets of the two parameterisations produced may be combined to form a powerful approximation to the data which may be used in efficient object tracking applications.

It has also been shown that canonical analysis and delta analysis do not always perform as well on the modified data set produced using PCA as on the raw data. In the eigenface data these methods applied to PCA data actually perform worse than PCA alone for the initial modes. The reason for these observations is that these methods work best when physical modes of variation are well de-coupled in the data representation. The eigen-space transform performed by PCA can, under certain circumstances, make the task of class separation harder by producing mixed modes which are combinations of two or more physical modes of variation. This is not to say this two stage approach is not valid as it may reduce the computational cost of creating and working with these inter and intra-class optimised spaces which is important in many applications.

The delta analysis approach is applicable to a wide variety of tasks including object recognition using outlines or grey-levels and deformable object tracking as well as non machine vision applications where a multivariate classifier is required. Delta analysis is particularly applicable to the principal application of this thesis (livestock tracking and analysis), allowing the variation within this object class to be separated into semantically meaningful independent components.

Chapter 6

Building Temporal Models of Object ‘Behaviour’

In this chapter various temporal modeling and prediction schemes are evaluated. The purpose of this is twofold; firstly a predictor is required for the stochastic object tracking scheme described in chapter 7 and secondly a method is required to classify object ‘behaviours’. In this thesis the term ‘behaviour’ is used to describe a sequence of spatial states which convey information about object activity and help in the separation of such activity into classes (for example normal or lame walking motions).

6.1 Choice of a Temporal Modelling Scheme

Related work on temporal modelling has been described in some depth in chapter 2. Livestock gait is a complex, non-linear cycle of movements and as such *a priori* knowledge must be built into any scheme which models these motions, as simple assumptions such as linear motion or constant velocity are invalid. Methods that include no *a priori* knowledge, such as Kalman filters, are thus unsuitable for these purposes. The choice that is left is then of using ‘hand crafted’ or ‘learned’ models of motion. It would be possible to hand build models of the dynamics of livestock from physiological studies on the constraints imposed by bone structure and muscle, however these would require a great deal of work and would only be relevant to the class of animals for which the study was undertaken. These studies are also beyond the scope of this thesis. For these reasons it was decided to use a temporal modelling scheme which could learn dynamics from observations of the objects in question (livestock in our case). Such a scheme would be more generally applicable in animal monitoring and other areas than a scheme based on physiological studies. There is also no clear way of mapping existing physiological studies onto the two dimensional spatial models described in previous chapters.

The techniques described in chapter 2 that belong to the category 'learned models' may be divided into two groups; 'Markov methods' and 'History Representation Classification' methods. In addition there are also hybrid methods such as the method described by Johnson *et. al* [62] which combine these two types of techniques. Within the livestock data sets used there is a fairly large amount of variation and it is probable that a practically sized training data set may not encapsulate 100% of the possible dynamics. For this reason it is important that the modelling scheme used has the ability to generalise and make a sensible prediction or classification for an unseen sequence. As described in chapter 2, Markov Chains have no ability to generalise, and thus are unsuitable (on their own) to model livestock dynamics, however other methods such as Hidden Markov Models (HMMs) and the 'History Representation Classification' methods (e.g. [13, 62]) do have the power to generalise and so are suitable for the task at hand.

If the multiple model spatial modeling scheme described in chapters 3-5 is to be used, the scheme chosen must deal with prediction between multiple continuous representation spaces. This may be achieved in two ways;

1. Using a multiple temporal model scheme with a continuous temporal model for each spatial model and a discrete temporal model to model the changes between spatial models.
2. By quantising each spatial model into a set of states and modelling the state transition probabilities.

The complexity of approach 1) increases with the number of spatial models used and the continuous temporal models must be re-initialised on each transition between spatial models. For these reasons approach 2) was chosen.

Two schemes have been implemented and evaluated as predictors for the livestock data. The first is a novel ‘History Representation Classification’ scheme with similarities to the model of Sumpter and Bulpitt [103] and the second a novel Hidden Markov Model architecture known as ‘Multi Stream Cyclic Hidden Markov Models’. For comparison Markov chains are also evaluated.

6.2 A Novel ‘History Representation Classification’ Scheme

A scheme has been developed which is based on Sumpter and Bulpitt’s [103] idea of a ‘history space’ based on a ‘winner takes all’ approach to spatial classification combined with a temporal classifier to make predictions about future events from this space (see chapter 2 for a full description of this scheme). The ‘winner takes all’ approach is not optimal as it makes the scheme more susceptible to false spatial classification and quantisation errors, however the unavailability of an error measure between spatial model representations precludes the use of a ‘distance from prototypes’ scheme such as that used by Johnson *et. al.* [62].

The scheme presented is designed to work in conjunction with an object tracking scheme based on a quantised model (see chapter 7) and as such there is no need for an explicit spatial classifier within the temporal model as in the Sumpter and Bulpitt scheme. A single discrete spatial classification is taken from the object tracker as the ‘winning state’ and a history space is formed. Each dimension of this space corresponds to a spatial state and the value of this parameter equates to the reciprocal of the time since a state last ‘won’. This formulation was used rather than Sumpter and Bulpitt’s leaky integrator space as there is a clearer and simpler relationship between a sequence of states and a history vector. There is no need to assign a time constant as in that method and the calculation is faster. This representation (as with the Sumpter and Bulpitt representation) is approximately frame rate independent and thus a model trained at one frame rate may be used at a (slightly) different frame rate. No evaluation of different history representations has been carried out as this scheme is sufficient for the data sets covered within the scope of this thesis. It is hypothesised that different representation schemes will work well for different data sets depending on the length and nature of the history required to make a prediction (for example the scheme presented here will not work well if a very long history is required to make a prediction).

Principal components analysis (PCA) is performed on a training set of vectors in the history space and the resultant Eigenspace truncated to reduce the dimensionality of the history vectors. This increases the speed of prediction greatly as temporal classification calculations in this reduced dimensionality are much faster. Experimental results (presented later) indicate a high degree of truncation is possible without appreciable loss in prediction and classification ability. For the temporal classification Gaussian mixture models are used rather than the vector quantising neural networks used by Sumpter and Bulpitt, as the variation in the history space for a given next state for our data is not a simple distribution (as can be seen in figure 6.3.a). A flow diagram of the scheme is given in figure 6.1 below.

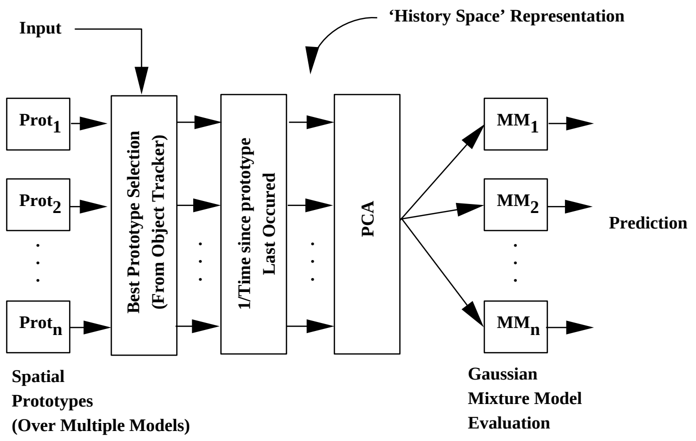

Input
'History Space' Representation
Prot1
Prot2
Protn
Best Prototype Selection (From Object Tracker)
1/Time since prototype Last Occured
PCA
MM1
MM2
MMn
Prediction
Spatial Prototypes (Over Multiple Models)
Gaussian Mixture Model Evaluation

Figure 6.1: The New Temporal Modelling Scheme

It should be noted that, in addition to the previously mentioned differences between this scheme and the Sumpter and Bulpitt scheme, there is no feedback loop. In this scheme the low dimensionality mixture models have the power to generalise and thus give a sensible prediction for unseen histories (this is demonstrated in the evaluation). There is therefore no need to force the history into a previously observed state using feedback. The spatial classification also comes from an object tracker rather than a set of neural networks and as such it must be assumed that this has some degree of correctness otherwise performing a prediction from this is not a sensible course of action.

6.2.1 Building Gaussian Mixture Models

There are two approaches that may be taken to build a Gaussian mixture model of a probability density function from a training set of multivariate data points (histories in our example). The first is the 'kernel method' or 'adaptive kernel method' in which each data point in the set of training examples is modeled by a single Gaussian. In the 'kernel method' the covariance and weight of each Gaussian is the same, the means being equal to the data points. The probability density function for the kernel method is given in equation 6.1 below.

$$P(x) = \sum_{i=1}^{N} \frac{1}{Nh^d} K(\frac{x - x_i}{h}) \tag{6.1}$$

Where:

$N =$ No. of Data Points

$K() =$ Gaussian Kernel Function (covariance equal to that of data points)

$h =$ Window Width

$d =$ Dimensionality of the data points

The optimal window width (h), in the sense of minimising mean integrated square error, may be determined using cross-validation. This is discussed by Silverman [96]. For a multivariate normal kernel (of which the multivariate Gaussian is the general form) the window width is given by equation 6.2.

$$h = \{4/(2d+1)\}^{1/(d+4)} \tag{6.2}$$

The adaptive kernel method is based on the kernel method but allows the scales of the kernels (Gaussians) in the mixture model to differ. Essentially, broader kernels are used where the density of the data points is low. This is achieved by constructing an initial mixture ($p'(x)$) model using the kernel method and calculating a local bandwidth factor (λ_i) for each Gaussian as in equation 6.3.

$$\lambda_i = (p'(x_i)/g)^{\frac{1}{2}} \tag{6.3}$$

Where:

$g =$ The geometric mean of $p'(x_i)$

The adaptive kernel PDF is then calculated as in equation 6.4.

$$P(x) = \frac{1}{N}\sum_{i=1}^{N}(h\lambda_i)^{-d}K(\frac{x-x_i}{h\lambda_i}) \qquad (6.4)$$

These methods result in models with a large number of Gaussian mixtures which can take considerable computational expense to evaluate. An alternative approach is to build a mixture model with fewer Gaussian mixtures than training data points which may be done using the Expectation Maximisation (EM) algorithm [29]. This is a two stage iterative scheme which computes the local maximum likelihood fit of an arbitrary number of Gaussians to a data set. This scheme is heavily dependent on how the Gaussians are initialised (this is not covered by the EM algorithm) and can lead to an over specific mixture model (see figure 6.3). Another drawback of this scheme is that, if the training data is locally sparse, mixtures may become singularities. Cootes and Taylor [20] present a scheme which combines the kernel method with the EM algorithm by altering the M step as in table 6.2. In this scheme singularities never occur and the result is less dependent on the initialisation of the mixture model.

Standard EM Algorithm	**Cootes and Taylor Modified EM Algorithm**
E-step: Compute the contribution of the i_{th} sample to the j_{th} Gaussian $p_{ij} = \frac{w_j G(\mathbf{x}_i:\mu_j,\mathbf{S}_j)}{\sum_{j=1}^{m} G(\mathbf{x}_i:\mu_j,\mathbf{S}_j)}$ **M-step:** Compute the parameters of the Gaussians $w_j = \frac{1}{N}\sum_{i=1}^{N} p_{ij}$, $\mu_j = \frac{\sum_{i=1}^{N} p_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} p_{ij}}$ $\mathbf{S}_j = \frac{\sum_{i=1}^{N} p_{ij}(\mathbf{x}_i-\mu_j)(\mathbf{x}_i-\mu_j)^T}{\sum_{i=1}^{N} p_{ij}}$	**E-step:** Compute the contribution of the i_{th} sample to the j_{th} Gaussian $p_{ij} = \frac{w_j G(\mathbf{x}_i:\mu_j,\mathbf{S}_j)}{\sum_{j=1}^{m} G(\mathbf{x}_i:\mu_j,\mathbf{S}_j)}$ **M-step:** Compute the parameters of the Gaussians $w_j = \frac{1}{N}\sum_{i=1}^{N} p_{ij}$, $\mu_j = \frac{\sum_{i=1}^{N} p_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} p_{ij}}$ $\mathbf{S}_j = \frac{\sum_{i=1}^{N} p_{ij}[(\mathbf{x}_i-\mu_j)(\mathbf{x}_i-\mu_j)^T+\mathbf{T}_i]}{\sum_{i=1}^{N} p_{ij}}$

Figure 6.2: EM Algorithms to fit a mixture of m *Gaussians to* N *samples $\mathbf{x}_i$. $\mathbf{T}_i$ is the kernel covariance of a sample calculated using the adaptive kernel method.*

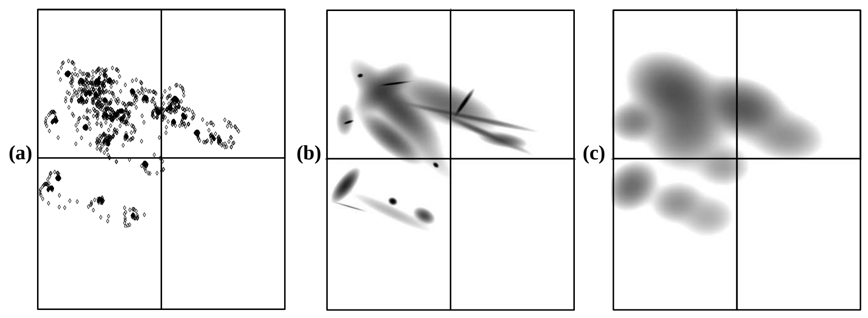


Figure 6.3: Building Gaussian Mixture Models: (a) Raw Data (b) Standard EM Algorithm (c) Cootes and Taylor Modified EM Algorithm

The Cootes and Taylor modification of the EM algorithm is used to build mixture models in the novel scheme implemented. In this scheme mixture means are initialised uniformly across the range of the data removing mixtures for which the mean is the nearest mixture mean to no data point in the training set. Initial mixture covariances are calculated from the covariance of the training set, the number of Gaussians and the dimensionality as in the kernel method [96]. The initialisation is such that more mixtures than required are used. This results in mixtures becoming co-incident (having equal mean and covariance) during the course of the EM iterations. Under such circumstances the scheme is modified to merge these mixtures after the M stage of the iteration. This results in a mixture model with an identical density description which is computationally more efficient to evaluate. The choice of the initial number of mixtures is important as this must be such that there are more mixtures than required. This may be determined by running the algorithm and seeing if mixtures are merged. If there is a significant drop in the number of mixtures ($>$10%) over the course of the algorithm run, it may be considered that enough mixtures have been used. Using too many mixtures is not observed to be a problem but adds computational expense to the method.

6.2.2 Comparing Mixture Model Densities

The scheme described previously produces a set of Gaussian mixture models which represent probability density functions (PDFs) within the history space. To make a prediction from this set of models that a newly observed history will result in a specific next state, the relative probability that the history comes from each model distribution must be determined. Statistically the probability of a continuous distribution resulting in a particular sample value is zero. To obtain a non-zero probability (P) the PDF (ϕ) must be integrated over some area local to the sample. Over a suitably small local area this is approximately equal to the multiple of the PDF magnitude at the sample and this area (δA_l) as in equation 6.5.

$$P(\overline{x}) = \int_{\delta A_l} \phi(\overline{x})\delta a \approx \phi(\overline{x}) \times \delta A_l \tag{6.5}$$

To the author's knowledge there is no statistically correct method to determine size of the local area (δA_l) over which to integrate, as this will depend on the range of, and magnitude of variation within the PDF. It can be shown however that it is valid to use the same area if the probability that a sample comes from either of two distributions is equal when considering the complete range of the data (see Appendix B for a proof of this). In the examples used there is significant variation in the range of the model PDFs and as such it would be incorrect to use the same value of δA_l for all models. As no scheme is available for the selection of δA_l, the approach of calculating uniform distributions with roughly similar ranges to the mixture model PDFs (ϕ_U) was taken. Examples of this can be seen in figure 6.4.

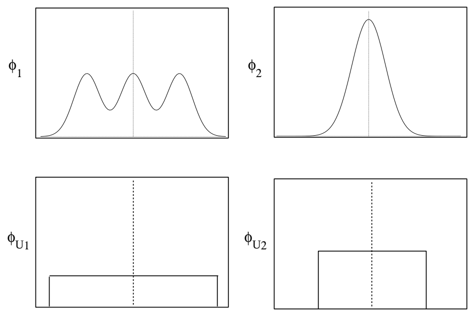


Figure 6.4: Comparing PDFs with uniform PDFs with the same Range

It is hypothesised that these distributions are similar enough in range and variation to use the same value of A_l for each pair of distributions. The relative probability that a sample results from a mixture model distribution or the corresponding uniform distribution can thus be calculated as in equation 6.6.

$$P_{Rel(Uniform)}(\bar{x}) = \frac{\phi(\bar{x}) \times \delta A}{(\phi(\bar{x}) \times \delta A) + (\phi_U(\bar{x}) \times \delta A)} = \frac{\phi(\bar{x})}{\phi(\bar{x}) + \phi_U(\bar{x})} \tag{6.6}$$

The relative probabilities obtained in equation 6.6 can be compared across mixture models and as such relative probabilities of a history resulting in a particular next state may be calculated as in equation 6.7.

$$P_{Rel}(\bar{x}_i) = \frac{P_{Rel(Uniform)}(\bar{x}_i)}{\sum_{n=1}^{N} P_{Rel(Uniform)}(\bar{x}_n)} \tag{6.7}$$

This method of comparison gives significantly more reliable prediction for seen and unseen data than simple comparison of PDF heights (for the example data sets).

6.2.3 Extension to a Multiple Input System

It is sometimes desirable to make a prediction from multiple input vectors. In the cow example the multiple vectors are the front and rear legs Eigenspace shape description components. These may be combined within the History Representation Classification (HRC) scheme either by combining the two Eigenspaces before vector quantisation or combining the resultant history representations after vector quantisation (but before PCA). The latter method was used in the predictor implemented as this results in a ‘history space’ with much lower dimensionality than the former method. The former method produces a higher dimensional space as every combination of front and rear legs positions must be represented by a different prototype where as the chosen method only requires one prototype for each front and each rear leg position. This extension is referred to as a ‘two space HRC’ in the evaluation in section 6.5.

6.3 Multi Stream Cyclic Hidden Markov Models

Hidden Markov Models were introduced in chapter 2 as a method of modelling stochastic time dependent processes using a Markov chain and a set of observation probability distributions. Cyclic

Hidden Markov Models use a hidden state architecture in which the first and last states are joined. This is illustrated in figure 6.5.

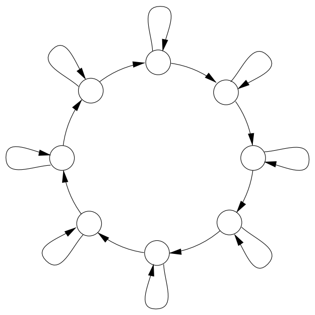


Figure 6.5: Cyclic Hidden Markov Model Hidden State Architecture

The Multi-stream Cyclic Hidden Markov Model (MSCHMM) is a Cyclic Hidden Markov Model with multiple sets of (discrete) visible state probability distributions which model observationally independent but related features of a system. In the cow example the front and rear leg pairs are modelled as two separate 'observational streams' based on a single underlying hidden state system. It should be noted that continuous (e.g. Gaussian mixture model based) observation probability distributions could be used, however these would be computationally more expensive to evaluate and as such are not used in the system implemented.

Hidden state transition probabilities are initialised by defining parameters P_s, P_c and P_x as the probability that the next hidden state remains the same, changes, or the current state is the last state in the sequence respectively. If it is assumed initially that these parameters are the same for each hidden state, this results in the probability distribution given in equation 6.8 for the cycle length in states.

$$P(n) = {P_s}^r P_x {P_c}^N \times {}^{(N+r-1)}C_r = P_s^r P_x P_c^N \times \frac{(N+r-1)!}{(N-1)!r!} \tag{6.8}$$

Where:

n	$=$	The Sequence Length
$P(n)$	$=$	The probability of a sequence of length n resulting from the CHMM
N	$=$	The No. of Hidden States in the MSCHMM

r = The No. of Hidden State Repetitions (n-N)

Training sequences are parsed into cycles (of various lengths) based on spatial model transitions. The number of hidden states (N) is selected to be the minimum cycle length in the training set. Values of N below this value may be used giving a MSCHMM with more generality and less specificity. Values of N above this value are not used as this would not allow generation of all training sequence lengths. An initial estimate of P_x as the reciprocal of the average sequence length is used and given $P_c + P_s + P_x = 1$ an exhaustive one dimensional search is performed in P_c (with fine quantisation) to minimise the square difference between the theoretical probability distribution of cycle length ($P(n)$) and a distribution calculated from the training sequences.

Initial visible state probability distributions are estimated by aligning each training sequence cycle with the set of hidden states, 'time stretching' such that the training sequence length is equal to the number of hidden states in the MSCHMM. Probabilities are estimated from the relative number of observations lying completely or partially over each hidden state as shown in figure 6.6.

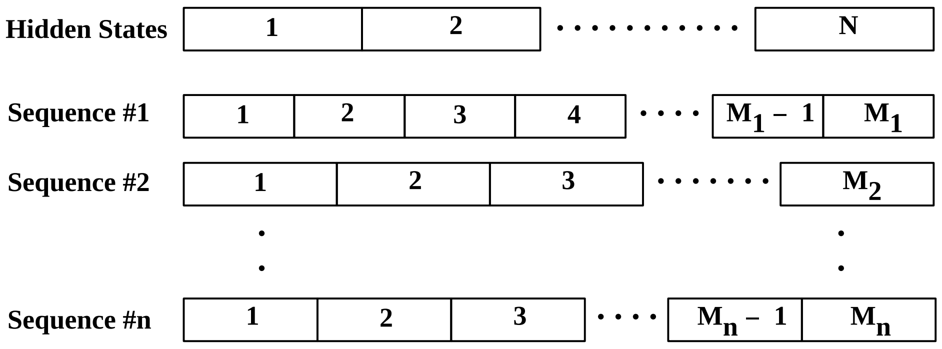


Figure 6.6: Initial Estimation of Observation Probabilities

The initial transition and observation probability estimates are improved using the Baum-Welch re-estimation method [85].

6.3.1 Prediction of Future States Using HMMs

The prediction of future states with HMMs is a two stage process. Firstly the current hidden state (or a probability distribution for this) must be determined. The Viterbi algorithm [112] (See Appendix C) may be used to determine a probability distribution for the current hidden state from the last N observations (where N is the order of the prediction). In chapter 7 an alternative (and more computationally efficient) approach is used to hidden state probability modelling, however the Viterbi algorithm is used for the evaluation within this chapter as it is more generally applicable.

Once the hidden state probability distribution has been determined the observation state probability distribution for the next timestep may be evaluated from the hidden state transition probabilities and the observation probabilities of the HMM as in equation 6.9.

$$P(O_{t+1} = o_n | O_t = o_{m0}, O_{t-1} = o_{m1}, \ldots) = \sum_{I=1}^{M} \sum_{j=1}^{M} P(q_t = h_i | O_t = o_{m0}, O_{t-1} = o_{m1}, \ldots) \times a_{ij} \times b_j(n) \tag{6.9}$$

Where:

O_t	$=$	Observation at time t
M	$=$	The number of hidden states in the HMM
q_t	$=$	The hidden state of the HMM at time t
a_{ij}	$=$	The hidden state transition probability between states i and j
$b_j(n)$	$=$	The probability of observation n assuming hidden state j

For multi-stream HMMs equation 6.9 must be evaluated for each set of visible states, although $P(q_t = h_i | O_t = o_{m0}, O_{t-1} = o_{m1}, \ldots) \times a_{ij}$ (the hidden state probability distribution at the next timestep) need only be evaluated once as it is the same for all observation streams. This calculation is further simplified for cyclic HMM architectures as there are only two possible hidden states at the next timestep for a given hidden state (a_{ij} for all other hidden state transitions is zero).

6.4 Prediction Results

Qualitatively the two predictor schemes described in this chapter produce similar results and so their relative merits must be evaluated quantitatively (see section 6.5). For visualisation purposes the output of a prediction scheme may be fed back into the input to create a ‘realistic’ virtual temporal sequence. Examples of such sequences are given in figures 6.7 and 6.8.

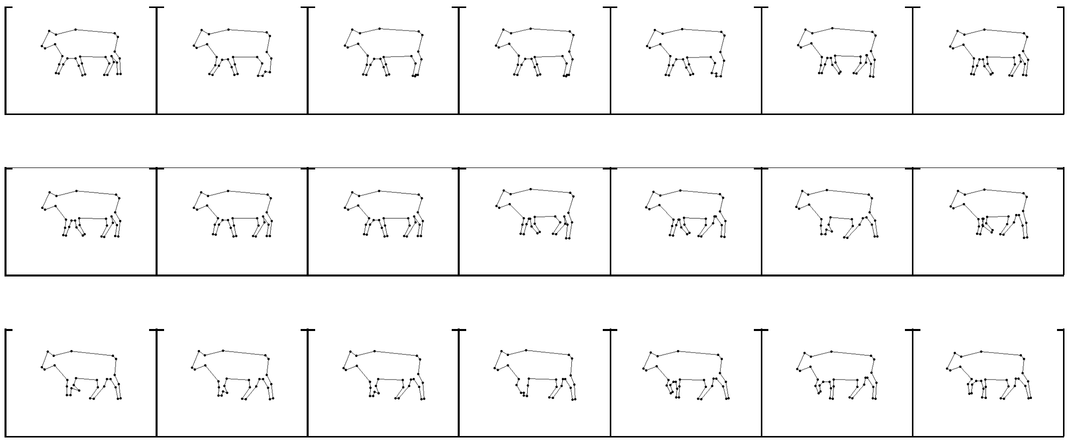


Figure 6.7: Virtual Temporal Sequence of a Cow Walking Created Using HSC Model

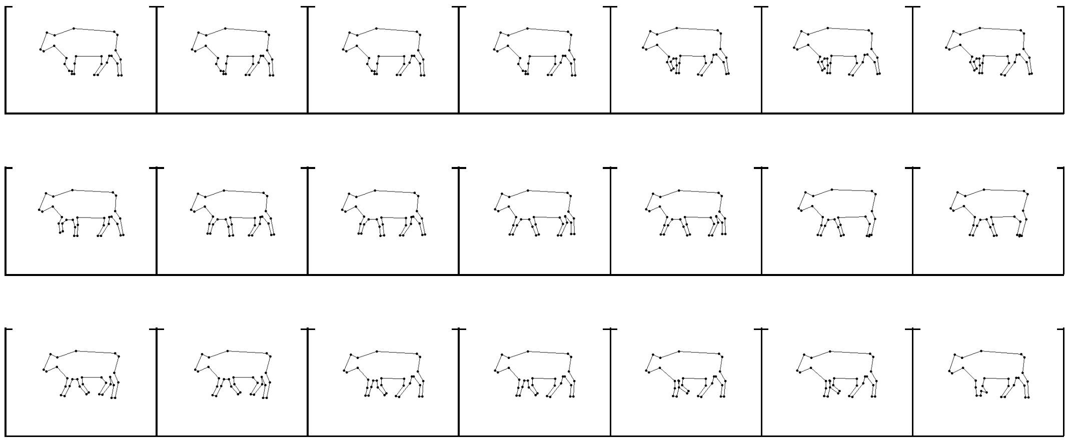


Figure 6.8: Virtual Temporal Sequence of a Cow Walking Created Using MSCHMM Model

6.5 Evaluation of Predictor Schemes

There are many metrics and tests that may be used to evaluate a predictor scheme, however these may be broadly divided into two types; success measures and failure measures. Success measures (such as average error) measure how well a predictor predicts future states for unseen data and failure measures (such as the proportion of erroneous predictions) measure the robustness of the prediction scheme. It is important to evaluate a predictor scheme using both success measures and failure measures to obtain an accurate description of its properties.

In the spatial modelling scheme (see chapters 3-5) there are multiple spatial models and thus a simple error measure is not always available, however the proportion of correct model predictions and the errors within the set of correct predictions may be examined. An alternative approach is to evaluate the scheme as a finite state predictor, looking at prediction probabilities for newly observed states. In this application multiple next states are possible from a given history and as such a unit (100% probability) prediction for the next observed state is not what is desired. The predictor should assign a ‘significant probability’ to any next state that has a ‘reasonable probability’ of actually occurring. It is difficult to determine what a ‘significant probability’ or a ‘reasonable probability’ should be as this depends on the nature of that particular prediction.

In this evaluation it is taken that any new state observed has a ‘reasonable probability’ of being observed (as it has been observed) and any probability greater than the probability given by a uniform predictor is a ‘significant probability’. From this, a measure ‘Probability Drop Out’ (PDO) is defined that is the proportion of unseen future states that are assigned a probability of less than that given by a uniform predictor. This is an important criterion as predictors with a significant ‘Drop Out Rate’ (DOR) are unsuitable for use in Condensation [57] type algorithms such as the scheme described in chapter 7, as low probability events will be evaluated with low probability resulting in erroneous tracking if they occur and are not evaluated. As such this failure measure is more important than any success measure.

As a measure of success the mean and standard deviation of predicted probabilities for each next state observed are examined. These are broken down by the next state observed as these predictions should have similar context (average results are shown). All results shown are from “leave 2 out tests” from a set of 12 sequences of healthy walking cows (approximately 1000 frames). The results for first and second order Markov chains are also presented for comparison.

Predictor	**Dimensionality**	**Drop Out Rate**
Single Space HRC	1	0.079
Single Space HRC	2	0.038
Single Space HRC	3	0.079
Single Space HRC	4	0.085
Two Space HRC	1	0.07
Two Space HRC	2	0.065
Two Space HRC	3	0.085
Two Space HRC	4	0.103
MSCHMM (1st order prediction)	N/A	0.061
MSCHMM (2nd order prediction)	N/A	0.078
MSCHMM (3rd order prediction)	N/A	0.064
MSCHMM (4th order prediction)	N/A	0.059
Single Space Markov Chain (1st order)	N/A	0.097
Single Space Markov Chain (2nd order)	N/A	0.221
Two Space Markov Chain (1st order)	N/A	0.252
Two Space Markov Chain (2nd order)	N/A	0.349
Uniform Predictor	N/A	0

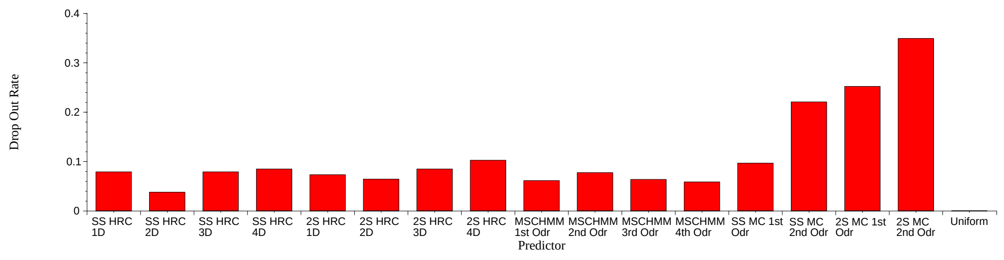


Figure 6.9: 'Drop Out Rates' for Front Legs Temporal Predictors

The main conclusion to be drawn from figures 6.9 and 6.10 is that both the History Representation Classification (HRC) and Multi-stream Cyclic Hidden Markov Model (MSCHMM) methods perform better than Markov Chains in terms of drop out rate. The only exception to this is is the four dimensional, two space HRC predictor which performs marginally worse than the first order single space Markov Chain predictor. As a general trend the drop out rate for the HRCs increases with dimensionality (after the first dimension) as the predictor becomes more specific and less

Predictor	Dimensionality	Drop Out Rate
Single Space HRC	1	0.129
Single Space HRC	2	0.126
Single Space HRC	3	0.167
Single Space HRC	4	0.191
Two Space HRC	1	0.082
Two Space HRC	2	0.111
Two Space HRC	3	0.164
Two Space HRC	4	0.155
MSCHMM (1st order prediction)	N/A	0.094
MSCHMM (2nd order prediction)	N/A	0.131
MSCHMM (3rd order prediction)	N/A	0.128
MSCHMM (4th order prediction)	N/A	0.130
Single Space Markov Chain (1st order)	N/A	0.176
Single Space Markov Chain (2nd order)	N/A	0.364
Two Space Markov Chain (1st order)	N/A	0.305
Two Space Markov Chain (2nd order)	N/A	0.326
Uniform Predictor	N/A	0

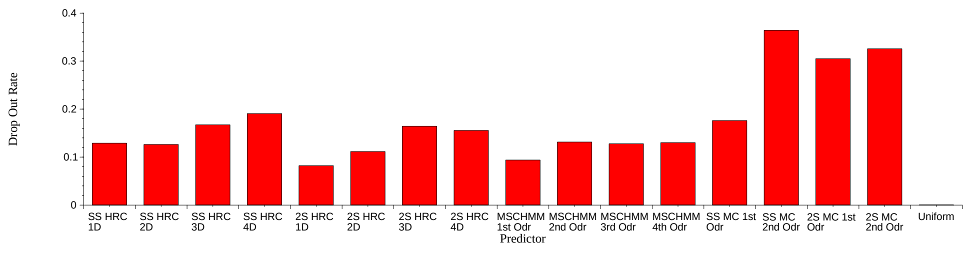


Figure 6.10: ‘Drop Out Rates’ for Rear Legs Temporal Predictors

general. The drop out rate for one dimensional HRCs is higher as they are over generalised and have poor prediction power. 2nd order Markov chains and Markov chains based on a two space input perform particularly badly as they have no ability to generalise to unseen inputs and as such can only predict from identical inputs in the training set. There is not much difference, in terms of drop out rates, between the MSCHMM predictors and the optimal HRC predictors and the effect of increasing MSCHMM prediction order is marginal (and only beneficial for the front legs predictor).

Predictor	Dimensionality	Av. Prediction	Av. SD of Prediction
Single Space HRC	1	0.092	0.030
Single Space HRC	2	0.173	0.065
Single Space HRC	3	0.187	0.065
Single Space HRC	4	0.195	0.083
Two Space HRC	1	0.079	0.020
Two Space HRC	2	0.180	0.062
Two Space HRC	3	0.182	0.064
Two Space HRC	4	0.188	0.074
MSCHMM (1st order prediction)	N/A	0.233	0.06
MSCHMM (2nd order prediction)	N/A	0.266	0.087
MSCHMM (3rd order prediction)	N/A	0.271	0.083
MSCHMM (4th order prediction)	N/A	0.275	0.084
Single Space Markov Chain (1st order)	N/A	0.313	0.126
Single Space Markov Chain (2nd order)	N/A	0.341	0.203
Two Space Markov Chain (1st order)	N/A	0.330	0.222
Two Space Markov Chain (2nd order)	N/A	0.197	0.221
Uniform Predictor	N/A	0.029	0

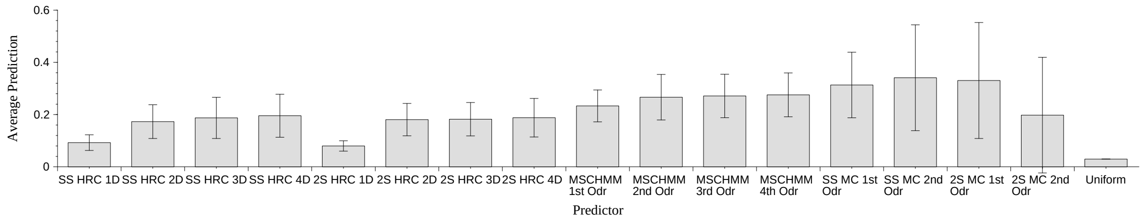


Figure 6.11: Average Predictions (+/- 1 Sd) for Front Legs Temporal Predictors

The results presented in figures 6.11 and 6.12 are very interesting. In terms of average predictions Markov Chains do the best (excepting the second order two space chain), however the standard deviations for these results are very high. The high standard deviations for the Markov Chains, along with the ‘Drop Out Rate’ results, illustrate that, although certain predictions are very high for the observed outcome, there is a significant proportion of events that are predicted very badly by Markov chains due to their lack of generality. Generally the MSCHMMs perform better than the HRCs, although prediction standard deviations are similar indicating that the two methods are similarly robust. Increasing the dimensionality of the HRC predictors increases the average prediction for the observed output (significantly between the first and second dimensions, marginally

Predictor	**Dimensionality**	**Av. Prediction**	**Av. SD of Prediction**
Single Space HRC	1	0.078	0.038
Single Space HRC	2	0.164	0.068
Single Space HRC	3	0.184	0.107
Single Space HRC	4	0.188	0.108
Two Space HRC	1	0.069	0.023
Two Space HRC	2	0.147	0.062
Two Space HRC	3	0.156	0.068
Two Space HRC	4	0.164	0.069
MSCHMM (1st order prediction)	N/A	0.199	0.053
MSCHMM (2nd order prediction)	N/A	0.221	0.091
MSCHMM (3rd order prediction)	N/A	0.228	0.074
MSCHMM (4th order prediction)	N/A	0.233	0.113
Single Space Markov Chain (1st order)	N/A	0.205	0.227
Single Space Markov Chain (2nd order)	N/A	0.319	0.219
Two Space Markov Chain (1st order)	N/A	0.300	0.220
Two Space Markov Chain (2nd order)	N/A	0.289	0.145
Uniform Predictor	N/A	0.028	0

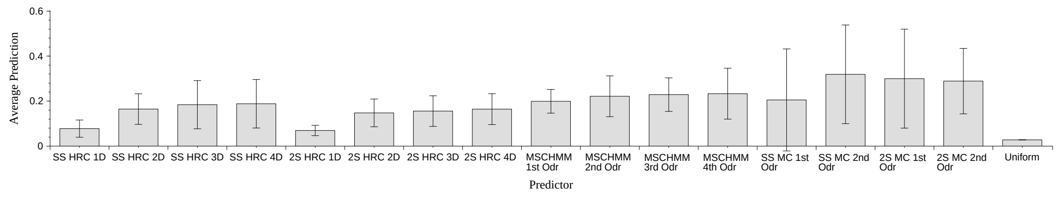


Figure 6.12: Average and Standard Deviation Predictions for Rear Legs Temporal Predictors

thereafter), but also increases the standard deviation. This is to be expected as the predictor becomes more specific (to the training set) and less general. As with the 'Drop Out Rate' results, increasing the order of the MSCHMM prediction only has a marginal effect, although this is beneficial in terms of average predictions for observed outcomes. It is unclear if there is a detrimental effect on the standard deviations (and as such the robustness of the predictor) from increasing the prediction order, however increasing the prediction order increases the computational cost of the prediction calculation and as such it is not advisable to increase the prediction order more than is necessary. For the data evaluated the benefit of predicting beyond second order is marginal and even first order predictions produce usable results in the scope of this thesis.

6.6 Computational Cost of Predictors

If the predictors are to be used in a ‘real-time’ system the computational cost (in terms of processor time) of the predictor is important. This is particularly important in particle filtering schemes such as the Condensation algorithm [57] where multiple predictions are made per timestep. The number of samples possible (and thus the accuracy) in such algorithms is partially determined by the speed of the predictor. Table 6.13 gives typical execution rates of the various predictor schemes on a relatively slow entry level workstation (SGI R5000 180MHz 96MB).

	HSCs				**MSCHMMs**				
	1D	2D	3D	4D	1st Order	2nd Order	3rd Order	4th Order	HS Known
Rate (no./sec)	900	350	135	91	1700 †	1667 †	1225†	1200 †	160,000 †

†MSCHMM predictions predict the output of the two streams simultaneously.

	Markov Chains			
	SS 1st Order	SS 2nd Order	2S 1st Order	2S 2nd Order
Rate (no./sec)	4350	65	65	⋆

⋆ 2 Space 2nd Order Markov Chain Evaluation takes several minutes due to high memory usage.

Figure 6.13: Evaluation Rates for Predictor Schemes

The results in figure 6.13 show that the fastest predictor performing a complete prediction (i.e. input to output) is the first order Markov chain. This is as expected as this is a very simple method. Of the other methods the MSCHMMs perform fastest, with the higher order and multiple space Markov chains performing worst of all. The higher order and multiple space Markov chains perform particularly badly due to their high memory requirement. The MSCHMM prediction from a known hidden state is faster than even the first order Markov Chain as it is based on a first order Markov Chain but only two possible next states need to be evaluated. This is not a complete prediction as it cannot predict from past observations (only from a known hidden state), however this scheme can be used within a particle filtering framework (e.g. Condensation [57]) as will be shown in chapter 7.

6.7 Discussion and Conclusions

In this chapter two novel temporal predictor schemes have been presented and their properties evaluated with reference to a widely used temporal prediction scheme (Markov Chains). Both schemes produce more robust predictions than Markov Chains due to their ability to generalise from training sequences to predict future observations for unseen sequences. The Multi Stream Cyclic Hidden Markov Model (MSCHMM) outperforms the History Representation Classifier (HRC) scheme in terms of average prediction ability, although the schemes are more or less equally robust. The MSCHMM scheme performed particularly well under evaluation as the test data used (sequences of walking cows) are very well modelled by the cyclic architecture of the scheme. This scheme also has the potential to produce faster predictions than any of the other schemes. The HRC scheme is more generally applicable as it is not constrained by any particular architecture (the data need not be cyclic), however Hidden Markov Models can take any architecture and there has been recent work on automatic architecture determination (e.g. [67]).

Modelling cow leg movements is an interesting problem as the positions of the two leg pairs are related (by the cyclic nature of the walking action) but have an independent component in their variation. Given this, it is not clear whether they should be modelled as a single system or as two independent systems. Results for the HRC predictors show that including information from both sets of legs marginally increases the robustness of the prediction (in terms of lower DOR and Prediction Standard Deviations) for rear leg predictions, however the benefits for front leg predictions are less clear. The reason for this is that there is more semantic information in the front leg position than in the rear leg position as the magnitude of variation within the front legs position is larger and thus less susceptible to noise caused by erroneous model fits (the front legs bend more). Given these observations it is desirable to model the legs as a single system to preserve the semantic information and exclude noisy training examples from the model as far as possible. The MSCHMM scheme is perfect for this purpose as the two distributions allow for an accurate specific description of front legs variation combined with a more general rear legs description within the same scheme. In this way semantic information may be extracted from rear leg position at points in the cycle where it is available without requiring a very specific model of rear leg variation that would not be able to match the wide range of variation observed.

In conclusion all of the predictor schemes evaluated in this chapter have their uses in particular circumstances. HRCs are a good general scheme which is not constrained by any particular data architecture. MSCHMMs model only cyclic data and can represent data with multiple observation streams well. MSCHMMs also have the potential for extremely fast prediction. Markov chains have their limitations (as has been shown in this chapter), however they can be extremely fast to evaluate and are probably the simplest temporal modelling scheme available.

Chapter 7

Object Tracking Using Stochastic Search Frameworks

Object tracking and model fitting methods have been discussed previously in chapter 2 where methods were divided into ‘global search’ and ‘local search’ methods. Global search methods use ‘intelligent’ sampling techniques to evaluate a large (or infinite) set of solutions to the problem of fitting a model to data. These methods may be iterative (e.g. Markov Chain Monte Carlo [25]) or use a single sample set (e.g. Condensation [57]). Global methods can be reasonably robust, however a large number of samples may be required when the tracking / model fitting problem is complex. Local methods use a single initial guess and improve on this iteratively by searching in the area local to the previous guess for a better solution. This search may be performed locally in the image domain, such as with ‘Snakes’ [66], or locally in an abstract model parameter domain, as with ‘Active Appearance Models’ [18]. These local search methods are efficient but prone to finding locally optimum solutions if the initial estimate is poor or the problem is complex (i.e. the solution space is not smoothly changing or contains large gradients in the quality of the solution).

In this chapter we present a ‘Global-Local’ framework for object tracking which combines global and local search methods in series. This has increased robustness over local search schemes but does not require the high number of samples required by global search methods for complex problems.

7.1 Overview of the ‘Global-Local’ Framework

The ‘Global-Local’ framework developed consists of three stages; i) Initialisation of scale and position, ii) Stochastic sampling and iii) Local search, as shown in figure 7.1. As such this is really a ‘Local - More Local’ framework as the search space of the Stochastic sampling stage (stage ii) is limited in scope by the initialisation, however the shape search is truly ‘Global-Local’ and this is how we will refer to this framework.

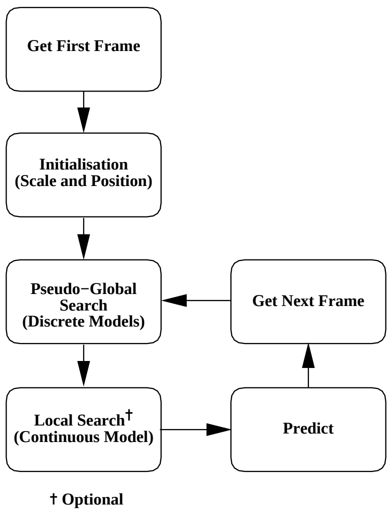


Figure 7.1: Overview of the ‘Global-Local’ Framework

7.1.1 Image Processing

There are tracking methods in the literature (for example Active Appearance Models [18]) that work on raw image data, however the majority of methods to date work in conjunction with some image pre-processing such as edge detection. Examples include ‘Snakes’ [66], Active Shape Models [19] and initial implementations of the ‘Condensation’ algorithm [57]. The reason behind using edges (or other features) is primarily to do with efficiency. Almost all model fitting / tracking methods use some form of sampling, and sampling features is far less computationally expensive than sampling an entire area of an image. Sampling with Active Appearance Models is very computationally expensive, however full use is made of the residual error vector in this scheme (rather than a single scalar magnitude value) which results in a viable scheme.

The methods described in this chapter are based on a pre-processed edge image primarily due to the edge based model used. Image colour/intensity based methods were rejected as, for the main application (livestock monitoring), objects exhibit significant variation in local colouring (i.e. marking patterns differ from one individual to another).

The pre-processed edge image used is created by statistically modelling the edges produced by a Canny edge detector over time. The mean and standard deviation of edge intensity is calculated for each pixel and new input images compared to these distributions. Any ‘unusual’ edges (i.e. intensity is greater than a fixed number of standard deviations above the mean) are classified as ‘moving edges’ and the image intensity set to a maximum. The resultant image is then blurred to aid with model fitting as in the ‘snakes’ algorithm [66]. This method is robust in extracting edges except when there is a strong edge in the background or the object to be detected remains still for a long time. The latter problem is solved by excluding edges classified as moving from the background statistics, although this requires a number of frames of object free background to initialise the statistics (typically 10-20 frames is enough). The former problem is partially alleviated by the blur stage that serves to fill small gaps in the contour, however this would be a problem on a very textured background and another method would be required. The former problem is not large within the context of the farmyard images that are used in this thesis. Figure 7.2 gives a typical example of this image processing.

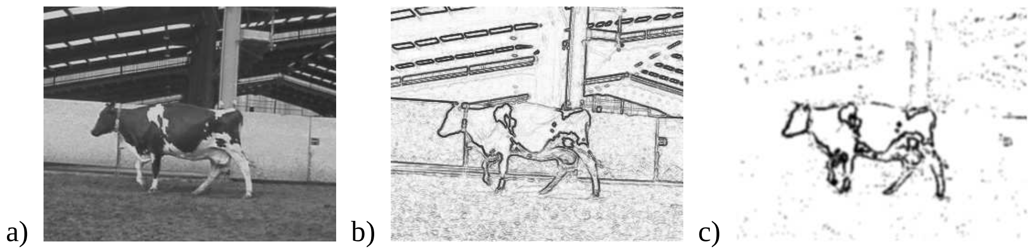


Figure 7.2: Extracting moving edges; a) Input image, b) Edge detection, c) Moving Edges

7.1.2 Initialisation of Scale and Position

As can be seen in figure 7.1, initialisation of scale and position is only required for one frame for each object to be tracked (the first frame in which the object occurs). Initialisation is carried out by clustering groups of ‘significant points’ and calculating the rectangular area enclosing these groups as in figure 7.3.

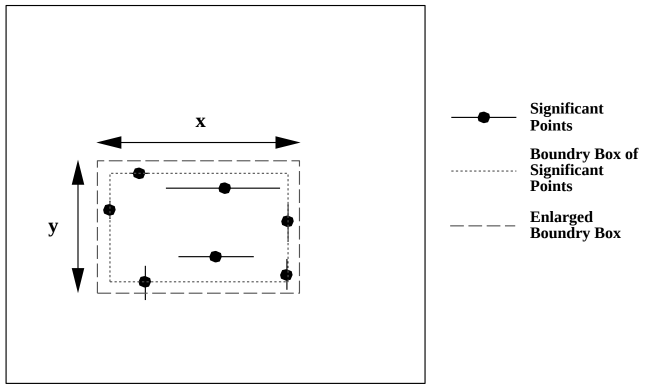


Figure 7.3: Calculating rectangular area covered by ‘significant points’

The enlarged bounding box shown in figure 7.3 is simply a pre-determined number of pixels larger than the bounding box to allow for the fact that significant points are not necessarily at the edge of the object. From the enlarged bounding box, and information about the average size of the objects, approximate position and scale may be calculated as in equations 7.1 to 7.3.

$$P_x = \frac{Max_X + Min_X}{2} \tag{7.1}$$

$$P_y = \frac{Max_Y + Min_Y}{2} \tag{7.2}$$

$$S = \frac{\frac{x}{\overline{X}} + \frac{y}{\overline{Y}}}{2} \tag{7.3}$$

Where:

$P_x, P_y =$ Initial position estimate

$S =$ Initial scale estimate

$\overline{X}, \overline{Y} =$ Mean size of objects (from training set)

The ‘significant points’ are chosen from the pre-processed image (see section 7.1.1) by the criteria that these points lie at the centre of a horizontal or vertical line with a given minimum length and intensity (see figure 7.3). The values of minimum length and intensity chosen are such that small areas of noise are not selected, but a representative selection of samples is taken from areas where objects are present. Values of 3 pixels for minimum line length and half of maximum intensity for minimum intensity are used in the application implemented.

When there are multiple objects present in the scene the ‘significant points’ are initially coarsely divided by screen area and a single iteration of the K-means clustering algorithm performed. Points associated with (i.e. falling within the bounds of) objects existing in previous frames are excluded from this process. In the application implemented this is sufficient to separate out objects and noise. Simple statistics (mean and standard deviation of object size) are used to reject objects that are obviously caused by noise.

This method is by no means the only method by which an initial guess may be made (for example background subtraction could be used, as by Baumberg and Hogg [3]), however it is reasonably fast ($>$27 fps on an entry level SGI workstation; R5000 180MHz 96MB excluding image processing) and, for this application, robust in giving a suitable initial estimate of scale and position. Some results from this ‘Position tracker’ are shown in figure 7.4.

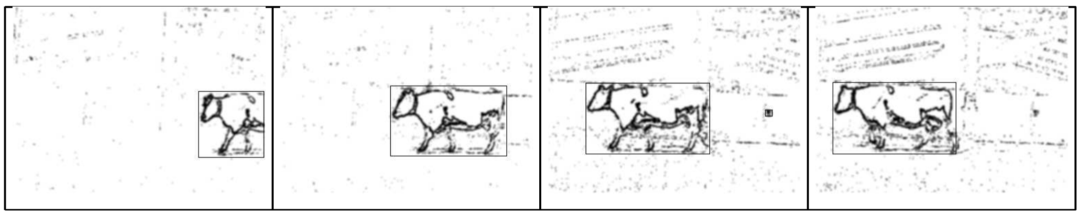


Figure 7.4: Object Location Using a Simple Position Tracker

7.1.3 Stochastic Sampling Using a Discrete Model

The Active Shape Model [19] constrains shape search by limiting the Mahalanobis distance of the shape representation in the shape eigenspace below a fixed maximum (see chapter 2). This limits the eigenspace representation of the shape to be within a hyper-ellipsoid centred at the origin. In chapter 2 it was discussed how such a hyperellipsoid may contain both valid and invalid shape representations due to the linear nature of Principal Component Analysis (and the non-linear nature of the variation of real life objects). Various solutions have been proposed to this problem. These solutions fall into two categories;

1. Valid/Invalid area classification, for example Heap and Hogg's K-means clustering [46] or Cootes and Taylor's mixture models [20].

2. Non-linearising methods such as Sozou's use of polynomial combinations of eigen vectors [98] and neural networks [99] or Romdhani *et al.*'s use of Kernel PCA [89].

The application that has been developed takes the former approach because of its potential simplicity (and thus efficiency).

The stochastic sampling tracker uses a discrete shape model that is built by performing vector quantisation on the three eigenspaces (inter-animal, front legs and rear legs) of each of the three continuous shape models. This is described in detail in chapter 5 and is summarised by figure 7.5.

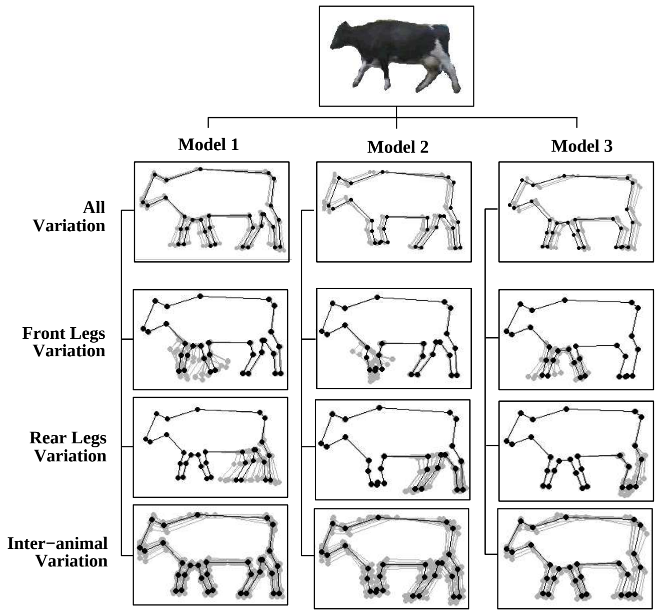


Figure 7.5: The Hierarchical Contour Model Used By The Trackers

The discretisation of the models allows multiple shape model tracking to be implemented in a simple manner and ensures the result of the search will be a valid shape. Two stochastic 'Predict and Sample' methods have been implemented and evaluated. The first is an iterative scheme with similarities to the Markov Chain Monte Carlo method [25] and the second is an extension of

Isard and Blake's 'Condensation' algorithm [57] which uses two sample sets. These methods are described fully in section 7.2.

All stochastic sampling methods require a 'fitness function' which defines how well a sample fits the data. A fitness function is defined based on edge intensity (using the 'moving edge' image described in section 7.1.1 rather than the raw edge image). This function is the mean of a set of linear functions of intensity ($F(intensity)$) with a maximum threshold (see figure 7.6) for each border pixel. The threshold is important as it reduces the influence of strong edges in the fitness function, thus increasing the influence of weaker edges (to a degree). This is a simple (and computationally inexpensive) approximation to human perception of edge images, which treats any pixel with a intensity above a certain value as an 'edge' pixel and uses contextual information to classify the remaining pixels probabilistically. The maximum intensity threshold of this function (I_t) is determined from a set of hand fitted examples as n standard deviations below the mean intensity around the perimeters of this training set. The gradient of the linear part of this line is simply calculated as the reciprocal of this value. Equation 7.4 and the graph in figure 7.6 illustrate the calculation of the fitness function.

$$Fitness = \frac{\sum_{x=0}^{N} F(intensity_x)}{N} \tag{7.4}$$

Where:

N = The number of perimeter points

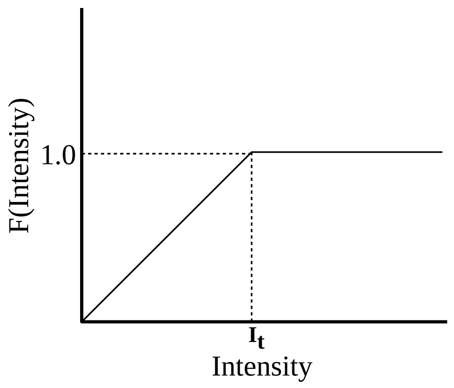


Figure 7.6: Fitness Function

This fitness function is simpler than the function used initially by Isard and Blake [57], which examines lines along normals to a spline representation for edges, and thus more computationally efficient to evaluate. Robustness of the fitness function is maintained by the blur stage in the image processing which ensures smooth edge gradients across the image. The extra computational cost of the blur is fixed whereas the saving from using a simpler fitness function is proportional to the number of samples used. Isard and Blake increase the computational efficiency of their fitness function by sampling only at particular intervals around the approximation perimeter. This can lead to erroneous results for noisy data where there is a break in the real object perimeter at a point of evaluation.

7.1.4 Local Search Tracker

The local search tracker takes the best result (or results) from the stochastic sampling tracker and performs a local search using the Active Shape Model (ASM) [19] search scheme with a limited range (see chapter 2 for a description of this algorithm) . The limited range of, and the tendency of the ASM scheme to find locally optimal solutions for noisy data restrict the final solution to be close to the starting point and thus to a be a valid (or near valid) shape. The range of the local search required can be estimated by comparing the differences between the training data shapes and the nearest vector quantisation prototype.

7.2 Stochastic Sampling Methods

In this section two alternative stochastic sampling methods are presented. It would be possible to use either method within the final system. An evaluation of the two methods, presented later (see section 7.4.1), clearly shows the superior performance of the latter method (Re-sampling Condensation) over the former method (Discrete Hypothesis Sampling). Both methods are presented to give a comparison, however only Re-sampling Condensation would be used in a final system.

7.2.1 Discrete Hypothesis Sampling: A Novel Iterative Scheme

Markov Chain Monte Carlo methods [36] are iterative sampling schemes to obtain approximations to unknown probability density functions (PDFs). These are described in detail in chapter 2, however the basic principle is simple and involves sampling and updating a probability density function (PDF) estimate iteratively. The location of new samples is based on the current estimated PDF and, under certain circumstances, the iteration will converge. If it can be determined that these methods will converge they can be used to fit shape models to data as shown by De Souza *et al.* [25], however the computational expense of these methods is high and the speeds achieved are orders of magnitude slower than current video frame rates.

A method has been developed based on the sample and update principal of Markov Chain Monte Carlo methods that can operate at approaching video frame rates by using discrete probabilities and a rolling average to update these probabilities using just the current generation of samples (rather than the complete set of samples taken). The search is restricted to positions and scales local to the position and scale from the previous frame (or the position tracker for the first frame) which further increases the efficiency of the scheme. The probabilities are initialised using a predictor based on the results from the previous frame (or initialised equal on the first frame). Various suitable predictor schemes are evaluated in chapter 6. Figure 7.7 gives an overview of the scheme.

The initial (prior) PDFs are estimated using a predictor from the final (posterior) PDFs from the previous time-step. This may be done on a winner-takes-all basis (i.e. select the best and predict from this) if the predictor scheme is computationally expensive, however it is more effective to

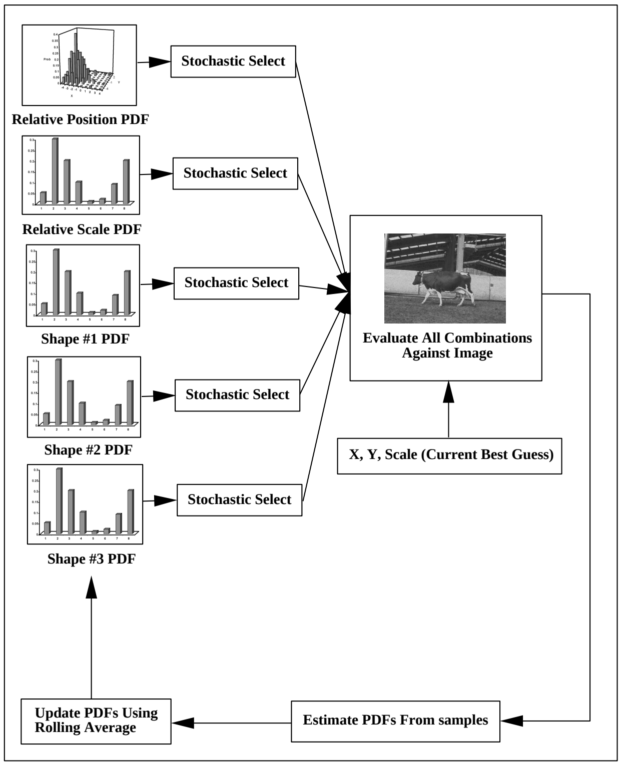


Figure 7.7: Iterative Scheme For Variable PDF Estimation

make a full probabilistic prediction based on the entire probability distribution as in equations 7.5 to 7.9.

$$P_{prior}(X,Y,T) = \sum_{\text{all x,y}} P_{post}(x,y,T-1) \times P_{pred}(x_T, y_T = X,Y | x_{T-1}, y_{T-1} = x,y) \quad (7.5)$$

$$P_{prior}(S,T) = \sum_{\text{all scales (s)}} P_{post}(s,T-1) \times P_{pred}(s_T = S | s_{T-1} = s) \quad (7.6)$$

$$P_{prior}(S1,T) = \sum_{\text{all s1}} P_{post}(s1,T-1) \times P_{pred}(s1_T = S1 | s1_{T-1} = s1) \quad (7.7)$$

$$P_{prior}(S2,T) = \sum_{\text{all s2}} P_{post}(s2,T-1) \times P_{pred}(s2_T = S2 | s2_{T-1} = s2) \quad (7.8)$$

$$P_{prior}(S3,T) = \sum_{\text{all s3}} P_{post}(s3,T-1) \times P_{pred}(s3_T = S3 | s3_{T-1} = s3) \quad (7.9)$$

Where:

$P_{prior}(Z,T)$	$=$	The prior probability distribution for Z to be used at time T
$P_{post}(Z,T-1)$	$=$	The final (posterior) probability distribution for Z produced by the object tracker at time T-1

$$P_{pred}(Z_T = Z1|Z_{T-1} = z1) \quad = \quad \text{The predicted output from the temporal predictor}$$

It is possible to use predictors based on more than one variable, for example to predict front leg position (shape #1) from both front and rear leg position (shape #1 and shape #2) posterior probabilities. This is achieved in a similar manner as given in equation 7.10.

$$P_{prior}(S1,T) = \sum_{\text{all s1,s2}} P_{post}(s1,T-1) \times P_{post}(s2,T-1) \times P_{pred}(s1_T = S1|s1_{T-1} = s1, s2_{T-1} = s2) \tag{7.10}$$

The predictors used in the evaluation (see section 7.4) are two stream cyclic Hidden Markov Models (see chapter 6) for the front and rear leg shape components, a first order Markov chain for the inter-animal characteristics and uniform probability predictors for relative position and scale. Absolute position and scale are updated by adding a constant value equal to the mean change from a training set, and the range of the relative position and scale distributions is selected based on the standard deviation of this training set (the range is fixed as +/-*n* standard deviations off-line, where n is typically around 3). These simple models for position and scale are sufficient as shape changes account for far more of the object's variation than position or scale changes from frame to frame.

The PDFs estimated from the samples are calculated as the sum of all the fitness for a particular value of a component divided by the sum of all sampled fitnesses and multiplied by the total probability of the selected component values in the original (prior estimate) PDF as given in equations 7.11 to 7.15.

$$P_{est}(X,Y) = \sum_{\text{all selected x,y}} P_{old}(x,y) \times \frac{\sum_{\text{x,y = X,Y}} f(x,y,scale,s1,s2,s3)}{\sum_{\text{all samples}} f(x,y,scale,s1,s2,s3)} \tag{7.11}$$

$$P_{est}(S) = \sum_{\text{all selected scales}} P_{old}(s) \times \frac{\sum_{\text{scale = S}} f(x,y,scale,s1,s2,s3)}{\sum_{\text{all samples}} f(x,y,scale,s1,s2,s3)} \tag{7.12}$$

$$P_{est}(S1) = \sum_{\text{all selected s1}} P_{old}(s1) \times \frac{\sum_{\text{s1 = S1}} f(x,y,scale,s1,s2,s3)}{\sum_{\text{all samples}} f(x,y,scale,s1,s2,s3)} \tag{7.13}$$

$$P_{est}(S2) = \sum_{\text{all selected s2}} P_{old}(s2) \times \frac{\sum_{\text{s2 = S2}} f(x,y,scale,s1,s2,s3)}{\sum_{\text{all samples}} f(x,y,scale,s1,s2,s3)} \tag{7.14}$$

$$P_{est}(S3) = \sum_{\text{all selected s3}} P_{old}(s3) \times \frac{\sum_{\text{s3 = S3}} f(x,y,scale,s1,s2,s3)}{\sum_{\text{all samples}} f(x,y,scale,s1,s2,s3)} \tag{7.15}$$

The rolling average is then performed by taking a weighted sum of the prior estimate PDF ($P_{old}()$) and the estimated PDF ($P_{est}()$) for selected component values as in equations 7.16 to 7.20. Non selected component values remain unchanged.

$$P_{new}(x,y) = W \times P_{est}(x,y) + (1-W) \times P_{old}(x,y) \tag{7.16}$$

$$P_{new}(scale) = W \times P_{est}(scale) + (1-W) \times P_{old}(scale) \tag{7.17}$$

$$P_{new}(s1) = W \times P_{est}(s1) + (1-W) \times P_{old}(s1) \tag{7.18}$$

$$P_{new}(s2) = W \times P_{est}(s2) + (1-W) \times P_{old}(s2) \tag{7.19}$$

$$P_{new}(s3) = W \times P_{est}(s3) + (1-W) \times P_{old}(s3) \tag{7.20}$$

The subject of Markov Chain Monte Carlo convergence is a complex one which has been dealt with in the mathematical literature (for example [61] or [12]), however this is beyond the scope of this thesis. It can be gathered, however, that it cannot be guaranteed that the algorithm will converge to a set of stable probability distributions within acceptable operating time on current hardware, however this does not mean that this is not a useful sampling method. Initialisation by predictor means that the probability distributions will have some relation to the data after only a few iterations and a track can be kept of the best sample(s) over all iterations.

7.2.2 Re-sampling Condensation

The main drawback with the standard Condensation algorithm [57] is that sample location is determined purely by prediction from past observations. Isard and Blake explain [58] that this results in sample locations clustering round regions of predicted high probability with few samples representing areas of lower predicted probability. This is a problem in applications where there are multiple possible outcomes of differing probabilities. Iterative sampling algorithms such as the method described in the previous section (or alternatively Genetic Algorithm based methods) do not exhibit this problem as they re-sample the solution based on previous results. These algorithms are however less suitable for 'real time' applications due to their high computational cost.

A method has been developed that combines the efficiency of the Condensation algorithm with the hypothesis selection ability of iterative methods. 'Re-sampling Condensation' is a two stage algorithm with no computational overhead over the standard Condensation algorithm that, for the application under consideration, gives more robust tracking results as image information is included in sample location. Re-sampling Condensation splits the samples into two groups. Sampling of the first group is performed using the standard select, predict and evaluate Condensation scheme described in chapter 2. Multiple samples are then selected stochastically from this initial sample set (based on a fitness function) and combined to give a new sample location. This is illustrated in figure 7.8.

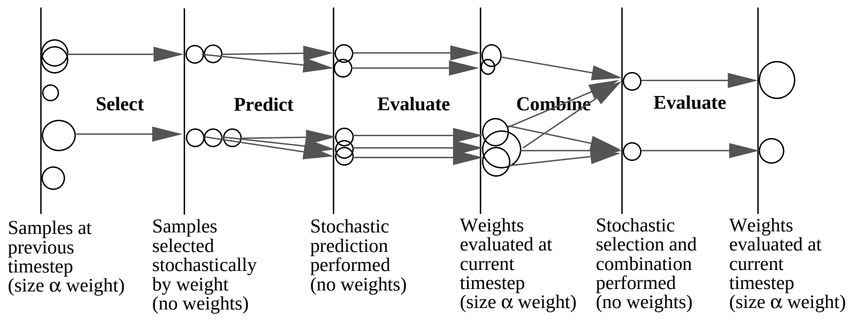

Select
Predict
Evaluate
Combine
Evaluate
Samples at previous timestep (size α weight)
Samples selected stochastically by weight (no weights)
Stochastic prediction performed (no weights)
Weights evaluated at current timestep (size α weight)
Stochastic selection and combination performed (no weights)
Weights evaluated at current timestep (size α weight)

Figure 7.8: Re-sampling Condensation Flow Diagram

Re-sampling is analogous to a single iteration of a genetic algorithm. In the livestock example three of the initial samples are selected and inter-animal, front legs and rear-legs characteristics taken from different samples to make up the new sample. Position and scale are determined stochastically from the mean and standard deviation of these parameters in the initial samples selected by assuming a normal distribution and selecting from this distribution (see [84] for method).

It should be noted that Re-sampling Condensation relies on the variation within the spatial model being separated into pseudo-independent components (position, scale, inter-animal, front legs and rear-legs characteristics in the livestock example). These are analogous to the ‘genes’ in a genetic algorithm. Experimental results suggest that the more components that a spatial model may be separated into, the better the final solution. This is intuitive as, considering a system with 2 independent parameters with N and M possible states for each parameter, searching using a model with a single parameter results in a search space of size N*M whereas searching using a model with two parameters results in two search spaces of size N and M respectively. In most tracking examples parameters are not truly independent, however component separation can still yield improved search efficiency. This is demonstrated in the livestock tracking example by the fact that a tracker based on a single shape description space cannot maintain ‘lock’ on an animal, whereas a tracker using spatial and temporal models based on three shape description spaces maintains a reasonable ‘lock’ on an animal.

The temporal prediction methods used in the Discrete Hypothesis Sampling scheme are also used in the Re-sampling Condensation framework with the complex front and rear leg movements being modeled by a two stream CHMM. Figure 7.9 shows the information contained in each sample which is propagated over time.

Initial samples are selected at random by selecting a hidden state, position andscale stochastically from uniform probability distributions. Position and scale are selected local to the initial guess from the position tracker by assuming these parameters are normally distributed with a predetermined standard deviation based on statistics derived from hand fitted data. The shape model and front and rear visible states are selected stochastically from the front and rear legs probability distributions given by the hidden state. From the model selected, inter-animal state is selected stochastically from a uniform probability distribution. Once the samples are initialised they are propagated through the image sequence using the re-sampling condensation framework as illustrated in figure 7.10.

Information	**Description**
Hidden State	Hidden State of CHMM
Model	The Shape Model Used (Derived stochastically from Hidden State)
Visible State (Front)	Vector quantisation prototype for Front Legs Space (Derived stochastically from Hidden State)
Visible State (Rear)	Vector quantisation prototype for Rear Legs Space (Derived stochastically from Hidden State)
Inter-animal State	Vector quantisation prototype for Inter-animal Space
Position	X and Y Position in image co-ordinates
Scale	Scale relative to average dimensions of training data

Figure 7.9: Sample Composition

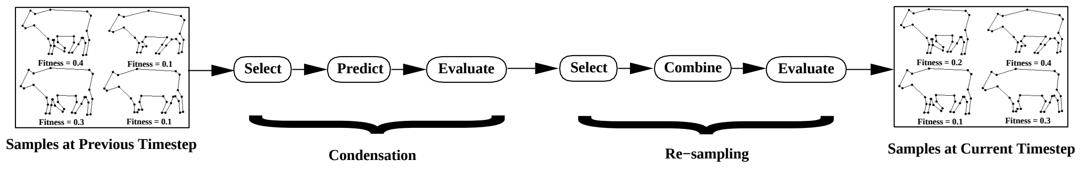


Figure 7.10: The ‘Re-sampling Condensation’ Framework

7.3 Using Multiple Temporal Models in a combined Tracker-Classifier Scheme

Chapter 2 described how the Condensation algorithm has wider application than simple object tracking. In particular Black and Jepson [6] use Condensation to track and classify the trajectories of a coloured white-board marker. In their scheme multiple trajectory models are used and the ‘gesture’ classified as one of six actions to be performed.

In the scheme implemented here the combined tracker and classifier paradigm of Black and Jepson is used to model object shape change over time in a ‘Re-sampling Condensation’ framework (see previous section). As an example ‘normal’ and ‘lame’ behaviours are modelled by separate Cyclic Hidden Markov Models (CHMMs, see chapter 6) within the ‘Re-sampling Condensation’ framework. Samples are allocated to each model initially in even proportion, and the CHMM is propagated through time with each sample. Over time the CHMM that best fits the observed object 'behaviour' will dominate (i.e. more samples will be associated with that model). A simple classification can then be performed by comparing the number of samples associated with each model. A more complex classification method involves summing the posterior probabilities (relative fitnesses) for samples associated with each CHMM.

7.4 Results and Evaluation

7.4.1 Evaluation of Tracking Results

'Discrete Hypothesis Sampling' and 'Re-sampling Condensation' were evaluated using a test set of 10 sequences of healthy cows walking from right to left in a farmyard setting as shown in figure 7.11. These sequences are of approximately 5 seconds in length at 25fps and contain at least three complete cycles of the MSCHMM. Ground truth about these sequences was obtained by hand fitting landmark points. Other sequences were also used in the construction of the spatial (shape) model.

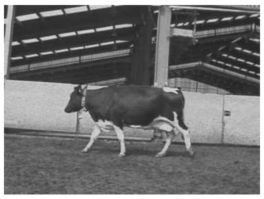


Figure 7.11: Typical Livestock Tracking Scenario

Sets of leave-one-out tests were performed by building MSCHMMs from 9 of these sequences and using the remaining sequence as a test sequence. The training data was obtained by projecting the hand fitted 'ground truth' points into the model parameter spaces and selecting the nearest vector quantisation prototype (a crude but reasonably effective method).

7.4.1.1 Evaluation of Discrete Hypothesis Sampling

‘Discrete Hypothesis Sampling’ is a complex method with many variables (range of local search, quantisation level for shape, scale and position, number of iterations, number of shapes, scales and positions selected per iteration, rolling average factors). To do a complete evaluation of all combinations of parameters is not feasible within the scope of this thesis due to the time involved (estimated to be several years at current computer speeds), however the most important parameter (the number of iterations) was evaluated with other parameters being set to near optimal values determined from ad-hoc tests and statistics from hand fitted data. The results for different numbers of iterations for the operating conditions given in figure 7.12 are given in figures 7.13 to 7.15. Errors are given with respect to a hand specified ‘best spatial model’ (BSM) and set of landmark points. The choice of BSM is subjective for a significant minority of frames (around 20%) where the animal shape is close to the boundry between spatial model representations. This partially accounts for the apparently high percentage of ‘wrong models’. Average and standard deviation of errors are given for frames classified with the ‘correct’ spatial model (in terms of the hand specified BSM) only.

Variable	**Value**
Range of Position Search in X	+/- 6 Pixels
Range of Position Search in Y	+/- 3 Pixels
Position Search Quantisation	1 Pixel
Range of Scale Search	+/- 1.8%
Scale Search Quantisation	0.6%
Shape Quantisation (Between Animal)	36 Shapes †
Shape Quantisation (Front Legs)	34 Shapes †
Shape Quantisation (Rear Legs)	36 Shapes †
Rolling Average Factors	0.5
Shapes, Scales Positions per Iteration	Determined from Number of Iterations and Total Samples

† Shape quantisation is determined by the vector quantisation process (see chapter 5)

Figure 7.12: Operating Variable Values for Discrete Hypothesis Sampling Evaluation

No. of Iterations	Average Error (Pixels)	Error SD (Pixels)	% Wrong Models
1	10.86	11.05	54.2
2	7.79	6.97	44.7
3	6.65	6.55	41.1
4	6.46	5.59	41.8
5	6.01	4.85	39.3
6	5.99	5.06	41.5
7	6.22	5.27	40.9
8	6.60	5.75	40.3
20	5.71	4.59	43.1
50	6.92	6.30	42.6

Figure 7.13: Results for Discrete Hypothesis Sampling Evaluation

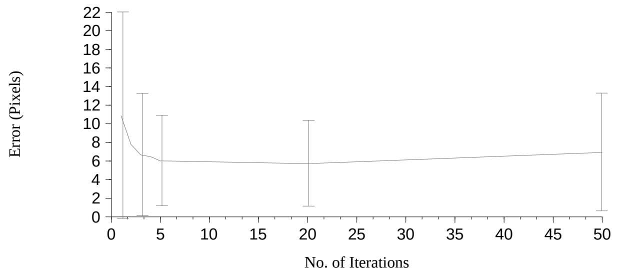


Figure 7.14: Average Point Error (+/- 1 SD) for Discrete Hypothesis Sampling with 1000 Total Samples w.r.t Hand Fitted Data

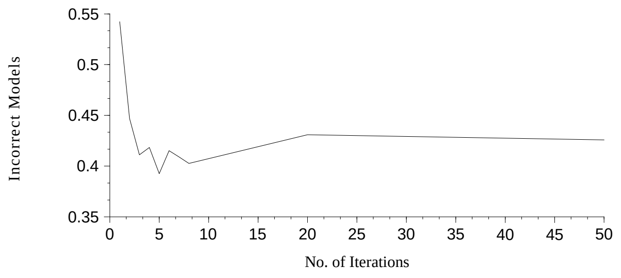


Figure 7.15: Proportion of ‘Wrong’ Spatial Models w.r.t Hand Fitted Data

There are two important points to be drawn from the results presented in figures 7.13 to 7.15. The first phenomenon to note is the sharp drop in average error, error standard deviation and percentage of ‘Wrong’ spatial models between 1 and 5 iterations. The second phenomenon to note is the gradual increase in average error (and standard deviation) after around 20 iterations. These phenomena are caused by the trade off between the number of iterations used and the number of samples per iteration. As the number of iterations increases the number of updates to the PDFs increases and, on average, samples are based on more image information. This principal has more effect when the total number of iterations is small. As the number of iterations increases further the number of samples per iteration drops. This means each iteration has a smaller effect on the PDFs in terms of the number of discrete probabilities affected. In extreme cases the sample size becomes so small that it ceases to be a statistically accurate sample of the data and the assumption that model components are independent breaks down due to interdependencies of components which can no longer be robustly averaged out. Increasing the number of iterations also increases the computational cost of the method, although the PDF update cost is small in comparison to sample evaluation. For this reason it would not seem sensible to use the method with large numbers of iterations. A sensible operating point for the scenario presented within this thesis is around 5 iterations for 1000 total samples.

Overall the performance of this algorithm is variable. Although probability distributions for shape are propagated through time, only a single hypothesis for position and scale is propagated. This can lead to errors in position and scale accumulating over time and accurate tracking being lost. This led to the development of the Re-sampling Condensation algorithm in which multiple hypotheses for scale, position and shape are propagated. The Re-sampling Condensation algorithm visibly outperforms the Discrete Hypothesis Sampling method (as the quantative results in the following section show) meaning further evaluation of Discrete Hypothesis Sampling is relatively pointless. The formulation of the Discrete Hypothesis Sampling is not completely without merit however, as the evaluation shows that re-sampling from a data set can yield improved solutions even if the sampling algorithm has not achieved convergence.

7.4.1.2 Evaluation of Re-sampling Condensation

To evaluate ‘Re-sampling Condensation’ tracking was performed at various levels of re-sampling (i.e. number of re-samples vs. number of condensation samples) from 0% (Standard Condensation) to 95%. The tracking results for the sample of maximum fitness were compared to the hand fitted 'ground truth', and statistics gathered. This evaluation was identical to the evaluation performed for the Discrete Hypothesis Sampling method (see section 7.4.1.1) and, as such, the results are comparable. Results at different numbers (250, 500 and 1000) of total samples are given in figures 7.16 to 7.19.

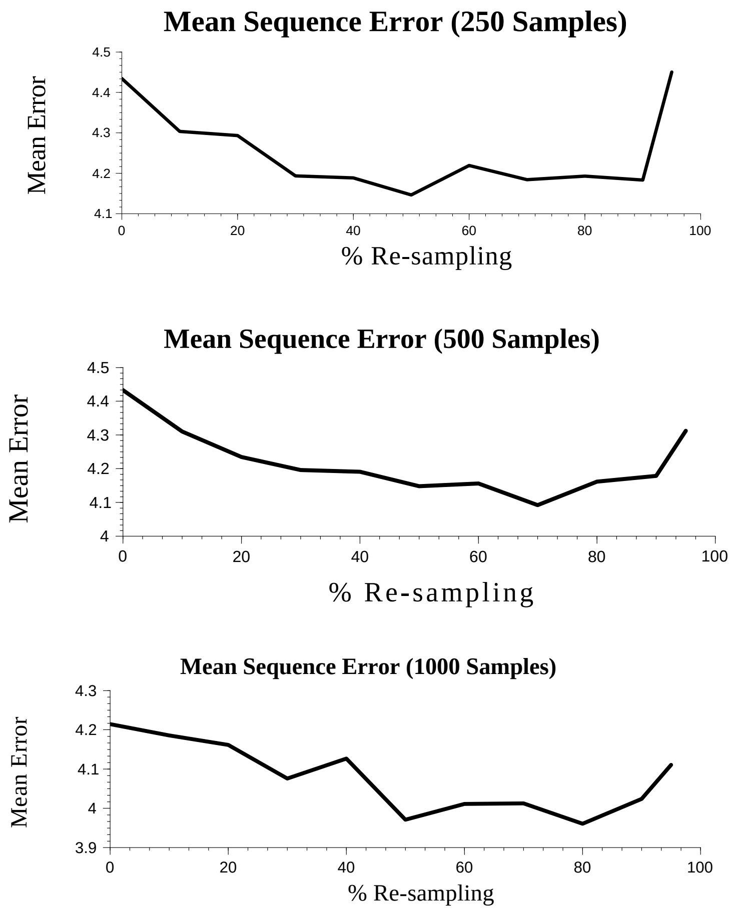


Figure 7.16: Mean Error (pixels) of Tracker w.r.t Hand Fitted Data at Different Levels of Re-sampling

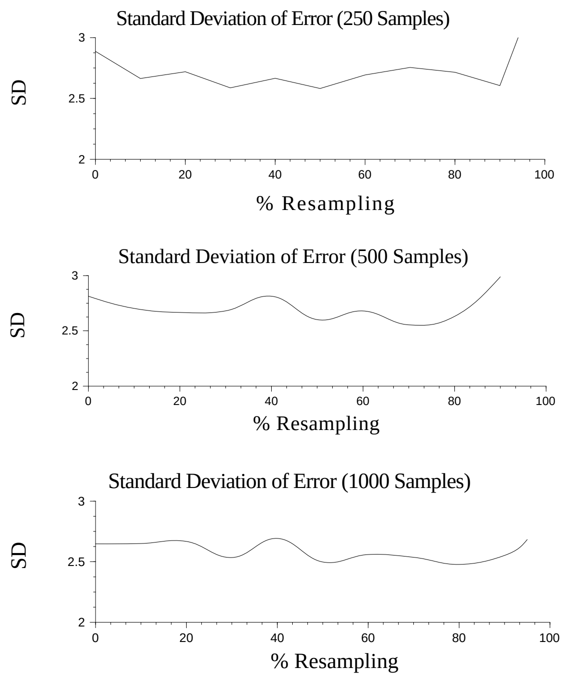


Figure 7.17: Standard Deviation (pixels) of Tracker w.r.t Hand Fitted Data at Different Levels of Re-sampling

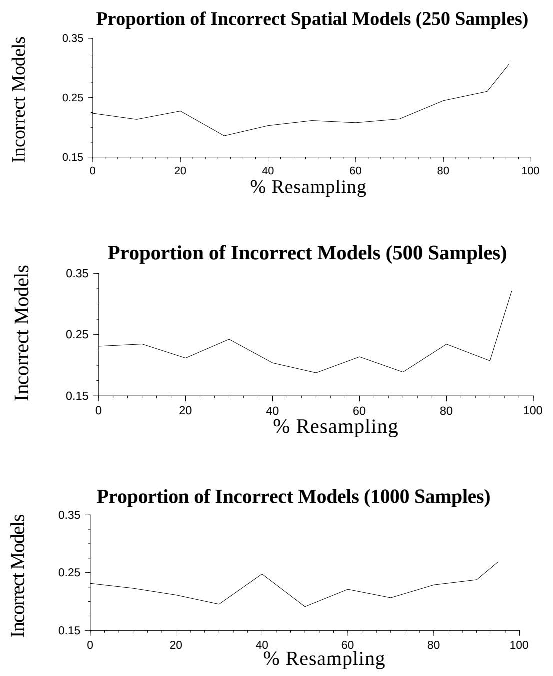


Figure 7.18: Proportion of 'Wrong' Spatial Models w.r.t Hand Fitted Data at Different Levels of Re-sampling

No. of Samples	% Re-sampling	Average Error (Pixels)	Error SD (Pixels)	% Wrong Models
250	0	4.43	2.89	22.4
	10	4.30	2.66	21.3
	20	4.29	2.72	22.7
	30	4.19	4.15	18.6
	40	4.19	2.66	20.3
	50	4.15	2.58	21.1
	60	4.21	2.69	20.8
	70	4.18	2.75	21.4
	80	4.19	2.71	24.5
	90	4.18	2.60	26.0
	95	4.45	3.09	30.6
500	0	4.43	2.81	23.1
	10	4.31	2.70	23.4
	20	4.23	2.67	21.1
	30	4.20	2.68	24.2
	40	4.19	2.81	20.4
	50	4.15	2.60	21.4
	60	4.16	2.68	21.4
	70	4.09	2.55	18.9
	80	4.16	2.63	23.4
	90	4.18	2.98	20.7
	95	4.31	2.87	32.1
1000	0	4.21	2.65	23.1
	10	4.19	2.65	22.3
	20	4.16	2.67	21.1
	30	4.07	2.53	19.5
	40	4.13	2.69	24.7
	50	3.97	2.48	19.1
	60	4.01	2.56	22.1
	70	4.01	2.54	20.1
	80	3.96	2.47	22.9
	90	4.02	2.55	23.8
	95	4.11	2.68	26.9

Results are given w.r.t. hand fitted data

Figure 7.19: Results for Re-sampling Condensation Evaluation

The results in figure 7.16 show lower average error rates for Re-sampling Condensation at intermediate levels of re-sampling, although there is a drop off in performance at high levels of re-sampling. It should be noted that the tracking error is superimposed on the quantisation error inherent in the quantised model. The average error for the quantised ground truth data used for training is 4.08 pixels using the crude quantisation method described previously. It is encouraging that the tracker can, at optimal re-sampling, improve on this value. Error standard deviation shows similar trends to mean error although this is always higher than for the quantised training data. Examining individual sequences around the optimum operating points indicated by the graphs in figure 7.16, the Re-sampling Condensation tracker performs better than Condensation alone (0% re-sampling) in 80-100% of cases. Even at very low ($<$20%) and very high ($>$90%) levels of re-sampling the Re-sampling Condensation tracker can perform better in more than 50% of cases.

The error standard deviation results (figure 7.17) and ‘Wrong’ models results (figure 7.18) support the results from the average error, however they are less conclusive. Performance definitely drops off at high levels of re-sampling, however it is difficult to say if there is any improvement at intermediate levels of re-sampling from these measures. There is no evidence of a drop off in performance at intermediate levels of re-sampling and as such conclusions from the average error results may be believed.

These results are as expected as the inclusion of current image information in sample location improves tracking performance. The fall off in performance at high levels of re-sampling is due to a ‘gene deficiency’ in the initial Condensation set of samples when few samples are allocated to the Condensation stage. It may be the case that this initial ‘population’ of samples contains no members with the ‘most correct’ individual characteristics and as such the most correct (highest fitness) solution cannot be found by the re-sampling stage. It may also be the case that the initial population of samples contains a high proportion of members with the most correct individual characteristics, in which case tracking performance will be particularly good. It is as such undesirable to operate at high levels of re-sampling as tracking robustness is lower than at intermediate levels of re-sampling. It should also be noted that the optimum level of re-sampling increases with the total number of samples due to this phenomenon.

7.4.2 Comparison of Different Tracking Methods

Three tracking methods (Discrete Hypothesis Sampling, Re-sampling Condensation and standard Condensation) have been evaluated in this chapter. In table 7.20 a comparison of the three methods at ‘optimal’ operating points is presented (where ‘optimal’ is defined as the best operating point observed within the evaluation in terms of average error, standard deviation of error and proportion of incorrect models).

Method	**Average Error (Pixels)**	**Error SD (Pixels)**	**% Wrong Models**
Discrete Hypothesis Sampling	6.01	4.85	39.3
Re-sampling Condensation	3.97	2.50	19.1
Standard Condensation	4.21	2.65	23.1

Figure 7.20: Comparison of Tracking Methods for 1000 Total Samples at ‘Optimal’ Operating Points

It can be seen that the two Condensation methods outperform the Discrete Hypothesis Sampling method and that the Re-sampling Condensation is the best method (for this data set). There is a significant proportion of ‘Wrong Models’ (around 20%) even at near optimal operating conditions. These errors are due to the subjective nature of model selection at the boundary between spatial models. The tracker and the human observer may disagree over the model to be used, however this is not a great problem as the prototypes selected will always have similar semantic meaning within the temporal model used. Typical results for the best method (Re-sampling Condensation) are given in figure 7.21.

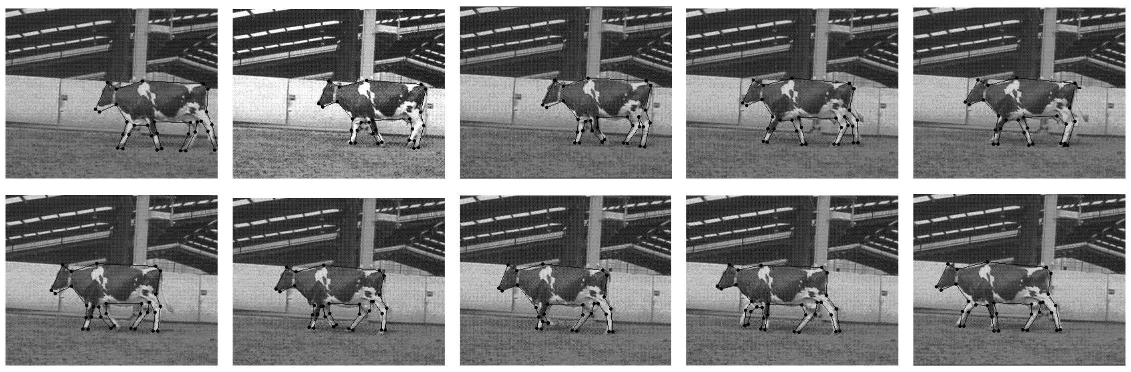


Figure 7.21: Typical Cow Tracking Results Using the Re-sampling Condensation Method

7.4.3 Evaluation of Lameness Classification

The evaluation of the scheme in the detection of lameness was performed on sequences of humans walking as insufficient amounts of lame cow data were available. The 11 (healthy) subjects were asked to perform a choreographed lame walking motion in addition to their regular walking motion as illustrated in figures 7.22 and 7.23 respectively.

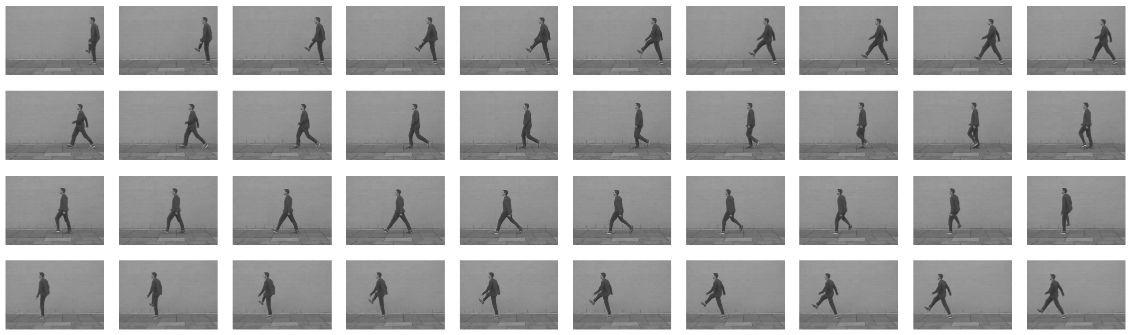


Figure 7.22: A 'typical' Lame Walk Sequence

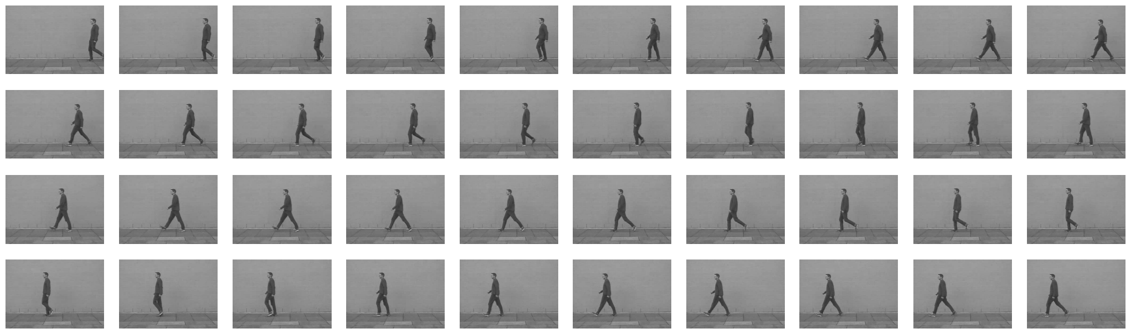


Figure 7.23: A 'typical' Healthy Walk Sequence

Two sets of each were taken for each person and a set of leave one out tests performed. The spatial (shape) model used was a B-spline based model (*a la* Baumberg and Hogg [3]) rather than the straight line approximation as used for the cows as people are not well approximated by straight lines. The sum of the posterior probability (normalised fitness) for each CHMM was recorded over time for each sequence. Taking a mean of these probabilities over time (excluding the first cycle to allow for initialisation of the tracker) gives an indication of the relative probability that each sequence is either 'Healthy' or 'Lame'.

Using this method all 44 sequences were correctly classified and in all but two cases the results were very clear as can be seen in figures 7.24 and 7.25.

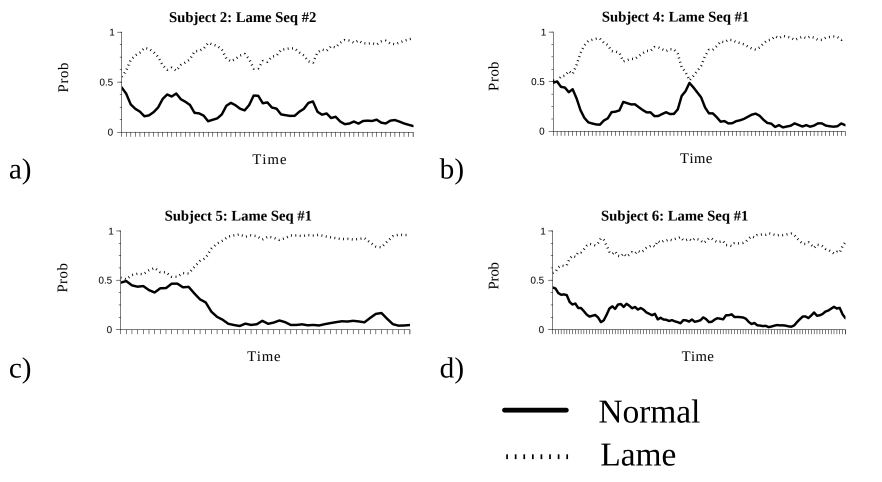


Figure 7.24: Model Probability vs. Time for 'typical' Lame Walk Sequences

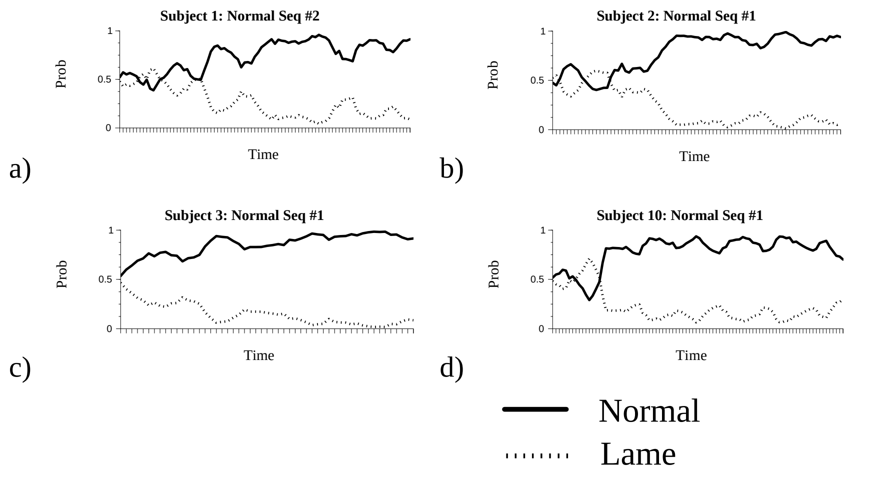


Figure 7.25: Model Probability vs. Time for 'typical' Normal Walk Sequences

Two sequences produced results which, although they were classified correctly, were not as clear (see figure 7.26). On investigation this was found to be related to the 'lame' portion [1] of the 'lame' CHMM being similar to the healthy walking motion in the rotationally normalised shape representation. The spatial model used to model the cows is not normalised by rotation and thus would not be subject to this problem. In the non-typical lame sequence a significant number of samples were being propagated half a cycle out of phase with the actual cycle resulting in poor

[1]N.B. The 'lame' walking cycle consists of approximately half a cycle that is identical to a 'healthy' walk and half a cycle that differs. In the 'healthy' walking cycle the two half cycles are identical from the view used.

tracking performance. Similarly for the healthy sequence samples relating to the 'lame' CHMM were being propagated in both phases resulting in reasonable tracking by the lame model compared to the healthy model. These problems would be eliminated by using a non-rotationally normalised model or including relative rotation as an additional stream in the CHMM as out-of-phase samples would not be propagated.

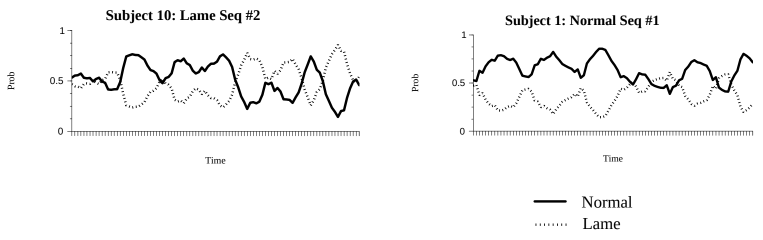


Figure 7.26: Model Probability vs. Time for the two 'non-typical' Sequences

7.5 Discussion and Conclusions

Two novel object tracking methods 'Discrete Hypothesis Sampling' and 'Re-sampling Condensation' have been presented in this chapter. These methods are both based on fitting a spatial model of an object to an image sequence by sampling the space of possible solutions in an 'intelligent' manner based on a temporal predictor. Image information is included in sample location in both schemes by 'Re-sampling' the solution space based on the results from initial samples. Discrete Hypothesis Sampling represents prior information from the temporal predictors and previous samples explicitly as a set of discrete probability distributions for shape ($\times 3$), scale and position. These distributions are completely independent within this method which is the main reason for the superior performance of the Re-sampling Condensation algorithm. Re-sampling Condensation represents prior information implicitly as a set of samples (or 'particles') as in the standard Condensation algorithm [57]. This particle representation is referred to as a probability density function within the literature (e.g. [57]) as it can be used to describe a complex probability distribution within a high dimensional space. The strength of the particle representation is that it approximates to a joint probability distributions for shape, scale and position, rather than a set of individual probability distributions. In the scenario presented within this thesis these properties are related (to a certain extent). This, along with the single hypothesis propagation of scale and position in the Discrete Hypothesis Sampling method, result in the superior performance of the Re-sampling Condensation algorithm.

There is an analogy to be made between Re-sampling Condensation and the Importance Sampling used by Isard and Blake (in the ICondensation algorithm [58]) in that information from the current time-step is included in the prior probability for the second set of samples. The important distinctions between these methods are that Re-sampling Condensation takes this information directly from the standard sampling method without the need for a secondary (low resolution) sampling method and that Re-sampling Condensation works by combining prior samples in a s-tochastic manner. The combination of prior samples can lead to the probability of samples being located in novel locations in the complete model configuration space being high. Within ICondensation/Condensation or standard Markov Chain Monte Carlo implementations (e.g. [25]) the probability of a sample being located in a novel position (i.e. a position not represented by the prior probability distribution) is always small. This leads to the inclusion of a small proportion of completely random samples in most applications of these algorithms. Re-sampling Condensation produces improved performance when these novel sample locations in the complete model configuation space have high probability (in terms of current observations). This is generally only the case if the components combined are sufficiently independent. This is the case in the model described in chapters 3-5.

No evaluation of local search schemes (such as the Active Shape Model [19]) has been presented in this thesis primarily because there has been no novel work in this area. Such schemes may be a useful extension to the methods presented within this chapter, however the extra computational cost of the Active Shape Model, in relation to ‘real time’ tracking applications, does not justify the increase in tracking accuracy (which is very marginal in the case of the livestock example). The extra tracking accuracy is not in fact required within the people and livestock examples presented within this thesis as all the semantic information required is present within the quantised models used.

In section 7.3 it was described how multiple temporal models may be used within a Condensation or Re-sampling Condensation framework as a combined tracker and motion analyser. Although this has not actually been applied to lame cows (only lame people), it has scope for use in a multitude of temporal analysis scenarios. The cyclic HMM used is appropriate for modeling lame and healthy walking behaviours, however more complex models may be required to model more complex behaviours such as those produced by oestrus (when the animal is ‘on heat’). There has been work on automatic architecture determination for HMMs such as that by Kettnaker and Brand [67] which would be essential in modelling such behaviours. Data collection for training and evaluating such models would be a very difficult task and is beyond the scope of this thesis.

In conclusion the novel ‘Re-sampling Condensation’ extension to the Condensation algorithm presented in this chapter can outperform both the original Condensation algorithm and the iterative sampling method (‘Discrete Hypothesis Sampling’) presented in situations where object variation can be separated into separate pseudo-independent components. This method has no computational overhead over original Condensation, and can in fact be faster as fewer temporal predictions are required. Multiple temporal models may be included in the framework in a similar manner to the original Condensation algorithm ([59]) and used for temporal classification ([6, 88, 89]).

Chapter 8

Identification of Individual Animals from their Markings

For a long time farmers, and others involved in farming, have identified individual animals within a herd of cattle by their markings (for breeds where this is appropriate). A high proportion of dairy cows within the United Kingdom are of a single breed (Holstein Friesians) which exhibit characteristic markings. It is believed by many that no two individuals have exactly the same markings. This has even been used as part of an animal's identification on the Ministry of Agriculture's (MAFF) paperwork.

The automatic identification of individuals is an interesting pattern recognition problem as it is very solvable and the domain is restricted. Dairy farm sizes in the UK range from around 60 animals to nearly 1000 animals. These are not large numbers by pattern recognition standards and the problem is further simplified by farms being split up into groups containing typically less than 50 animals. A further simplification is the size of the milking parlour. Each milking parlour has only a finite number of milking machines (6 or 8 is typical, however larger numbers are possible) with animals being milked in small groups. Many modern farms contain electronic milking machines which identify individuals by an electronic tag with a very short range of transmission. This potentially reduces the pattern recognition problem to a one in eight identification. The method presented in this chapter is an example application which is well able to solve problems of this magnitude and larger.

8.1 Building Appearance Classifiers for Livestock

The input to the classification system is an image (or image sequence) containing the individual to be identified, background and possibly other animals and people as can be seen in figure 8.1a. The separation of the individual from the background has been dealt with in previous chapters by fitting a piecewise linear contour model to the image as seen in figure 8.1b.

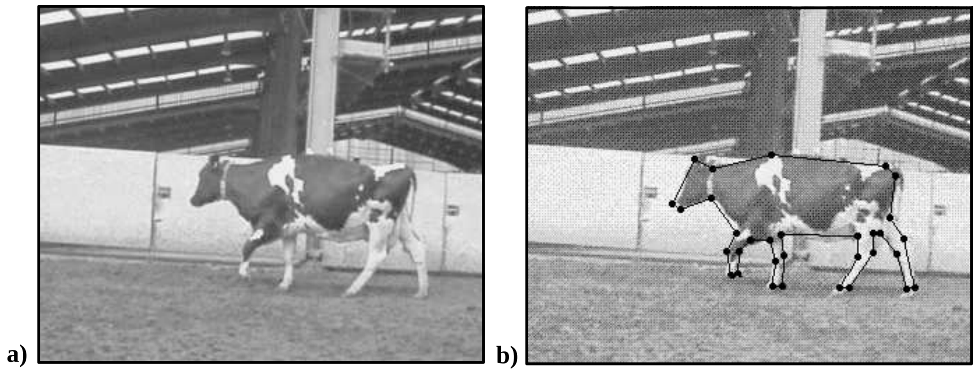


Figure 8.1: Fitting a contour model to an image using an object tracker

It is sensible to use this contour information to extract a description vector from the image. There are several different ways that this could be performed including warping the image to a normalised grid or dividing the animal up into sections and building histograms, however a simpler method is available that has proved robust under evaluation. This method involves taking a set of values along the lines between the centroid of the landmark points and the landmark points. This is illustrated in figure 8.2.

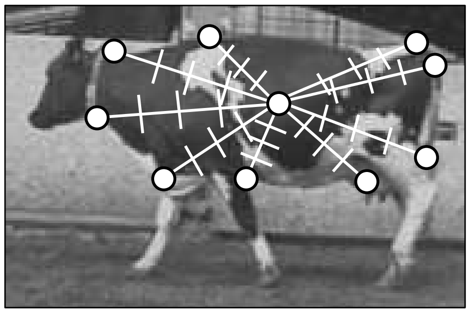


Figure 8.2: Identification Scheme Topology

Only landmark points on the torso of the animal are used as points on the legs and head may be occluded or the lines may cross non-animal areas of the image. In the system implemented this is a hand-selected subset of the landmark points presented by the tracker. An additional advantage of this subset of points is that the centroid of the landmarks is more stable than if leg and head landmarks were used in the calculation of the centroid. This stability is demonstrated in figure 8.3.

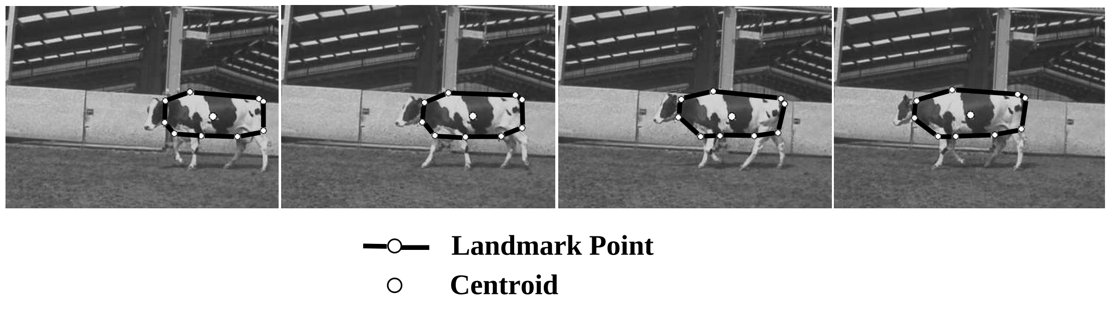


Figure 8.3: Location of the Landmark Centroid Over an Image Sequence

Each line is divided up into a number of sections and the distribution of intensity values along each section recorded. Different numbers of sections were tried and evaluated (see section 8.2). Descriptions of the nature suggested are collected for a training set of individuals (several per individual) and statistics gathered for each individual. It was found that building a ‘description vector’ consisting only of the mean intensity of each section of each line is sufficient to describe individuals in the evaluation presented later. More complex statistics, such as variances or histograms, could be used if this proved not to be the case for larger data sets.

8.2 Evaluation

An evaluation of the method described previously was carried out using twenty five animals, with ten examples of each animal. Twenty five was chosen as this is typical of the size of group that dairy animals are kept in, and because this is a practically obtainable data set. This data set was divided equally into a training set and a test set with five examples of each animal in each set. The average description vector values were calculated for each animal in the training set as described previously. These were used as classification prototypes by taking the closest animal average (in terms of Euclidean distance) to each description vector in the test set. The results of this evaluation are presented in figures 8.4 to 8.6. In addition to classification results details of the ratios between the distances between the data description vector and the closest and second closest prototypes are presented to give an idea of the robustness of the classification.

		Ratio of Closest to Second Closest	
No. of Sections / Line	Correct Classifi cations (/125)	Average	Standard Deviation
1	113 (90.4 %)	2.44	1.47
2	122 (97.6 %)	2.73	1.75
3	119 (95.2 %)	2.73	1.44
4	122 (97.6 %)	2.65	1.35
5	120 (96.0 %)	2.47	1.21
6	119 (95.2 %)	2.45	1.07
7	124 (99.2 %)	2.34	0.93
8	123 (98.4 %)	2.29	0.91
9	123 (98.4 %)	2.24	0.90
10	125 (100 %)	2.21	0.86
11	122 (97.6 %)	2.20	0.87
12	124 (99.2 %)	2.17	0.71
13	124 (99.2 %)	2.15	0.81
14	124 (99.2 %)	2.11	0.80
15	124 (99.2 %)	2.11	0.90
16	123 (98.4 %)	2.06	0.77
17	123 (98.4 %)	2.05	0.74
18	123 (98.4 %)	2.04	0.74
19	124 (99.2 %)	2.02	0.71
20	124 (99.2 %)	2.01	0.71

N.B. At greater than 20 sections the pixel resolution is reached for some of the shorter lines resulting in less than one pixel per section.

Figure 8.4: Animal Identification Results

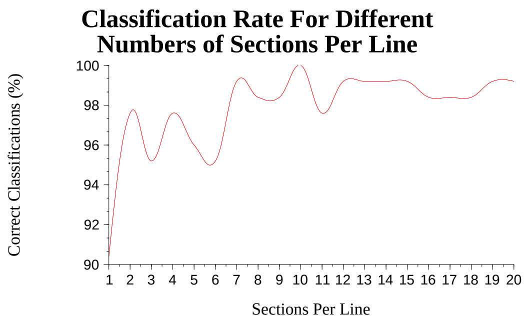


Figure 8.5: Identification Scheme Classification results

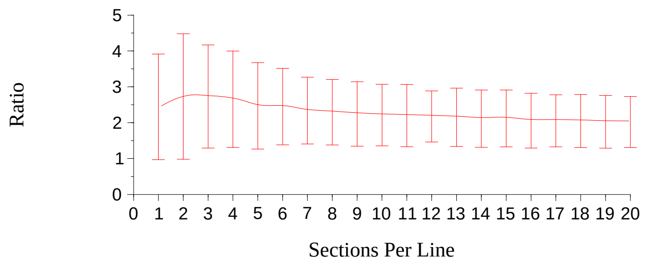


Figure 8.6: Ratios of Distances between closest and second closest Averages (+/- 1 SD)

8.3 Discussion

The results presented in the previous section are very good considering the simplicity of the method, illustrating the robustness of the method. This is even the case for very low numbers of sections per line, indicating that high resolution in not necessary for accurate classification. The decrease in the average ratio between the closest and the second closest animal prototypes with number of sections per line when using over three sections suggests higher resolution may have a detrimental effect on classification accuracy. The explanation for this is that alignment errors caused by poor landmark point placement and changes in animal orientation have more effect when there are more sections per line. This is because small features may fall into different sections in different examples. The choice of the number of sections per line is a trade off between representation resolution and the occurrence of such errors. For images of the resolution used within this thesis (half frame - 352x272 pixels), such as the example in figure 8.1, a choice of around 8 sections per line is sensible. This allows high enough resolution to model most marking exhibited while minimising the risk of alignment errors. It should be noted that it is better to choose a lower number of sections as the computational cost of the classification increases with the number of sections per line, which is an issue in real time systems. Figure 8.7 illustrates the animals incorrectly classified at this operating point.

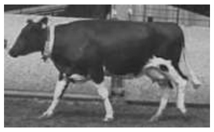
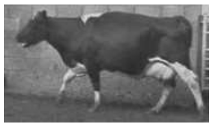
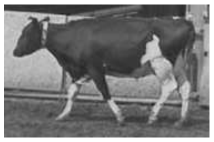
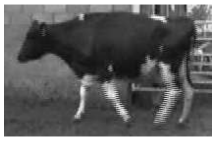
Incorrectly Classifi ed Animal	Example of Animal Classifi ed As	Ratio of Distances (Incorrect/Correct)
		1.09
		1.04

Figure 8.7: Incorrect Animal Identification Results

Two things should be noted from figure 8.7; Firstly the incorrect classifications are for individuals with very similar, but not identical, markings. Secondly the ratio between the distances to the means for the correct and incorrect classifications is near unity. These results show the limitations of the system as it stands. It is plain to see that introducing more lines into the system would increase the robustness of classification, however this also increases the computational cost of evaluation. This could be achieved by adding extra lines beginning at uniform intervals between landmark points. This is not necessary in the scenario presented within this thesis as the system is given an entire sequence of frames to work with. With classification accuracy of over 95% a voting mechanism may be used to determine the correct classification. This could take the form of a binary voting mechanism, based on the best classification for each frame considered, or a system using the mean distance from average prototypes. Either system results in 100% classification accuracy for the data set used in this evaluation.

8.4 Conclusions

What is presented in this chapter is a simple solution to a relatively simple problem. The solution is robust for data sets of a realistic size and relatively computationally efficient (far faster than 25Hz frame rate on a PIII 450MHz [1]) for this size of data set (25 animals). The computational cost of the method increases in proportion to the number of animals in the database, however the method could still run in 'real time' with databases of the size of the number of animals on a large scale farm (up to 1000 animals). The classification accuracy may however be lower with such a large database. It is not practical, within the scope of this thesis, to evaluate the method with databases of this size and, in practice, this would not be necessary as animals are divided up into groups. The only limitation of the system is in the distinction of animals with very similar markings, or individuals whose markings only differ on the head or legs. The system could be extended to include head markings by using a separate centroid for head landmark points, however legs are a more difficult issue as they can be occluded in some frames. The system could also be extended to work with multiple cameras and thus include the markings on both sides of the animal. This may yield benefits, however similar individuals tend to be predominantly black all over and as such hard to classify by any method (including by hand). These individuals are thankfully a minority and this problem may be alleviated by careful selection of sub-groups of animals within the farm, or using alternative methods of identification such as electronic or text based tagging or even gait recognition.

[1]This does not include the object tracking stage.

Chapter 9

Conclusions

9.1 Summary of Work

The focus of the work within this thesis has been the modelling of deformable objects, in particular livestock, in both the spatial and temporal domains. It has been shown how these models may be used for the tracking and analysis of these objects using methods suitable for 'real time' application. Although this work was motivated by a particular application field (the analysis of Dairy Cows) there is a large body of novel theoretical work presented within this thesis. This may be separated into three main areas; spatial modelling, temporal modelling and object tracking and analysis.

The novel work on spatial modelling is in three areas; automatic acquisition of contour training data for 'learned' statistical models, a new type of Point Distribution Model based on a vector representation (the 'Vector Distribution Model') and the separation of spatial variation into pseudo-independent components using 'Delta Analysis'. The advantages of these individual methods have been discussed and evaluated within this thesis, and as a set of methods they represent an automatable way of building contour models that are both highly specific (to the object class used) and highly general (encapsulating the wide range of variation possible within the object classes used).

The novel work on temporal modelling involved the design, implementation and evaluation of two novel methods; 'History Space Classifiers' (HSC) and 'Multi-stream Cyclic Hidden Markov

Models' (MSCHMM). The former method is a variation of schemes proposed by Bulpitt *et al.* [13, 103] using Gaussian mixture models in place of the neural networks used previously. The latter method is a specific formulation of a discrete Hidden Markov Model scheme which uses a cyclic hidden state architecture and has multiple observational streams. These schemes were evaluated against the benchmarks of a first and second order Markov Chain and were found to outperform these benchmarks due to their ability to generalise to unseen input sequences. The MSCHMM method marginally outperforms the HSC method in terms of prediction ability and is also more computationally efficient than this method.

The novel work on object tracking and analysis is based around incorporating the spatial and temporal models described previously into a stochastic tracking/analysis framework. Two schemes were implemented and evaluated; 'Discrete Hypothesis Sampling' (DHS) and 'Re-sampling Condensation' (RC). DHS is a method based on the iterative hypothesise-sample-update hypothesis paradigm of Markov Chain Monte Carlo methods [36]. RC is an extension to Isard and Blake's Condensation algorithm [57] which incorporates a single stage of re-sampling (similar in concept to the DHS method, but using a genetic algorithm type method) into the Condensation framework. By the use of re-sampling both methods incorporate image information into sample location (an aspect that is missing from the original Condensation algorithm [57] and only partially addressed by ICondensation [58]). These methods have been evaluated against the original Condensation algorithm (using the same spatial and temporal models), however only the RC method outperforms this benchmark.

The novel work on animal identification is based on classifying grey level intensities along lines radiating from the centroid of an animals body to landmarks. These landmarks are determined by the object tracker described previously. A simple, but robust, nearest neighbour classifier has been described. This gives very high correct classification rates at low computational cost.

9.2 Discussion

The methods described within this thesis should be thought of as a toolkit of methods that may be incorporated with existing techniques. This idea has been demonstrated by the incorporation of the spatial and temporal models presented within a standard Condensation framework, however this is simply one example. The methods also work well as a set within the context of the Livestock and Human motion analysis applications presented within this thesis. This is as to be expected as these methods were designed with these applications in mind, however their use is not limited to these specific scenarios and there is potential for application in many areas of object analysis from biology to traffic analysis.

The spatial modelling scheme presented is particularly applicable to modelling objects that are well approximated by straight lines and have a complex, but highly constrained set of variations within the 2D field of view. Livestock are surprisingly well approximated by a piecewise linear representation and have a complex range of variation in both 3D and the 2D field of view used. There is an argument that says using a 3D model simplifies the spatial modelling problem, as the self occlusion problem disappears. This is correct, however this must be traded against the added complexity of 3D modelling schemes which, at current computer power, preclude their use in most ‘real time’ applications. The spatial variation present in livestock is reasonably complex in the 3D domain, with only the addition of the leg self occlusion problem in 2D. It could even be argued that the 2D model is simpler than the 3D approach if the field of view is restricted as in the example application. For this application a 3D model would add computational complexity without perceivable benefit, however this may not be the case for other applications. In such cases the Delta Analysis method described may be used to separate out the different sorts of variation within a 3D representation.

The HSC temporal model described is generally applicable to any temporal modelling problem where the spatial representation may be quantised. The MSCHMM method is only applicable to problems where the system is cyclic in nature, however using a different hidden state architecture would make it more globally applicable. The MSCHMM is a computationally more efficient scheme, in both training and use, than the HSC. Neither method matches the low computational cost of the Markov chain (although the MSCHMM approaches this when predicting from a known hidden state), however the extra generality and of these schemes is essential in many applications, ruling out the use of the simple Markov chain method.

The improved tracking results obtained by the RC algorithm over the original Condensation algorithm demonstrate the power of including current image information in the choice of sample location in stochastic model fitting schemes. This was pointed out by Isard and Blake in their I-

Condensation paper [58], however their solution involved including information from a secondary (low level) sensor and, as such, is not globally applicable. The re-sampling solution presented within this thesis is only applicable when the spatial variation may be separated into suitably independent components, however this is the case for many applications including Livestock, Humans and internal parts of the human body such as arterial structures and the heart. The method of determining optimal proportion of samples allocated to the re-sampling stage currently is based on comparing experimental results to a hand fitted ground truth. There is room for theoretical work in this area, however this is beyond the scope of this thesis.

The animal identification method presented is simple and robust under evaluation conditions. The method presented is satisfactory for the purposes intended, however the method may be extended to cope with more challenging scenarios. This would be achieved by using more landmark points for identification and a more complex classifier than the nearest neighbour method used currently.

9.3 Future Work

It has been discussed how the methods described in this thesis are widely applicable to a number of tasks within machine vision (and beyond). There lies much work in these areas, especially in human motion tracking where the modelling of human arm movements is an interesting problem due to their fairly unconstrained nature. Within the livestock analysis domain it would be interesting to investigate the use of 3D models, however the problem of data acquisition in this domain is a practical (but not a theoretical) obstacle as usual. Examining other, less spatially constrained, scenarios is also an interesting avenue of research, as this would really stretch the multiple spatial model paradigm to the full. Tackling the modelling of other animal ‘behaviours’ such as the behaviour exhibited when an animal is ‘on heat’ (oestrus) is also an interesting and challenging problem. This would require the use of more complex temporal models, possibly utilising hierarchies of models as used by Galata *et al.* [35] and Wren and Pentland [118]. Other avenues to be explored include the analysis of non machine vision data such as polyphonic musical recordings which exhibit an inherent component separation, due to the different instruments used, making them an ideal target for RC.

Appendix A

Procrustes Alignment of Translation, Scale and Rotation

Cootes *et al.* [21] present a method for aligning a set of shapes which is a form of Generalised Procrustes Analysis [39]. This is based on aligning pairs of shapes in an iterative manner. The original presentation (see appendix of [21]) contained a number of typographical errors and as such a corrected derivation is presented in this appendix.

A.1 Aligning Pairs of Shapes

To align a pair of shapes described by the points $x_{1,n}, y_{1,n}$ and $x_{2,n}, y_{2,n}$ (where n=1 to N) a rotation (θ), scale (s) and translation (t_x, t_y) must be chosen that minimise the weighted sum in equation A.1. This minimises the weighted square difference of the distances between pairs of points in the two shapes.

$$E = (x_1 - M(x_2))^T W (x_1 - M(x_2)) \tag{A.1}$$

where:

$$M\begin{pmatrix} x_{jk} \\ y_{jk} \end{pmatrix} = \begin{pmatrix} (s\cos\theta)x_{jk} - (s\sin\theta)y_{jk} + t_x) \\ (s\sin\theta)x_{jk} + (s\cos\theta)y_{jk} + t_y) \end{pmatrix} \tag{A.2}$$

Combining equations A.1 and A.2 gives:

$$E = \sum_{n=1}^{N} w_n \left((x_{1,n} - ((s\cos\theta)x_{2,n} - (s\sin\theta)y_{2,n} + t_x))^2 + (y_{1,n} - ((s\sin\theta)x_{2,n} + (s\cos\theta)y_{2,n} + t_y))^2 \right) \tag{A.3}$$

If the variables a_x and a_y are defined as $scos\theta$ and $ssin\theta$ respectively then equation A.3 may be re-written in terms of t_x, t_y, a_x or a_y as in equations A.4 to A.7.

$$E = \sum_{n=1}^{N} w_n \left(t_x^2 + t_x \left(a_x x_{2,n} - a_y y_{2,n} - x_{1,n} \right) + F_1(a_x, a_y, t_y) \right) \tag{A.4}$$

$$E = \sum_{n=1}^{N} w_n \left(t_y^2 + t_y \left(a_y x_{2,n} + a_x y_2 - y_1 \right) + F_2(a_x, a_y, t_x) \right) \tag{A.5}$$

$$E = \sum_{n=1}^{N} w_n \left(a_x^2 \left(x_{2,n}^2 + y_{2,n}^2 \right) + a_x \left(t_x x_{2,n} + t_y y_{2,n} - x_{1,n} x_{2,n} - y_{1,n} y_{2,n} \right) + F_3(a_y, t_x, t_y) \right) \tag{A.6}$$

$$E = \sum_{n=1}^{N} w_n \left(a_y^2 \left(x_{2,n}^2 + y_{2,n}^2 \right) + a_y \left(t_y x_{2,n} - t_x y_{2,n} + x_{1,n} y_{2,n} - y_{1,n} x_{2,n} \right) + F_4(a_x, t_x, t_y) \right) \tag{A.7}$$

Assuming all other parameters are static the minima of E for equations A.4 to A.7 are at the point of zero differential for t_x, t_y, a_x and a_y respectively as given in equations A.8 to A.11.

Note: These are minima and not maxima as the differential of the differential is positive in all cases

$$\sum_{n=1}^{N} w_n \left(2t_x + a_x x_{2,n} - a_y y_{2,n} - x_{1,n} \right) = 0 \tag{A.8}$$

$$\sum_{n=1}^{N} w_n \left(2t_y + a_y x_{2,n} + a_x y_2 - y_1 \right) = 0 \tag{A.9}$$

$$\sum_{n=1}^{N} w_n \left(2a_x \left(x_{2,n}^2 + y_{2,n}^2 \right) + t_x x_{2,n} + t_y y_{2,n} - x_{1,n} x_{2,n} - y_{1,n} y_{2,n} \right) = 0 \tag{A.10}$$

$$\sum_{n=1}^{N} w_n \left(2a_y \left(x_{2,n}^2 + y_{2,n}^2 \right) + t_y x_{2,n} - t_x y_{2,n} + x_{1,n} y_{2,n} - y_{1,n} x_{2,n} \right) = 0 \tag{A.11}$$

It is not always possible to satisfy equations A.8 to A.11 simultaneously and as such a least squares approach may be taken. This can be posed as a least squares matrix problem as in equation A.12.

$$\begin{pmatrix} X_2 & -Y_2 & 2W & 0 \\ Y_2 & X_2 & 0 & 2W \\ 2Z & 0 & X_2 & Y_2 \\ 0 & 2Z & -Y_2 & X_2 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} X_1 \\ Y_1 \\ C_1 \\ C_2 \end{pmatrix} \tag{A.12}$$

Where:

$$X_i = \sum_{k=1}^{N} w_k x_{i,k} \tag{A.13}$$

$$Y_i = \sum_{k=1}^{N} w_k y_{i,k} \tag{A.14}$$

$$Z = \sum_{k=1}^{N} w_k (x_{2,k}^2 + y_{2,k}^2) \tag{A.15}$$

$$W = \sum_{k=1}^{N} w_k \tag{A.16}$$

$$C_1 = \sum_{k=1}^{N} w_k (x_{1,k} x_{2,k} + y_{1,k} y_{2,k}) \tag{A.17}$$

$$C_2 = \sum_{k=1}^{N} w_k (y_{1,k} x_{2,k} - x_{1,k} y_{2,k}) \tag{A.18}$$

A.2 Aligning Pairs of Shapes in Scale and Translation only

It is sometimes desirable to align shapes in scale and translation only to preserve rotational information. In such cases equation A.3 simplifies to the equation given in A.19.

$$E = \sum_{n=1}^{N} w_n \left((x_{1,n} - (s x_{2,n} + t_x))^2 + (y_{1,n} - (s y_{2,n} + t_y))^2 \right) \tag{A.19}$$

Writing equation A.19 in terms of t_x, t_y and s respectively gives equations A.20 to A.22 respectively.

$$
\begin{aligned}
E &= \sum_{n=1}^{N} w_n \left(t_x^2 + t_x(-x_{1,n} + sx_{2,n}) + F_5(s,t_y)\right) && \text{(A.20)}\\
E &= \sum_{n=1}^{N} w_n \left(t_y^2 + t_y(-y_{1,n} + sy_{2,n}) + F_6(s,t_x)\right) && \text{(A.21)}\\
E &= \sum_{n=1}^{N} w_n \left(s^2(x_{2,n}^2 + y_{2,n}^2) + s(-x_{1,n}x_{2,n} - t_x x_{2,n} - y_{1,n}y_{2,n} - t_y y_{2,n}) + F_7(t_x,t_y)\right) && \text{(A.22)}
\end{aligned}
$$

As in the previous section the minima of equations A.20 to A.22 are found by differentiating by s, t_x and t_y respectively and equating to zero. This results in the linear system given in equation A.23.

$$
\begin{pmatrix} X_2 & 2W & 0 \\ Y_2 & 0 & 2W \\ 2Z & X_2 & Y_2 \end{pmatrix} \begin{pmatrix} s \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} X_1 \\ Y_1 \\ C_1 \end{pmatrix} \tag{A.23}
$$

A.3 Aligning Sets of Shapes

To build models of shape sets of shapes must be aligned. Cootes *et al.* present the following method for aligning a complete set of shapes:

1. Align each shape in the set to the first shape in the set using the method in section A.1 or section A.2.
2. Calculate the mean of the transformed shapes.
3. Either align the mean to a default scale, orientation and origin or align the mean to the unaligned first shape using the method in section A.1 or section A.2.
4. Align each shape to the adjusted mean using the method in section A.1 or section A.2.
5. Repeat (2)-(4) until convergence.

Appendix B

Comparing Probability Density Function Probabilities

If it is the case that there are multiple Probability Density Functions (PDFs) representing the outputs from multiple processes it is difficult to assign probabilities that a particular observation results from a particular process. In this appendix a proof is presented that shows it is only correct to compare the magnitudes of two PDFs if the probability that a sample results from either distribution is equal over the range of the distributions.

B.1 The Proof

Consider two PDFs ϕ and ψ. Integrating over a range of values R (as shown in figure B.1 in 1D) gives the probability that the distribution results in an observation contained within R as given in equations B.1 and B.2. For small values of R this is approximately the product of R and the PDF magnitude local to R.

$$P(x \in R|\phi) = \int_R \phi(x)dx \approx R \times \phi(x_R) \tag{B.1}$$

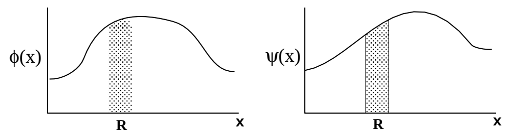


Figure B.1: Comparing Two PDFs in 1D

$$P(x \in R|\psi) = \int_R \psi(x)dx \approx R \times \psi(x_R) \tag{B.2}$$

Using Bayes' rule it is possible to calculate the probability of an observation resulting from ϕ as:

$$P(\phi|x \in R) = \frac{P(x \in R|\phi) \times P(\phi)}{P(x \in R)} \tag{B.3}$$

If the observation must come from either ϕ or ψ:

$$P(x \in R) = P(x \in R|\phi) \times P(\phi) + P(x \in R|\psi) \times P(\psi) \tag{B.4}$$

Substituting equation B.4 into equation B.3 gives:

$$P(\phi|x \in R) = \frac{P(x \in R|\phi) \times P(\phi)}{P(x \in R|\phi) \times P(\phi) + P(x \in R|\psi) \times P(\psi)} \tag{B.5}$$

In the case where $P(\phi) = P(\psi)$ this simplifies to:

$$P(\phi|x \in R) = \frac{P(x \in R|\phi)}{P(x \in R|\phi) + P(x \in R|\psi)} \tag{B.6}$$

Substituting equations B.1 and B.2 into equation B.6 gives:

$$P(\phi|x \in R) \approx \frac{\phi(x_R)}{\phi(x_R) + \psi(x_R)} \tag{B.7}$$

This shows that is valid to compare PDF magnitudes if the probability that an observation results from the two independent PDFs is equal over the range of interest (i.e. the probabilities are independent of the observation).

Appendix C

Working with Hidden Markov Models

Hidden Markov Models (HMMs) were introduced in chapter 2. In this appendix the maths required to work with HMMs is presented. Rabiner [85] states that there are three problems to be solved when using HMMs; evaluation, recovery of hidden states and training. The standard solutions to these problems are presented within this appendix in addition to notes on ‘problem 4’; the prediction of future events using HMMs. The following notation (taken from [85]) will be used in this appendix:

λ	$=$	A HMM Consisting of Parameters A, B and π.
A	$=$	The Hidden State Transition Probability Matrix
B	$=$	The Observation Probability Matrix
π	$=$	The Initial state distribution
a_{ij}	$=$	Member i,j of A representing the transition from state i to state j
$b_j(k)$	$=$	Member j,k of B representing the probability of observation k given hidden state j
M	$=$	The number of states of the model
N	$=$	The number of distinct observational symbols
O	$=$	An observation sequence $O_1, O_2, \cdots, O_T$
Q	$=$	A hidden state sequence $q_1, q_2, \cdots, q_T$

C.1 Problem 1: Evaluation of HMMs

The most straightforward way of calculating the probability that an observed sequence was produced by a given HMM ($P(O|\lambda)$) is through enumerating every possible state sequence of length T (the number of observations in O). Considering an arbitrary hidden state sequence (Q) the probability of the observation sequence (O) resulting from the HMM (λ) is the multiple of the observation probabilities over the entire sequence as given in equation C.1.

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) \tag{C.1}$$

Assuming statistical independence of observations this gives us equation C.2.

$$P(O|Q,\lambda) = b_{q_1}(O_1) \times b_{q_2}(O_2) \times \cdots \times b_{q_T}(O_T) \tag{C.2}$$

The probability of such a sequence (Q) can be written as in equation C.3.

$$P(Q|\lambda) = \pi_{q_1} \times a_{q_1,q_2} \times a_{q_2,q_3} \times \cdots \times a_{q_{T-1},q_T} \tag{C.3}$$

The joint probability of the observation (O) and the state sequence (Q) occurring given the HMM (λ) is simply the product of equations C.2 and C.3. Summing this over all possible sequences gives us the probability of the observation sequence (O) given the HMM (λ) as shown in equation C.4.

$$P(O|\lambda) = \sum_{allQ} P(O|Q,\lambda) \times P(Q|\lambda) \tag{C.4}$$

The evaluation of equation C.4 is computationally expensive (of the order of $2TN^T$), however this can be speeded up using the ‘Forward-Backward Procedure’. Consider the ‘forward variable’ $\alpha_t(i)$ which defines the probability that a given partial observation sequence ($O_1, O_2, \cdots, O_t$) and a final hidden state (S_i) result from a given HMM (λ) as in equation C.5.

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i|\lambda) \tag{C.5}$$

The probability of the complete observation sequence (O) given the HMM (λ) is simply the sum of alpha over all hidden states as in equation C.6.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{C.6}$$

$\alpha_t(i)$ can be found iteratively as in equations C.7 and C.8 thus reducing the computation of $P(O|\lambda)$ from order $2TN^T$ in equation C.4 to order N^2T.

$$\alpha_1(i) = \pi_i b_i(O_1) \tag{C.7}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \tag{C.8}$$

C.2 Problem 2: Recovery of Hidden States

In the general case there is no single correct hidden state sequence for a given observation sequence however an ‘optimal’ sequence may be calculated. There are several possible optimality criteria, however the most widely used is the maximisation of $P(Q|O,\lambda)$ (which is equivalent to maximising $P(Q,O|\lambda)$). This is done using the Viterbi Algorithm [112] to find the single best state sequence ($Q = q_1, q_2, \cdots, q_T$) for the given observation sequence ($O = O_1, O_2, \cdots, O_T$). This is based on dynamic programming.

The variable $\delta_t(i)$ is defined as the highest probability along a single path, at time t, which accounts for the first t observations and ends in state S_i as in equation C.9.

$$\delta_t(i) = \max_{q_1,q_2,\cdots,q_{t-1}} P(q_1, q_2, \cdots, q_t = i, O_1, O_2, ...O_t|\lambda) \tag{C.9}$$

$\delta_t(i)$ may be calculated iteratively over time as in equations C.10 and C.11.

$$\delta_1(i) = \pi_i bi(O_1) \tag{C.10}$$

$$\delta_{t+1}(i) = \max_{1 \leq i \leq N} [\delta_t(i) a_{i,j}] b_j(O_{t+1}) \tag{C.11}$$

To retrieve the state sequence it is necessary to keep track of the argument that maximises $\delta_t(i)$ for each t and j. This is stored in an array $\psi_t(j)$ as in equations C.12 and C.13.

$$\psi_1(i) = 0 \tag{C.12}$$

$$\psi_t(j) = \underset{1 \leq i \leq N}{\text{argmax}} [\delta_{t-1}(i) a_{i,j}] \tag{C.13}$$

The sequence is then extracted by starting at the most probable hidden state at time T and 'back-tracking' through the sequence until the beginning (time 1) as in equations C.14 and C.15.

$$q_T = \underset{1 \leq i \leq N}{\text{argmax}} [\delta_T(i)] \tag{C.14}$$

$$q_t = \psi_{t+1}(q_{t+1}) \tag{C.15}$$

C.3 Problem 3: HMM Training

According to Rabiner [85] there is no known way to determine analytically the model parameters ($\lambda = (A, B, \pi)$) which maximises the probability of an observation sequence (O). It is however possible to determine a local maximum of $P(O|\lambda)$ by taking an initial estimate to λ and refining it using the Baum-Welch re-estimation procedure (equivalent to the Expectation Maximisation (EM) algorithm). It is possible that a global search method such as genetic algorithms or simulated annealing could be used to optimise the model parameters to maximise $P(O|\lambda)$, however Baum-Welch is generally used due to its high speed of operation compared with such methods.

To describe the Baum-Welch re-estimation algorithm it is necessary to define the 'backwards-variable' $\beta_t(i)$ which is the probability of a partial observation sequence ($O = O_{t+1}, O_{t+2}, \cdots, O_T$) given a state S_i at time t and the model λ as in equation C.16.

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \cdots, O_T | q_t = S_i, \lambda) \tag{C.16}$$

This may be calculated iteratively in a similar manner to the forward variable ($\alpha_t(i)$) described in section C.1 as shown in equations C.17 and C.18.

$$\beta_T(i) = 1 \tag{C.17}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} bj(O_{t+1}) \beta_{t+1}(j) \tag{C.18}$$

The variable $\xi_t(i,j)$ must also be defined as the probability of being in state S_i at time t and state S_j at time $t+1$ as in equation C.19.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{C.19}$$

Using Bayes theorem $\xi_t(i,j)$ may be given in terms of the forward-variable ($\alpha_t(i)$) and the backward-variable ($\beta_t(i)$) as in equation C.20.

$$\begin{aligned} \xi_t(i,j) &= \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \end{aligned} \tag{C.20}$$

If $\gamma_t(i)$ is the probability of being in state S_i at time t, given the observation sequence (O) and the model (λ), this may be calculated by summing $\xi_t(i,j)$ over j as in equation C.21.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{C.21}$$

The sum of $\gamma_t(i)$ over time (excluding time t=T) may be interpreted as 'the expected number of transitions from S_i'. Similarly the sum of $\xi_t(i,j)$ may be interpreted as 'the expected number of transitions from S_i to S_j'. Utilising these definitions the Baum-Welch re-estimation procedure may be defined as in equations C.22 to C.24.

$$\begin{aligned}
\bar{\pi_i} &= \text{expected frequency in state } S_i \text{ at time t=1} \\
&= \gamma_1(i) && \text{(C.22)} \\
\bar{a_{ij}} &= \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} \\
&= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} && \text{(C.23)} \\
\bar{b_i}(k) &= \frac{\text{expected times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \\
&= \frac{\sum_{t=1,O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} && \text{(C.24)}
\end{aligned}$$

It can be shown (see [85]) that this re-estimation procedure will always increase (or not alter) $P(O|\lambda)$. The procedure can easily be extended to multiple observation sequences simply by extending equations C.23 and C.24 to sum over multiple sequences and taking an average over all sequences for equation C.22.

C.4 Problem 4: Prediction

Prediction of future observations is not an area that is widely covered in the HMM literature. This is possibly due to the lack of need for prediction in the principal application of HMMs (speech recognition) and the problems with prediction highlighted by the ‘hello’ example in chapter 2. There are however several ways of using a HMM for prediction depending on knowledge of the hidden state and computational considerations.

If the current hidden state (S_i) is known prediction is very efficient as it is simply a matter of traversing the Markov chain. A complete probabalistic prediction would involve evaluating the probability of being in each hidden state at the next time step and multiplying the observation probabilities (B) by these probabilities as in equation C.25.

$$P(O_{t+1} = v_k | q_t = S_i, \lambda) = \sum_{j=1}^{M} a_{ij} b_j(k) \tag{C.25}$$

In the general case this requires $N \times M$ multiplications to calculate $P(O_{t+1} = v_k | q_t = S_i, \lambda)$ for all possible observations, however in specific cases possible transitions may be limited decreasing the computational cost of this operation. If a fast random number generator is available an alternative approach is it perform a stochastic prediction in which a single next hidden state is selected stochastically using the transition probability matrix (A). From this observation probabilities may be deduced using the observation probability matrix (B) or a single observation may be selected either stochastically or by maximum likelihood. Stochastic prediction is particularly useful in a multiple prediction scheme such as ‘Condensation’ [57] in which multiple hypothesis are propogated from one time step to the next.

If the current hidden state is not available this must be estimated from current and past observations as in section C.2. The Viterbi algorithm is one method for performing this estimation. This produces a single estimate for the current hidden state which may be used as previously. Alternatively the Viterbi algorithm gives a matrix of relative state probabilities ($\delta_t(j)$). The relative probabilities at time $t = T$ may be used to perform a complete probabalistic prediction as in equation C.26.

$$P(O_{t+1} = v_k|\lambda) = \sum_{i=1}^{N} \left[\frac{\delta_T(i)}{\sum_{l=1}^{N} \delta_T(l)} \sum_{j=1}^{M} a_{ij} b_j(k) \right] \tag{C.26}$$

Appendix D

Principal Components Analysis and Linear Discriminant Analysis

D.0.1 Principal Components Analysis

Principal components analysis (PCA) is a method of finding a set of orthogonal linear vectors (Eigenvectors) that code a data set such that the maximum variance possible is encapsulated in each successive vector. PCA is effectively a rotation (and translation) of the data co-ordinate system such that this criterion is met. This is performed by finding the set of Eigenvectors and corresponding Eigenvalues that solve the Eigen equation D.1.

$$Ax = \lambda x \tag{D.1}$$

Where:
$A =$A D$\times$D covariance matrix (symmetric, real and known)
$x =$An ‘Eigenvector’ (real and unknown)
$\lambda =$An ‘Eigenvalue’ (real and unknown)

The covariance matrix (A) is derived from the data set as in equation D.2.

$$A = \frac{1}{N}\sum_{n=1}^{N}(d_n - \bar{d})(d_n - \bar{d})^T \tag{D.2}$$

Where:

$d_n =$ Data item number n (dimensionality D)

$\bar{d} =$ The mean of all data items (dimensionality D)

$N =$ The number of data items

There are many methods in the mathematical literature for finding the set of solutions for equation D.1 (see for example [84]) and several widely available pieces of software that implement these methods. As such this appendix will not deal with such methods. Once a set of Eigenvectors and Eigenvalues has been computed, data from the training set, or novel data, may be coded using equation D.3.

$$b = P^T(d - \bar{d}) \tag{D.3}$$

Where:

$b =$ A vector of Eigen components

$P =$ A matrix of Eigenvectors $(x_1 x_2 .. x_N)$

Equation D.3 is derived by re-arranging equation D.4, which defines the relationship between the data and the Eigen components.

$$d = \bar{d} + Pb \tag{D.4}$$

This re-arrangement is based on the knowledge that the matrix P is orthogonal and hence:

$$P^{-1} = P^T \tag{D.5}$$

PCA may be used as a data dimensionality reduction tool by discarding components of the vector b that correspond to Eigenvectors with small (or zero) Eigenvalues (N.B. the Eigenvalue is equal to the variance of the data set along the corresponding Eigenvector). A good approximation to novel data items using this truncated representation will only be obtained if the novel data has similar variance characteristics to the training data set.

PCA works well for data sets with an underlying structure, particularly if this is linear in nature. If the underlying structure of the data is sufficiently linear the Eigenvectors can map to particular aspects of the data variation and, as such, be used in analysis and classification applications. For most data sets in machine vision the underlying structure is non-linear, however PCA may still be a powerful tool in dimensionality reduction. In such cases particular aspects of the variation are not coded by individual Eigenvectors, but approximated by combinations of multiple vectors. This can lead to regions of the 'Eigenspace' defined by the Eigen components (b) that do not represent the data set well being poorly separated from regions that do. This precludes such an Eigenspace being easily used in applications that require the Eigenspace to provide data classification (a number of ways round this are discussed in chapters 2 and 7). In the case of such 'non-linear' data sets Eigenvectors with small Eigenvalues do not necessarily represent noise in the data set and may be required for fine representation of the data set. Great care is needed in the choice of truncation level which is often a subjective trade off between representation accuracy, dimensionality reduction and noise exclusion.

D.0.2 Linear Discriminant Analysis

Canonical analysis or Linear Discriminant Analysis [69] is a statistical technique used on multivariate data sets which contain multiple data classes. The objective of this technique is to separate out inter-class variation from intra-class variation by creating a new space in a similar way to PCA. For this method two covariance matrices are required: an 'intra-class' matrix (S_w) and an 'inter-class' matrix (S_b). These are formed as in equations D.6 and D.7 respectively. Once the covariance matrices have been formed the generalised Eigen equation given in equation D.8 is solved to give a set of Eigenvectors which describe a new space. It should be noted that this space is not necessarily optimised to allow dimensionality reduction as with PCA, however the first components describe inter-class variation and can as such be used as a data classifier. This has been used recently for face recognition [32, 104], image retrieval systems [104] and gait classification [51].

$$S_w = \frac{1}{n_t} \sum_{i=1}^{n_c} \sum_{j=1}^{n_i} (\mathbf{y_{i,j}} - \mu_\mathbf{i})(\mathbf{y_{i,j}} - \mu_\mathbf{i})^T \tag{D.6}$$

$$S_b = \frac{1}{n_t} \sum_{i=1}^{n_c} (\mu_\mathbf{i} - \mu_\mathbf{y})(\mu_\mathbf{i} - \mu_\mathbf{y})^T \tag{D.7}$$

$$S_b E = \lambda S_w E \tag{D.8}$$

Where:

S_w = Intra-class Covariance Matrix	n_i = Number of data items in class i
S_b = Inter-class Covariance Matrix	$\mathbf{y_{i,j}}$ = Data item j of class i
n_t = Total number of data items	$\mu_\mathbf{i}$ = Mean vector for class i
n_c = Number of classes	$\mu_\mathbf{y}$ = Mean vector for entire data set

It can be seen from equation 2 that the notion of ‘inter-class variance’ is based on the variance of the means. This means that if the means of the data sets do not have comparable intra-class characteristics, intra-class variation will be included in the optimisation (to some extent). In this situation the result of canonical analysis is a compromise as intra-class variation is both maximised and minimised simultaneously. Care should be taken when selecting the data set to ensure that this is not the case.

D.0.3 Combining PCA and LDA

There have been several publications [104, 4, 51] in which PCA and LDA are combined by performing LDA on the projections of a data set in a truncated Eigenspace produced using PCA. The motivation given by Swet and Weng [104] and Belhumeur *et al.* [4] for this is to ensure that the LDA within covariance matrix (S_w in equation D.6) is non-singular in cases where the dimensionality is higher than the number of samples (for example Eigenfaces). Huang *et al.* [51] use this technique to combine the dimensionality reduction properties of PCA with the class separation properties of LDA by having a large truncation of the PCA Eigenspace. Truncating the PCA Eigenspace limits the LDA space to a subspace of the original data space, the higher the truncation the smaller the subspace. The optimal set of perpendicular discriminating vectors for a data set may lie outside this sub-space, in which case the discriminating power of the solution will be sub-optimal. As such the effect of truncation will be heavily dependent on the data sample used to build the spaces in terms of how well this describes the variation within the complete data set or probability distribution and how much correlation there is within the data set. No evaluation of the effect of high truncation on the discriminating power of these spaces has been performed at this time to the author's knowledge.

Appendix E

Vector Quantisation

Vector Quantisation is a term that describes any method that produces a set of prototypes that in some way describe a multivariate data set or process. These methods were originally developed for the purposes of data compression but are now widely used in many applications for the representation of the inherent structure in a multivariate data set or process (e.g. the structure present in speech [68]). These sets of prototypes effectively approximate to a probability density function for the data set or process that they are constructed from. Grey [41] states that the purpose of a 'Vector Quantiser' is to minimise the expectation of a distortion measure which is a function of an input vector and the prototype that it is quantised to. In most cases the expected value of this measure ($d(x, \hat{x})$) is taken as the long term sample average as given in equation E.1.

$$\lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} d(x, \hat{x}) \tag{E.1}$$

The formulation of the distortion measure is application specific, however a common measure is the squared error distortion measure (i.e. the square of the Euclidean distance between the input vector and the prototype). Equation E.1 is in practice replaced by the average of a finite sequence which is minimised by the vector quantiser. It is often impossible to minimise this function directly, however a locally optimal solution may be found by taking an initial estimate and refining iteratively using the generalised Lloyd or LBG algorithm [70]:

1. Encode the training sequence into a sequence of symbols using the current set of vector quantisation prototypes so as to minimise the distortion measure for each data item. If average distortion is small enough, quit.

2. Replace the old set of vector quantisation prototypes by the centroid† of all data items that mapped to a given prototype. Go to step one.

† Note: For the squared error distortion measure this is simply the Euclidean centroid, however for more complex distortion measures other centroids are required (this is discussed in [41]).

If an absolute error distortion measure is used this algorithm is identical to the k-means clustering algorithm [72]. The selection of initial prototype locations may be performed completely randomly, by randomly selecting a number of data items or by evenly distributing prototypes (or any other sensible method). The choice of the number of prototypes is a difficult task and is often tackled by starting with too few and splitting prototypes that represent data items spanning a wide area of the representation space into two seperate prototypes after convergence and repeating the clustering operation (see [70]). An alternative approach is to start with too many prototypes and merge adjacent prototypes. The quantisation produced by the algorithm is a locally optimal solution to the minimisation of equation E.1 over a finite sample set and as such depends on the quality of the initial prototype estimate. ‘Fuzzy’ versions of the generalised Lloyd algorithm have been proposed, such as the fuzzy-c-mean algorithm [28], that assign data items to multiple prototypes with a weight for each one. These are less susceptible to small changes in initial prototype location but still produce a locally optimal solution. Alternatives to the generalised Lloyd algorithm have been proposed and are discussed in the following sections.

E.1 The Self-Organising Map

The Self-Organising Map [68] is a type of neural network trained by unsupervised competitive learning that also serves as a vector quantiser. This method differs from the generalised Lloyd algorithm in that the training data is presented to the vector quantiser one item at a time rather than as a complete set. Prototype locations are updated after the presentation of each data item. Prototypes in the area (N_c) local to the prototype with the lowest distortion for each data item are updated as in equation E.2.

$$m_i(t+1) = \left\{ \begin{array}{l} m_i(t) + \alpha(t)[x(t) - m_i(t)] \\ m_i(t) \end{array} \right\} \begin{array}{l} \text{if } i \in N_c(t) \\ \text{if } i \notin N_c(t) \end{array} \tag{E.2}$$

The size of the local area ($N_c(t)$) and adaptation gain ($\alpha(t)$) are usually defined as monotonically decreasing functions of time to allow coarser adaptation initially with increasingly fine adaptation over time. Prototypes are defined as an interconnected grid to allow the specification of the local area ($N_c(t)$). This method produces a quantisation with more prototypes in areas of high sample density / high probability with few prototypes in areas of low sample density / low probability. This makes it an ideal method for forming vector quantisations for use in describing a probability distribution (e.g. [63]).

E.2 Stochastic Optimisation

The methods described previously are prone to finding locally optimal solutions to the problem of minimising vector quantisation distortion as they are essentially local optimisation techniques. A number of methods have been suggested that take a global approach to the minimisation of average distortion using stochastic search techniques such as simulated annealing [111] and genetic algorithms [27]. These algorithms can, in many cases, produce a more globally optimal solution than the local search methods described previously. They also have the advantage that other qualities (in addition to average distortion) may be included in the optimisation, for example the relative distribution of prototypes. The disadvantage of these methods is that they require a number of parameters to be set and are computationally very expensive compared to the local search schemes, taking a long time to reach an optimal solution for many complex data sets.

E.3 Vector Quantisation For Livestock Analysis

The aim of vector quantisation, within the scope of this thesis, is to produce a set of prototypes for the object tracking and temporal modelling algorithms to work with. These algorithms require that all data items are represented (to a minimum distortion) by at least one prototype. In addition computational efficiency considerations dictate that this should be performed with the minimum number of prototypes. When using the History Space Classifier temporal model (presented in chapter 6) it is also desirable that prototypes are not closely spaced as this can lead to ambiguity in the 'winner takes all' approach if there is a deficiency in the quantity of training data. Given these additional considerations the use of stochastic optimisation techniques is attractive, however when these considerations are built into the function to be optimised the problem becomes complex such that the algorithms take a great deal of time to converge on a solution. These algorithms require the setting up of parameters which is a task generally performed iteratively by running the algorithm and adjusting the parameters as a result of the output produced. In the context of the vector quantisation tasks in this thesis this is a very time consuming process and it was considered that a simple clustering algorithm (k-means) was adequate for these tasks. The results of several runs of k-means clustering algorithm (with random initialisation) were used, a result that met the distortion criteria (a maximum on Euclidean distance of any data item from a prototype) selected and unnecessary prototypes (in terms of meeting this criteria) removed. This method, although crude, was effective and orders of magnitude more computationally efficient than stochastic methods.

References

[1] S. Basu, N. Oliver, and A. Pentland. 3d modeling and tracking of human lip motions. In *Proc. Int. Conference on Computer Vision*, pages 337–343, 1998.

[2] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In *Proc. IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 194–199, 1994.

[3] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *European Conference on Computer Vision*, pages 299–308. Springer Verlag, 1994.

[4] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:711–720, 1997.

[5] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1996.

[6] M. Black and A. Jepson. Recognizing temporal trajectories using the condensation algorithm. In *Proc. Initernational Conference on Automatic Face and Gesture Recognition*, pages 16–21, 1998.

[7] M. Black and Y. Yacoob. Tracking and recognising rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. International Conference on Computer Vision*, pages 374–381, 1995.

[8] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11:127–145, 1993.

[9] M. Brand. Shadow puppetry. In *Proc. International Conference on Computer Vision*, pages 1237–1244, 1999.

[10] C. Bregler. Learning and recognising human dynamics in video sequences. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 568–574, 1997.

[11] C. Bregler and S. Omohundro. Surface learning with applications to lipreading. *Advances in neural information processing systems 6*, pages 43–50, 1994.

[12] R. Bubley and M. Dyer. Path coupling: a technique for proving rapid mixing in markov chains. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 223–231, 1997.

[13] A. Bulpitt and N. Allinson. Motion recognition using moving light displays. In *Proc. IEE 2nd International Conference on Artificial Neural Networks*, 1991.

[14] M. Buteau and S. Makram-Ebeid. A computer vision approach to emphysema detection and quantification. In *Proc. International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1089–1090, 1996.

[15] L. Campbell and A. Bobick. Recognition of human body motion using phase space constraints. In *Proc. International Conference on Computer Vision*, pages 624–630, 1995.

[16] M. La Cascia, J. Isidoro, and S. Sclaroff. Head tracking via robust registration in texture map images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 508–514, 1998.

[17] T. Cootes, C. Beeston, G. J. Edwards, and C.J. Taylor. A unified framework for atlas matching using active appearance models. In *Proc. Int. Conf. on Image Processing in Medical Imaging*, pages 322–333, 1999.

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

[19] T. Cootes and C. Taylor. Active shape models - ‘smart snakes’. In *Proc. British Machine Vision Conference*, pages 266–275, 1992.

[20] T. Cootes and C. Taylor. A mixture model for representing shape variation. In *Proc. British Machine Vision Conference*, pages 110–119, 1997.

[21] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Training models of shape from sets of examples. In *Proc. British Machine Vision Conference*, pages 9–18, 1992.

[22] D. Davies, P. Palmer, and M. Mirmehdi. Detection and tracking of very small low contrast objects. In *Proc. British Machine Vision Conference*, pages 599–608, 1998.

[23] J. Davis and A. Bobick. The representation and recognition of action using temporal templates. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[24] A. Davison and D. Murray. Mobile robot localisation using active vision. In *Proc. European Conference on Computer Vision*, pages 809–825, 1998.

[25] K. de Souza, J. Kent, and K. Mardia. Stochastic templates for aquaculture images and a parallel pattern. *Applied Statistics*, 48:211–227, 1999.

[26] D. DeCarlo and D.Metaxas. Blended deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):443–448, 1996.

[27] V. Delport and M. Koschorreck. Genetic algorithm for codebook design in vector quantisation. *Electronics Letters*, 31(2):84–85, 1995.

[28] V. Delport and D. Liesch. Fuzzy-c-mean algorithm for codebook design in vector quantisation. *Electronics Letters*, 30(13):1025–1026, 1994.

[29] A. Dempster and D. Rubin N. Laird. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39:1–38, 1977.

[30] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1973.

[31] G. Edwards, A. Lanitis, C. Taylor, and T. Cootes. Statistical models of face images - improving specifity. *Image and Vision Computing*, 1997.

[32] K. Etemad and R. Chellappa. Discriminant analysis for recognition of human face images. In *First International Conference on Audio and Video Based Biometric Person Authentication*, pages 127–142, 1997.

[33] O. Faugeras, N. Ayache, and Z. Zhang. A preliminary investigation of the problem of determining ego- and object motions from stereo. In *Proc. IEEE International Conference on Pattern Recognition*, pages 242–246, 1988.

[34] J. Ferryman, S. Maybank, and H. Worrall. Visual surveillance for moving vehicles. In *Proc. IEEE Workshop on Visual Surveillance*, pages 73–80, 1998.

[35] A. Galata, N. Johnson, and D. Hogg. Learning variable length markov models of behaviour. *Computer Vision and Image Understanding*, 2001. To appear.

[36] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

[37] S. Gong. Visual observation as reactive learning. In *Proc. SPIE International Conference on Adaptive Learning Systems*, pages 175–186, 1992.

[38] N. Gordon, D. Salmond, and R. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F : Radar and Signal Processing*, 140(2):107–113, 1993.

[39] J. Gower. Generalised procrustes analysis. *Psychometrika*, 40:33–51, 1975.

[40] H. Graf, E. Cosatto, and T. Ezzat. Face analysis for the synthesis of photo-realistic talking heads. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 189–194, 2000.

[41] R. Grey. Vector quantization. *IEEE ASSP Magazine*, 2(1):4–29, 1984.

[42] I. Guyon and F. Pereira. Design of a linguistic postprocessor using variable memory length markov models. In *Proc. International Conference on Document Analysis and Recognition*, pages 454–457, 1995.

[43] I. Haritaoglui, D. Harwood, and L. Davis. W4: Who? when? where? a real time system for detecting and tracking people. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 222–227, 1998.

[44] A. Heap and D. Hogg. Extending the point distribution model using polar co-ordinates. *Image and Vision Computing*, 14(8):589–599, 1995.

[45] A. Heap and D. Hogg. Towards 3d hand tracking using a deformable model. In *Proc. IEEE International Conference on Automatic Face and Gesture Recognition*, pages 140–145, 1996.

[46] A. Heap and D. Hogg. Improving specificity in PDMs using a hierarchical approach. In *Proc. British Machine Vision Conference*, volume 1, pages 80–89, 1997.

[47] D. Heymer, P. McLauchlan, H. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[48] A. Hill and C. Taylor. Automatic landmark generation for point distribution models. In *Proc. British Machine Vision Conference*, volume 2, pages 429–438, 1994.

[49] D. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1:5–20, 1983.

[50] J. Hu, W. Turin, and M. Brown. Language modelling using stochastic automata with variable length contexts. *Computer Speech and Language*, 11(1):1–16, 1997.

[51] P. Huang, C. Harris, and M. Nixon. Recognising humans by gait via parametric canonical space. In *ICSC Symposium on Engineering of Intelligent Systems*, 1998.

[52] P. Huang, C. Harris, and M. Nixon. Comparing different template features for recognising people by their gait. In *British Machine Vision Conference*, pages 639–648, 1999.

[53] T. Huang and A. Netravali. Motion and structure from feature correspondences: a review. *Proceedings of the IEEE*, 82(2):252–268, 1994.

[54] T. Ikeda, S. Ohnaka, and M. Mizoguchi. Traffic measurement with a roadside vision system-individual tracking of overlapped vehicles. In *Proc. International Conference on Pattern Recognition*, pages 859–864, 1998.

[55] C. Ingrassia, P. Windyga, and M. Shah. Segmentation and tracking of coronary arteries. In *Proc. Joint BMES/EMBS Conference*, page 203, 1999.

[56] S. Intille, J. Davis, and A. Bobick. Real-time closed-world tracking. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 697–703, 1997.

[57] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[58] M. Isard and A. Blake. Icondensation: Unifying low-level tracking in a stochastic framework. In *Proc. 5th European Conference on Computer Vision*, volume 1, pages 893–908, 1998.

[59] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc. International Conference on Computer Vision*, pages 107–112, 1998.

[60] G. Jacob, J. Noble, and A. Blake. Robust contour tracking in echocardiographic sequences. In *Proc. International Conference on Computer Vision*, pages 408–413, 1998.

[61] M. Jerrum and A. Sinclair. *The Markov Chain Monte Carlo method: an approach to approximate counting and integration*. PWS Publishing, 1996.

[62] N. Johnson, A. Galata, and D. Hogg. The acquisition and use of interaction behaviour models. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 866–871, 1998.

[63] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996.

[64] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

[65] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, pages 35–45, 1960.

[66] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. First International Conference on Computer Vision*, pages 259–268, 1989.

[67] V. Kettnaker and M. Brand. Minimum entropy models of scene activity. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 281–286, 1999.

[68] T. Kohonen. The self organising map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[69] W. Krzanowski. *Kendall's library of statistics: Multivariate analysis*. Edward Arnold, 1994.

[70] Y. Linde, A. Buzo, and R. Grey. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

[71] M. Lyons, J. Budynek, A. Plante, and S. Akamatsu. Classifying facial attributes using a 2d gabor wavelet representation and discriminant analysis. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 202–207, 2000.

[72] J. MacQueen. Some method for the classification and analysis of multivariate observations. In *Proc. Fifth Berkley Symposium on Mathematics, Statistics and Probability*, pages 281–296, 1967.

[73] J. Marchant, C. Schofield, and R. White. Pig growth and conformation monitoring using image analysis. *Animal Science*, 68(1):141–150, 1999.

[74] K. Mardia, J. Kent, and A. Walder. Statistical shape models in image analysis. In *Proc. 23rd Symposium on the Interface*, pages 550–557, 1991.

[75] S. McKenna, S. Jabri, Z. Duric, and H. Wechsler. Tracking interacting people. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 348–353, 2000.

[76] D. Metaxas. Deformable model and hmm-based tracking, analysis and recognition of gestures and faces. In *Proc. IEEE International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pages 136–140, 1999.

[77] H. Murase and R. Sakai. Moving object recognition in eigenspace representation: Gait analysis and lip reading. *Pattern Recognition Letters*, 17:155–162, 1996.

[78] J. Nash, J. Carter, and M. Nixon. Extraction of moving articulated-objects by evidence gathering. In *Proc. British Machine Vision Conference*, volume 2, pages 609–618, 1998.

[79] S. Niyogi and E. Adelson. Analyzing and recognizing walking figures in xyt. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 469–474, 1994.

[80] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 130 –136, 1997.

[81] P. Van Otterloo. *A Contour-Oriented Approach to Shape Analysis*. Prentice Hall, 1991.

[82] T. Phillips and A. Rosenfeld. A method of curve partitioning using arc-chord distance. *Pattern Recogntion Letters*, 5:285–288, 1987.

[83] R. Polana and R. Nelson. Low level recognition of human motion (or how to get your man without finding his body parts). In *Proc. IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 77–82, 1994.

[84] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[85] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[86] Y. Raja, S. McKenna, and S. Gong. Colour model selection and adaptation in dynamic scenes. In *European Conference on Computer Vision*, pages 460–474, 1998.

[87] G. Rigoll, S. Eickeler, and S. Muller. Person tracking in real-world scenarios using statistical methods. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 342–347, 2000.

[88] J. Rittscher and A. Blake. Classification of human body motion. In *Proc. International Conference on Computer Vision*, pages 634–639, 1999.

[89] S. Romdhani, S. Gong, and A. Psarrou. A multi-view nonlinear active shape model using kernel pca. In *Proc. British Machine Vision Conference*, pages 483–492, 1999.

[90] D. Ron, S. Singer, and N. Tishby. The power of amnesia. *Advances in Neural Information Processing Systems*, 6:176–639, 1994.

[91] R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *Proc. IEEE Workshop on the Interpretation of Visual Motion*, pages 31–38, 1998.

[92] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, 1994.

[93] S. Sclaroff and A. Pentland. Closed-form solutions for physically-based shape modeling and recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 238–243, 1991.

[94] D. Sergeant, R. Boyle, and J. Forbes. Computer visual tracking of poultry. *Computers and Electronics in Agriculture*, 21(1):1–18, 1998.

[95] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proc. European Conference on Computer Vision*, pages 702–718, 2000.

[96] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[97] L. Sirovich and M. Kirby. A low-dimensional procedure for the characterisation of human faces. *Journal of the Optical Society of America*, 4(3):519–524, 1987.

[98] P. Sozou, T. Cootes, C. Taylor, and E. Di Mauro. A non-linear generalisation of pdms using polynomial regression. In *Proc. British Machine Vision Conference*, pages 397–406, 1994.

[99] P. Sozou, T. Cootes, C. Taylor, and E. Di Mauro. Non-linear point distribution modelling using a multi-layer perceptron. In *Proc. British Machine Vision Conference*, pages 107–116, 1994.

[100] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proc. International Symposium on Computer Vision*, pages 265–270, 1995.

[101] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[102] N. Sumpter, R. Boyle, and R.Tillett. Modelling collective animal behaviour using extended point-distribution models. In *Proc. British Machine Vision Conference*, pages 242–251, 1997.

[103] N. Sumpter and A. Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. In *Proc. British Machine Vision Conference*, pages 649–658, 1998.

[104] D. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:831–836, 1996.

[105] D. Terzopoulos and R. Szeliski. Tracking with Kalman snakes. *Active Vision*, pages 3–20, 1992.

[106] D. Terzopoulosi and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superrquadratics. In *Proc. International Conference on Computer Vision*, pages 606–615, 1990.

[107] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society B*, 12(3):611–622, 1999.

[108] I. Tomek. Two algorithms for piecewise-linear continuous approximation of functions of one variable. *IEEE Transaction on Computers*, pages 445–448, April 1974.

[109] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[110] B. Ulmer. Vita-an autonomous road vehicle (arv) for collision avoidance in traffic. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 36–41, 1990.

[111] J. Vaisey and A. Gersho. Simulated annealing and codebook design. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1176–1179, 1988.

[112] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:250–269, 1967.

[113] K. Wall and P. Danielsson. A fast sequential method for polygonal approximation of digitised curves. *Computer Vision, Graphics and Image Processing*, 28:220–227, April 1984.

[114] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[115] A. Wilson and A. Bobick. A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1325–1337, 1997.

[116] H. Worrall, G. Sullivan, and K. Baker. Advances in model-based trafic vision. In *Proc. British Machine Vision Conference*, pages 559–568, 1993.

[117] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[118] C. Wren and A. Pentland. Understanding purposeful human motion. In *Proc. IEEE International Workshop on Modelling People (MPEOPLE)*, pages 19–25, 1999.

[119] Y. Wu and K. Toyama. Wide-range, person and illumination-insensitive head orientation estimation. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 183–188, 2000.

[120] H. Xin and J. Shao. Application of machine vision to swine environmental control. In *Proc. IEEE International Conference on Advanced Intelligent Mechatronics*, page 14, 1997.

[121] M. Yang, N. Ahuja, and D. Kriegman. Face detection using mixtures of linear subspaces. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 70–76, 2000.

[122] A. Yuille and P. Hallinan. *Active Vision*, chapter 2 (Deformable Templates), pages 21–38. MIT Press, 1992.