Exploration of Point Distribution Models in Machine Vision

Catherine Freeburn
HP Laboratories Bristol
HPL-2000-14
31st January, 2000*

machine vision, automatic content classification, statistical model building, point distribution models

The thesis explores the use of statistical point distribution models in machine vision. The effectiveness of traditional machine vision three-dimensional model building techniques has been limited, particularly when the objects being modelled are complex, smoothly deforming bodies such as human beings. Recently, a new approach to model building has been found that allows effective two-dimensional image space models of complex non-rigid objects to be automatically generated by statistically analysing a set of training images. One such technique creates what is known as a point distribution model.

In this thesis I explore whether point distribution models still perform well in areas where the objects being modelled are much less complicated than have hitherto been tried. This was achieved by creating a rational reconstruction of an image interpretation system that automatically generated and used a point distribution model to recognize human beings, but to then use the resulting system to build (and use) models of much more simple rigid objects: specifically, convex polygonal prisms. The system that was reconstructed was a pedestrian tracking system designed and implemented by Baumberg, as part of a Leeds/Reading university collaboration project.

It was found that for simple convex prismatic objects Baumberg's mechanism for automatically extracting point descriptions of object silhouettes (which are then be used to train a model) fails to produce outline descriptions in which the positions of the points move smoothly as the silhouette of the objects change. This problem meant that the resulting models were unstable and in the experiments done could only correctly classify objects just under 50% of the time.

In addition, early indications were found that suggest that even if a more stable outline extraction method can be found, there is unlikely to be enough variation in the outlines of simple objects to enable this kind of model to be able to systematically distinguish between them.

* Internal Accession Date Only

Acknowledgements

First I would like to extend warm thanks to Dave Cliff, Simon Lewis and Martin Merry all of who provided unwavering encouragement and several beers.

I am also, as ever, grateful to Rob for his sympathy and seemingly infinite supply of doughnuts.

Finally I would like to thank Hewlett Packard for providing me with the funding and the time to be able to undertake this M.Sc.

Table of Contents

1 Introduction....3

1.1 Overview of the Thesis:....4

2 Background....6

3 Related Work....12

3.1 Building an Object model....13

3.1.1 Image Capture....14

3.1.2 Differencer....15

3.1.3 Outline Extraction.....16

3.1.4 Re-pointing.....17

3.1.5 Normalisation....19

3.1.6 Algorithm Outline....23

3.1.7 Principal Component Analysis....25

3.2 Object Tracking....30

3.3 Summary.....33

4 Implementation....35

4.1 General Comments....36

4.2 Input Data....36

4.3 Building the Object Model....38

4.3.1 The Training Data.....38

4.3.2 Re-Pointing.....40

4.3.3 Normalisation Algorithm....41

4.3.4 Principal Component Analysis.....43

4.4 Object Tracker.....45

4.4.1 Normalisation.....45

4.4.2 Object Identification.....47

4.5 Summary....48

5 Results: Preliminary Exploration....49

6 Discussion and Further Exploration....56

6.1 Discussion of Initial Results....56

6.2 Further Results 59
6.3 Further Discussion 61
7 Further Work 65
8 Conclusions 67
References in order of citation 69
Bibliography of unquoted references. 70

1 Introduction

The ultimate goal of machine vision for a number of years has been to understand and appropriately interpret images. This involves not only being able to discover the structure of the image, but also to know what that image represents. Once this problem is unlocked a huge number of application areas open out involving image classification, such as allowing content based image storage and retrieval, and face recognition. In addition, higher level interpretation becomes possible if moving images are considered. For instance, once it is possible to identify an object within a single image it is possible to track that object through an image sequence and interpret that object's behaviour.

One major difficulty with image interpretation is the need for a system to have *a priori* knowledge of things that it will be "looking" at. Without this prior knowledge a system could be likened to a new born child looking at the world for the first time and lacking the experience necessary to interpret what it is seeing.

To provide the requisite knowledge for an image interpretation system it is necessary to use models that describe the expected structure of the real world. Until recently the effectiveness of model building has been limited to reasonably simple images of man made objects. Even then these models are only practical to use in very controlled conditions such as a factory floor. However, recently new techniques have been discovered that allow models of complex non-rigid objects (like people) to be automatically generated.

The new techniques differ from traditional model building, as the new techniques do not involve absolute descriptions of objects in three dimensions. Instead the models are built up by statistically analysing changes in the two-dimensional projections of three-dimensional objects as they appear in the image plane. This means that these techniques are particularly suited to the analysis of image sequences as the models are generated by, and describe, two-dimensional object representations, which is precisely what images are.

At the core of these new approaches to machine vision lies the multivariate statistical technique of Principal Component Analysis (PCA). It is the PCA that is responsible for the analysis of the objects in the image sequences. One particularly nice feature of this analysis technique is that it lends itself to approximation by certain neural networks. This potentially allows image interpretation systems to be implemented as massively parallel architectures, which should be very fast and robust. In addition, the fact that PCA can be implemented by a neural network lends weight to the argument that the human visual system interprets data in a similar manner.

It is the exploration of these new techniques that formed the core of the work undertaken in this project. To date these techniques have met with great success in areas that the more traditional model-based systems failed in; i.e. areas where the objects being modelled are smoothly deforming, non-rigid, and (relatively) complex objects like people or faces. But as yet it would appear that nobody has ascertained how well model construction using PCA performs when used in situations where the objects of interest are "simple" rigid objects. The goal of this project was to do precisely that and see if the new techniques are still valid in areas were the objects being modelled are much less complicated. If the answer to this question is "yes", then it becomes possible to implement powerful image interpretation systems designed to identify any kind of object using a single, unified system architecture.

1.1 Overview of the Thesis:

- Chapter 2 presents a brief history of relevant work in machine vision, leading up to the development of point distribution models as studied in this thesis.
- Chapter 3 describes in detail the work presented in Baumberg's PhD thesis, the core of which is rationally reconstructed in this thesis. This includes my own critique of Gower's paper introducing generalised procrustes analysis, a statistical technique used by one component of Baumberg's system.
- In Chapter 4 full details of the implementation of my reconstruction of Baumberg's work are presented.

- Chapter 5 shows results from initial exploration where, surprisingly, Baumberg's system fails to build effective models of simple prismatic objects.
- In Chapter 6 the failures revealed in Chapter 5 are discussed and a hypothesis is formed to account for the cause of these failures. Results from further experiments are present which support this hypothesis.
- Chapter 7 presents a discussion of several promising avenues of further research building on the work presented here.
- Finally, Chapter 8 concludes by summarising the aims, methods and achievements of this thesis.

2 Background

For a number of years the “Holy Grail” of machine vision has been to automatically understand and appropriately interpret images. This does not simply involve being able to analyse the structure of an image, it also involves being able to understand what that image represents. Being able to achieve such automated image interpretation potentially provides solutions to a whole host of image classification problems, ranging from face recognition to automatic diagnosis from medical images. In addition, there is a school of thought that believes that any progress made in this area has the potential to shed further light on how the human visual system functions.

For pragmatic reasons, early work in this area concentrated on the analysis of individual still images, but more recently the arena of image interpretation has expanded to include moving image sequences. The desire here is to recognise not only objects but also object behaviours over time. Having expanded the size of the problem area, the number of applications made possible by finding solutions also increases. For instance natural gesture interpretation allows handwriting or facial expression monitoring to become feasible.

It has long been understood that to be able to perform image interpretation tasks a system must have some prior knowledge about what it is “looking at”. Without prior information which describes and labels the expected structure of the real world any interpretation system would simply be like a new born baby looking at the world for the first time, and lacking the experience necessary to understand any of it.

The approach to the problem of providing an image interpretation system with the requisite knowledge has been to wrap the necessary information up as a model. This idea of constructing a model (otherwise known as model-based vision) has strong biological foundations. The human visual system is incredibly powerful and is capable of interpreting a huge range of often-subtle information present in visual data. For example when looking at a person we are able to determine information about the age or gender of the person from the gait and posture of their walk or we can often tell what mood that person is in from their facial expression. All of this kind of interpretation is possible because we have

learnt characteristic behaviours of people and have therefore built up a mental model that we are able to use when analysing visual data.

Until recently the success of such model-based visual systems has been limited, tending to achieve real world applications only when the objects being modelled are simple man-made bodies, and when the situation of operation is severely constrained (such as on a factory floor). These limitations are probably mostly due to the fact that model-based vision has been dominated by the use of "hand-crafted" three-dimensional models.

This kind of model building is prone to inaccuracies for two reasons. Firstly, when interpreting input data it is necessary to construct representations of three-dimensional objects from two-dimensional image input. This is a non-trivial, under-specified, and often computationally expensive problem, as the complete set of features that are incorporated into such an object model is rarely present in an image. For example, a picture of a car taken from the front will probably not explicitly show the position of the back wheels.

The second problem with this type model building is the fact that often, for practical reasons, there is no alternative but to construct the model by hand. This inevitably leads to inaccuracies, because to build such a model simplifications need to be made which often fail to describe the natural variations in real world objects and object behaviours. For example, when constructing a three-dimensional model of walking human beings it would be very difficult to incorporate variations in object shape due to clothing.

Recently a new development in model building has lead to a number of systems that have been successful in image interpretation tasks involving complex non-rigid bodies in very uncontrolled environments. These systems have not been based on the assumption that three-dimensional models are necessary to perform object recognition. Instead these methods employ multivariate statistical analysis of objects *in the two-dimensional space of images*, and it is possible to build models automatically without simplifying the nature of the objects, by simply statistically analysing training images.

At the core of these new techniques lie statistical analysis methods such as principal component analysis (PCA). PCA can be used to model objects in a number of ways, which differ depending on which features of the images the analysis is applied to. For example, PCA can be used to statistically analyse the differences that occur in the grey-scale levels of images of a certain object, and produce a model describing the grey-scale levels that a particular object is likely to have. Alternatively, PCA can be used to analyse the variability of a set of two-dimensional points, each of which is placed at a particular feature of the object as it appears in the image plane. For example, when building a model of human hands the points might be placed at the fingertips and at the troughs between the fingers. Models based on the statistical analysis of such landmark points are more commonly known as point distribution models.

Figure 2.1 *Examples of the point descriptions of training shapes used in order to construct a model of pedestrians. (Reproduced from Baumberg **[1]**).*

It is also possible to construct a point distribution model based solely on the outlines of an object. For example, a model of human beings could be constructed by analysing the point descriptions of human silhouettes as they appear in different poses, such as is shown in Figure: 2.1. In these cases, when PCA is applied to the set of point descriptions of possible object outlines, the outcome is a point description of the average object outline along with the primary modes of variation that distinguish that average shape from the original object outlines.

It is these results that form the basis of the object model. As the model is inherently two-dimensional this kind of technique avoids the difficulty of having to extract three-dimensional information from an image and hence avoids some of the difficulties involved with more traditional techniques.

Also, as the model can, in principle, be built by simply providing appropriate training images, a model can be tuned to a number of scenarios without having to re-engineer the whole system. This feature, enticingly, opens up the possibility that a particular task (such as object tracking) could be achieved with a unified architecture regardless of the nature of the objects being tracked. (Baumberg **[1]**, Johnson**[2]**) For example, theoretically the same system could be used to track pedestrians through a shopping mall as could be used to track the movement of mice in a laboratory. The only difference would be the image data used to generate the model.

Another particularly nice feature of this analysis technique is that PCA lends itself to approximation by certain neural networks (Haykin **[2]**). This potentially allows image interpretation systems to be implemented as massively parallel architectures, which should be very fast and robust.

Interestingly, experiments have been done, such as the moving light displays of Johansson **[4]**, that show that the human visual system is very capable of interpreting objects when presented only with minimal representations of those objects. Specifically, Johansson's **[4]** work showed that humans could distinguish between people walking and running when only presented with points marking the major body joints. Alternatively, Baumberg **[1]** suggested that only silhouettes of pedestrians are needed for the human visual system to be able to correctly identify those objects as pedestrians. These experiments, coupled with the ease with which neural networks can approximate statistical analysis techniques such as PCA, has added fuel to the debate that the human visual system interprets data in a similar manner, thus circumventing the complexity of generating and manipulating internal representations of three-dimensional objects (Cliff & Noble **[5]**, Baumberg **[1]**).

Even though model building using statistical analysis is a reasonably recent phenomenon, already several applications have been developed that successfully apply this technique to accomplish automated image interpretation. Initially one of the focuses for this work was the medical domain, and

appears to have been dominated by the work of Cootes et al **[6]**,**[7]**. Examples of the work achieved by these people include the interpretation of echocardiograms, images of the spine, and radiographs of hip replacements. Unfortunately the model building phase of these systems was not fully automated and required point descriptions of the significant contours of an image to be extracted by hand.

Since the early days several advances have been made. In the first place several methods have been developed that allow the significant contours of an image to be extracted automatically. For example Baumberg's work on pedestrian tracking **[1]** or Robinson's [8] investigation of differential geometry (as referenced by Taylor et al **[6]**).

In addition, the application space in which successful image interpretation systems have been made has opened out to include things like automated pedestrian tracking systems (Baumberg **[1]**), and facial recognition (Lanitis et al **[9]**). The success obtained in these areas has allowed such systems to be developed further to include identification not only of objects as they appear in individual images but also of object behaviour over time. An example of such work is Johnson's **[2]** extension to Baumberg's pedestrian tracker that allowed atypical (possibly criminal) pedestrian activity to be flagged.

Most recently, Reading and Leeds universities have been involved in a collaborative project to develop an integrated traffic and pedestrian vision system **[10]**,**[11]**. This system was designed, not only to track pedestrians and cars through an outdoor scene, but also to produce natural language descriptions of the interactions between those pedestrians and vehicles. In this work the pedestrian tracker was an implementation of the above-mentioned work done by Baumberg **[1]**, and the behavioural analysis work of Johnson **[2]** was extended by the Reading team, to include Bayesian belief nets to produce natural language descriptions of object interactions. However, the vehicle tracker was implemented using traditional geometric three-dimensional models. This project is an impressive piece of work, not least because it integrates two completely different modelling techniques into a single system. However, it does beg the question; "why was an integrated technique not chosen?"

It is reasonably clear from the literature that although developing a traditional three-dimensional model to track complex objects like pedestrians is achievable (Hogg, as referenced by Cliff & Noble **[5]**), it is

very difficult and can only be applied to real world situations with limited success. However, it is not clear that the new statistically formed model cannot be applied to simple, man-made, rigid bodies.

It would appear that nobody to date has attempted to investigate how well model construction using PCA performs when applied to “simple” rigid objects. This could well be because there is simply not enough variation in the silhouette of such an object for a statistical analysis of that variation to yield a working model. However, this is by no means a foregone conclusion. Hence the goal of this project is to empirically ascertain whether or not these statistical techniques are still valid in areas where the objects being modelled are much less complicated than have hitherto been tried. If the answer to this question is “yes”, then it becomes possible to implement powerful image interpretation systems designed to identify *any* kind of object using a single, unified system architecture. In addition, success here will further support the claim that the human visual system performs a similar form of statistical analysis in visual data interpretation tasks.

The approach taken in this project, to achieve the above goal, was to implement a rational reconstruction of an interpretation system that generated and used a point distribution model to recognise complex bodies, but then to use this system to build (and use) models of much more simple objects such as cars. The system chosen as the basis of this work was the pedestrian tracker that Baumberg implemented as part of the Leads/Reading collaboration. This system was chosen because it was a tried and tested vision system that functioned very well in the real world, in addition this piece of work was undertaken as a PhD project and hence (theoretically) the documentation was rich enough to allow an accurate reconstruction to be achievable.

3 Related Work

As mentioned before, the goal of this piece of work is to determine if point distribution models are still valid in areas where the objects being modelled are simple rigid bodies like cars. The way chosen to achieve this aim was to rationally reconstruct a tried and tested vision system that was initially developed to use these techniques to model more complex bodies like people.

The remainder of this chapter is a description of the pedestrian tracking system implemented by Baumberg **[1]** as this was the system that was chosen to be reconstructed and therefore is the system that forms the basis of this work.

The purpose of Baumberg's research was to address the problem of automatically tracking a number of pedestrians in an outdoor scene. The system specifications were to extract the position of moving pedestrians in a scene filmed by a static surveillance camera and follow each person throughout the captured video sequence. The system needed to be robust against noise and the tracking (although not the initial acquisition of the model) needed to happen in real time (i.e. the tracking system needed to have a frame processing rate equal to or less than normal video frame rates). Such a system could become part of several applications, such as animation generation or human motion analysis, however the key-motivating factor for this work was an automated surveillance system. Indeed, the success of Baumberg's work enabled the resulting system to become part of the Leeds/Reading joint traffic and pedestrian vision system **[10] [11]**.

The system that Baumberg implemented has two main parts. The first of these is the portion of the system that is responsible for building the model, and the second is the part that employed the resulting model to identify and track pedestrians through image sequences. It is the model building process that first will be described in depth.

3.1 Building an Object model

Baumberg's pedestrian model was automatically constructed by statistically analysing the two-dimensional outlines of people taken from several sequences of real video footage of a street scene. The system, as represented in Figure 3.1, took live images from a static camera, processed this data, and extracted point descriptions of the outlines of any moving objects within the scene, in real time. This data was then analysed off-line to generate the model.

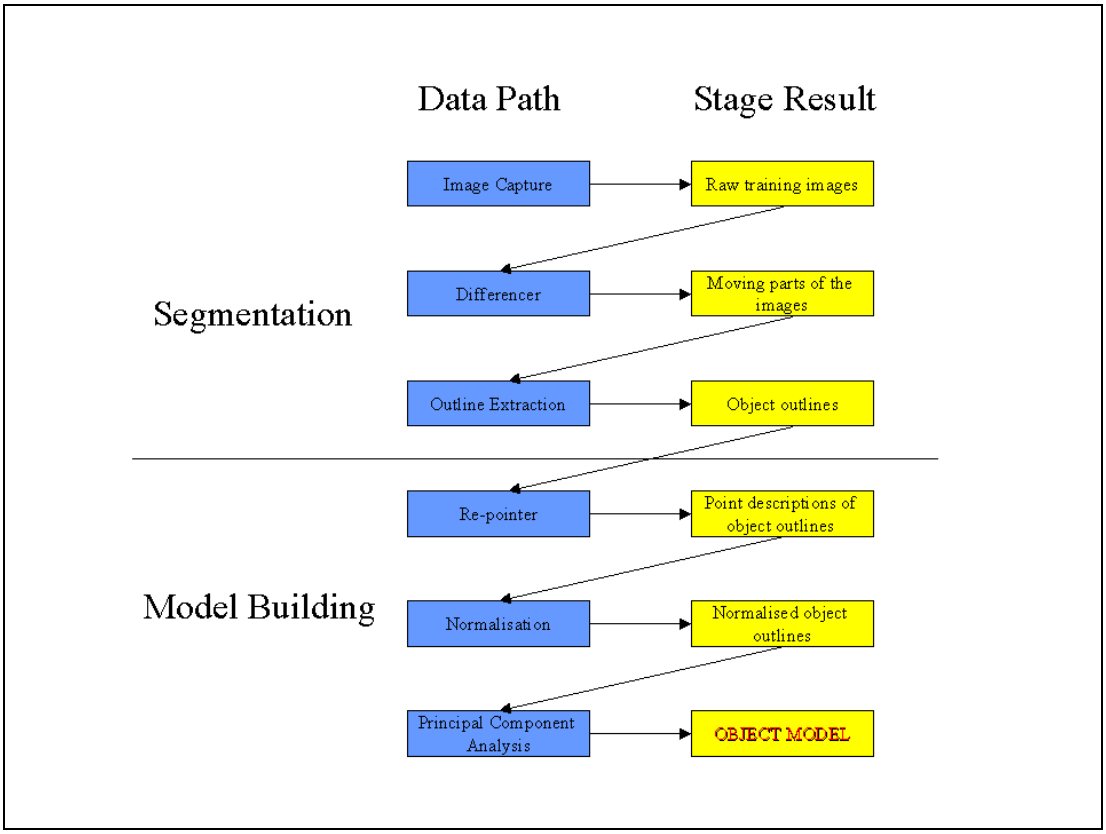

Figure 3.1: *A diagram representing the six stages involved in model construction.*

As can be seen in Figure 3.1, there are six main stages to this process: -

- *Image capture* to obtain the raw images to be used to train the model.
- *Differencer* to obtain the moving parts of the image that correspond to pedestrians.
- *Outline extraction* to obtain an outline description of each pedestrian shape.

- *Re-Pointer* to obtain a point description of each pedestrian shape outline.
- *Off-line normalisation* to align the training shapes so that differences in scale, translation and rotation are not incorporated in the model.
- *Off-line Principal Component Analysis* to build the shape model.

3.1.1 Image Capture

To some extent the choice of what training data is to be used is dependent on the final application. In this case the final system was to analyse images taken with a static camera of a street scene. See Figure 3.2 for an example of that scene. So the input used to train the model was simply a sequence of images taken with that same camera of that same street scene. This means that physical characteristics of the system, such as position of the camera with respect to the ground plane, are implicitly incorporated into the model. It also means that the model developed does not contain the full set of possible "views" of human beings. These points might be considered as a significant drawback; however, in real applications, the possible views of objects are rarely completely unconstrained. For instance it is unlikely that in a surveillance system that the camera will be looking directly up at a person's feet. However, if this were that case then the system could still be trained using those images. In addition, this concept of using training data comprised of images that would typically be those that the system would eventually need to analyse, means that the system does not need to be re-engineered for every different location it is used in. Instead, a model specific for each location can be simply and automatically trained.

Figure 3.2: *An example of a typical street scene as viewed by the surveillance camera that provided training and input images for Baumberg's pedestrian tracker. (Reproduced from Baumberg **[1]**).*

3.1.2 Differencer

As the camera in Baumberg's system was fixed, a simple background subtraction scheme was used to obtain the moving parts of the training images. Background subtraction relies on the existence of a static reference image of the scene's "background". This was obtained from a sequence of images median filtered over time. This process of using a median filter can then be applied periodically using input data from the camera to account for variable effects such as changing light conditions, or camera shake.

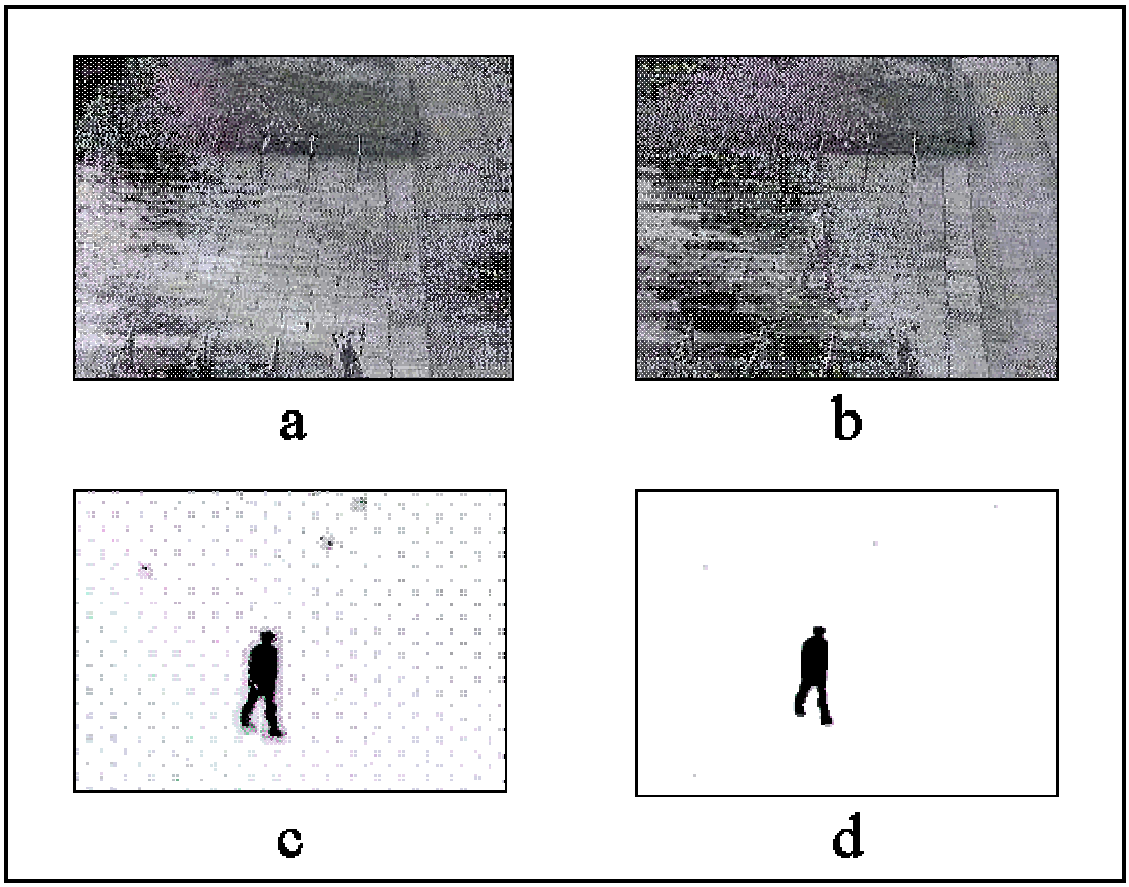

Figure 3.3: *An illustration of Baumberg's image differencing stage. (a) Background image, (b) Training image, (c) Differenced image, (d) Blurred and thresholded image. (Reproduced from Baumberg **[1]**).*

Subtracting the background image from each frame then identifies the moving objects within that frame. Working with grey-scale images, this subtraction is achieved by subtracting the image intensities at each pixel. To reduce the effects of noise in the images this differenced image is then blurred using a standard Gaussian blur filter. Finally the resulting blurred differenced image is

thresholded to obtain a binary image where pixels corresponding to a significant change in intensity (i.e. a moving object) are all coloured black, whereas the rest of the image is set to white. (Figure 3.3).

Where there is poor contrast between the moving object and the background, fragmentation can occur, causing several small moving objects to be identified where there should be only one. Morphological filters are applied to these gaps, where the foreground regions are successively grown and the background regions successively shrunk until the fragmentation has been removed.

3.1.3 Outline Extraction.

Baumberg's differencer, as described above, generates a binary image in which every pixel that is part of a moving object is set to black, hence each black 'blob' is a potential silhouette of a single moving object in the scene. As the model building process in the system is entirely automatic there is no guarantee that each identified moving object is actually a pedestrian and therefore to be included in the model. So some specific constraints need to be used to reject regions which are unlikely to be a single pedestrian. Baumberg used bounding box characteristics to achieve this, and discounted any differenced blob that did not fall within a maximum/minimum size range, or that did not have an appropriate height to width ratio.

Once the 'blobs' have been filtered in this way, the outline of the 'blob' is identified by employing a straight forward edge-detection algorithm; made easier by the fact that the 'blobs' are in binary images. The output of the edge detector is a set of candidate edge-pixels; possibly disconnected and/or discontinuous, and probably affected by noise. Baumberg the fitted a continuos set of spline curves to give a smooth approximation to the raw outline. The control points for these spline curves represented a set of points whose distribution varied between images. It was then the task of principal component analysis to capture that variation.

3.1.4 Re-pointing.

There seem to be few prescriptive details available in the literature concerning what properties the point descriptions of training shape outlines need to have for principal component analysis of those shapes to be possible. However at least the following three constraints appear as implicit assumptions.

1. The number of points used to describe each shape needs to be the same.
2. For two similar shapes the *i*'th point on each shape should idealy to correspond to the same feature in that shape. For example, when looking at two silhouettes of humans from the front, if the first point of the first silhouette is the top of the head, then the first point of the second silhouette also needs to correspond to the top of the head. If this is not the case then the model building process will incorporate variation in the positioning of points into the model rather than simply the variation of the shape itself. In cases where the shapes are not similar this constraint does not apply.
3. Enough points to adequately describe the shape need to be used; i.e. more points need to be used to describe more complex shapes.

To achieve this Baumberg first needed a method to find a fixed boundary point that would appear at the same position on all contours that would be consistent and robust in the face of image noise. Once a reference point was found for each contour then any consistent method for placing the rest of the points around the outline would mean that the first two conditions would be met.

Baumberg's method for obtaining the reference point was to find the principal axis of the object and select the uppermost of the two points where the axis crossed the boundary of the shape. The assumption being made here was that this point would be fixed for humans as they appear in the input data. This was a reasonable assumption to make given that in the particular situation for which the system was to be used people would always appear in an upright position.

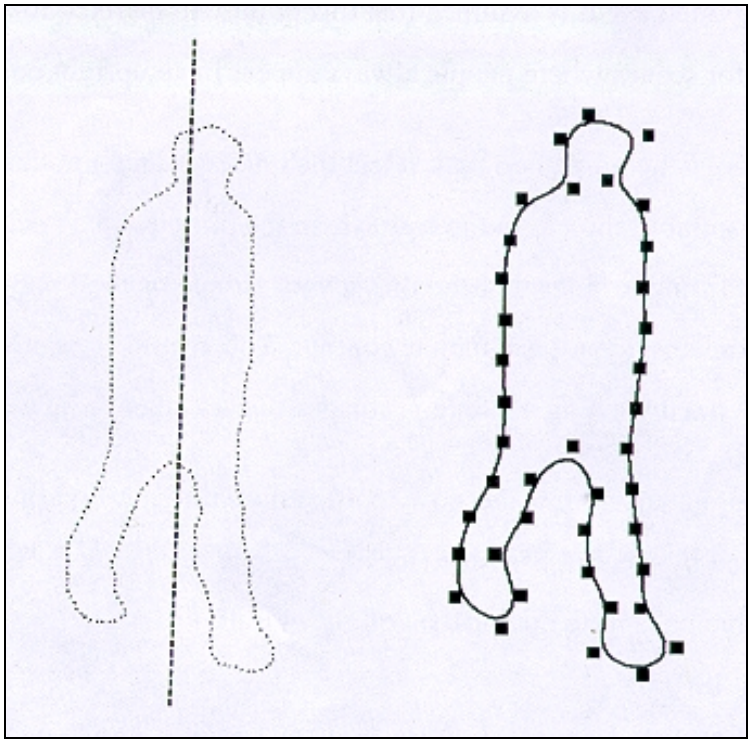

Figure 3.4: *A diagram representing Baumberg's method for extracting a point description of a training shape. (Reproduced from Baumberg* ***[1]****).*

The rest of the points were determined by approximating the shape outline as a length-wise uniformly spaced B-spline and using its control points in combination with the reference point as the outline description.

To meet the final criteria mentioned above Baumberg, seemingly arbitrarily, chose that each shape should be described using 40 control points, presented as a single column shape-vector of the form:

$$\mathbf{x} = \left(x_0, y_0, x_1, y_1, \ldots x_{n-1}, y_{n-1}\right)^T \qquad \textbf{(3.1)}$$

where $\left(x_i, y_i\right)$ is the position of the i^{th} point on the training shape outline, n is the number of points used to describe the outline and T represents vector transpose.

3.1.5 Normalisation

Before principal component analysis (PCA) can be applied to the point descriptions of the training shapes, these shapes need to be aligned (i.e. rotated, scaled and transformed to a common origin), otherwise the PCA will unnecessarily incorporate variation in size and location into the model. For example, if two shapes that are very similar appear at different positions in the image frame then the variation due to the different locations will actually swamp out any variation of shape.

To achieve normalisation Baumberg implemented a standard technique known as generalised procrustes analysis, as derived by Gower **[13]**.

When applied to a set of shapes, the goal of generalised procrustes analysis is to translate, rotate and scale those shapes such that the total of the differences between each shape in the set and the resulting average shape is minimised. In other words after normalisation, there is a set of shapes such that the total of the distances between each point of each shape and the corresponding point of that set's mean shape is minimised. In analysis of variance terms, this is equivalent to minimising the residual sum-of squares.

Specifically if there is a set of *m* shapes described by matrices of the form: -

$$\mathbf{S}_i = \begin{bmatrix} x_0, & y_0 \\ x_1, & y_1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{n-1}, & y_{n-1} \end{bmatrix} \qquad \textbf{(3.2)}$$

where *n* is the number of points used to describe each shape, and each row (x_k, y_k) is the x, y co-ordinates of the i^{th} point used to describe the shape. Then generalised procrustes analysis will produce a set of *m* shapes $\mathbf{Z}_k$, whose average is $\mathbf{M}_z$, that minimise the equation for the residual sum-of-squares given by:

$$\mathbf{RSS} = \sum_{i=0}^{m-1} tr\left(\left(\mathbf{Z}_i \mathbf{Z}_i^T\right) - \left(\mathbf{M}_z \mathbf{M}_z^T\right)\right) \quad \textbf{(3.3)}$$

The transformation that each shape undergoes to achieve this normalisation involves translation, rotation and scaling. The rotation can be achieved by post multiplying the original shape, $\mathbf{S}_i$, with an orthogonal matrix $\mathbf{H}_i$, and uniform scaling is expressed as a multiplicative constant ρ_i. Translation to a new origin involves adding the same (1 x 2) row-vector, t_i, to every row of the original shape, $\mathbf{S}_i$. Expressing the (n x 2) translation matrix, whose rows are all set to be t_i, as $\mathbf{T}_i$ the transformation that each shape undergoes can algebraically be expressed as: -

$$\mathbf{Z}_i = \rho_i \mathbf{S}_i \mathbf{H}_i + \mathbf{T}_i \quad \textbf{(3.4)}$$

where $\mathbf{Z}_i$, is the resulting normalised shape.

Now, it should be reasonably clear, that one way to minimise the sum of differences that each shape has with the mean shape is to simply set each scaling factor, ρ_i, to be zero, and effectively reduce each shape to a single point. This is a trivial solution and to have a more satisfactory method for calculating the scaling factors one actually needs to minimise Equation 3.3 subject to some constraint. The constraint chosen by Gower is given by: -

$$\sum_{i=0}^{m-1} \rho_i^{\,2} tr\left(\mathbf{S}_i \mathbf{S}_i^T\right) = \sum_{i=0}^{m-1} tr\left(\mathbf{S}_i \mathbf{S}_i^T\right) \quad \textbf{(3.5)}$$

Calculation of the translation matrices, $\mathbf{T}_i$, is relatively simple, as all that needs to be done to minimise Equation 3.3 is to translate all of the input shapes to have the same centroid. This centroid can actually be anywhere without breaking any constraints and is therefore, for convenience, chosen to be the system's origin. In geometrical terms, this is achieved by calculating the co-ordinates of the shape's centroid and subtracting these from each of the points describing the shape.

Once the original shapes have been translated, they need to be rotated and scaled. The problem of simply rotating a matrix, $\mathbf{X}$, to fit another matrix that has the same centroid, $\mathbf{Y}$, is a well-understood problem. A rotation matrix, $\mathbf{H}$, such that $\mathbf{XH}$ best fits $\mathbf{Y}$ is given by: -

$$\mathbf{H} = \mathbf{V}^T\mathbf{U} \tag{3.6}$$

Where $\mathbf{U}$ and $\mathbf{V}$ are calculated by obtaining the singular value (or Eckart-Young) decomposition of the matrix $\mathbf{Y}^T\mathbf{X}$ as described by: -

$$\mathbf{Y}^T\mathbf{X} = \mathbf{U}^T\Gamma\mathbf{V} \tag{3.7}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and Γ is diagonal.

However, where it is necessary to scale matrices as well as rotate them, the above method can not be applied directly. This is because if a matrix is first rotated to fit another using the above method and then a scaling factor found, these transformations in combination do not actually satisfy the constraint of minimising Equation 3.3 above. In fact the problem of *both* scaling and rotating a matrix $\mathbf{X}$ to fit another matrix $\mathbf{Y}$ does not actually have an analytical solution, therefore the approach needed to solve this kind of problem has to be iterative. In other words the matrix $\mathbf{X}$ needs to be rotated, then scaled. Then the matrix resulting from these operations needs to be re-rotated and scaled until some tolerance has been reached.

In cases where more than two matrices are needed to be aligned the problem is simply a generalisation of the above case, and the way that the various rotation and scaling stages are calculated can be derived by using standard calculus of variation techniques. The mathematics involved in doing this is beyond the scope of this thesis, but a full description can be found in Gower **[13]**. However, a description of the algorithm designed by Gower to achieve normalisation of *m* matrices will now be described.[1]

[1] It should be noted that whilst Gower's description of the mathematical derivation of this algorithm is sound, his description of the algorithm itself is, frankly, awful and contains at least one highly misleading logical error in addition to several typographic errors. Therefore I would strongly

The first task is to translate all of the shapes to the origin, in analysis of variance terms this removes the between-groups sum-of-squares variance leaving the rest of the process to minimise the within-groups sum-of-squares variance. Then, so that the numbers are "friendly" each translated shape is scaled such that: -

$$\sum_{i=0}^{m-1} \lambda^2 tr(\mathbf{S}_i \mathbf{S}_i^T) = m \qquad \textbf{(3.8)}$$

If after normalisation is complete, and the original units for the shapes are important, the normalised shapes should be multiplied by the scaling factor λ.[2]

The next step is an initialisation phase, so that all necessary variables are given appropriate values before entering the iteration process. First of all the second shape in the set, $\mathbf{S}_1$, is rotated to fit the first shape, $\mathbf{S}_0$. And then the matrix, $\mathbf{M}_s$, which is the average of $\mathbf{S}_0$ and the rotated $\mathbf{S}_1$ is calculated. The next shape, $\mathbf{S}_2$, is then rotated to fit the current value of $\mathbf{M}_s$ and $\mathbf{M}_s$ is recomputed to be the average of $\mathbf{S}_0$, $\mathbf{S}_1$ and $\mathbf{S}_2$. Subsequent shapes are similarly rotated to fit the mean, and the mean recomputed. This gives initial values for the normalised shapes ($\mathbf{Z}_i$), their mean ($\mathbf{M}_z$), and an initial value for the residual sum-of-squares ($\mathbf{RSS}$) can be calculated.[3]

Now keeping the mean fixed, each shape can be rotated again to fit this value for the mean. After all *m* shapes have been re-rotated a new mean ($^{Rotated}\mathbf{M}_z$) and a new residual sum-of-squares ($^{Rotated}\mathbf{RSS}$) can be calculated.

---

recommend that, should anybody be wishing to implement this algorithm, they use the description found here.

[2] This initial scaling factor is not strictly necessary, as Gower only included it in his computation to make sure that the numbers being dealt with were large enough for the computers available in 1975 to be able to handle them with any accuracy. I chose to retain this stage, as the only way to verify the operation of the algorithm was to reproduce the example results that Gower provides in his paper.

[3] Again, this stage is not strictly necessary. Gower includes this initial rotation phase in an attempt to make the resulting algorithm symmetrical (i.e. in the case of two matrices, rotating and scaling **X** to fit

Next the scaling factors $\rho_i \ (i = 0,1,...,m-1)$ can be calculated, and the scaled and rotated shapes found by multiplying the previously rotated shapes by the scaling factors. Once this has been done for all *m* shapes a new mean (${}^{Scaled+Rotated}\mathbf{M}_z$) and a new residual sum-of-squares (${}^{Scaled+Rotated}\mathbf{RSS}$) can be calculated.

Now if the difference between the sum of squares for the rotated and scaled shapes (${}^{Scaled+Rotated}\mathbf{RSS}$) and the sum of squares for the initial shapes ($\mathbf{RSS}$) is below some tolerance nothing more needs to be done. However if further reductions to the value of ${}^{Scaled+Rotated}\mathbf{RSS}$ are needed then further rotation needs be done and an improved scaling factor needs to be found. This is achieved by re-rotating the previously rotated (*but not scaled)* shapes to fit the mean of the rotated *and* initially scaled shapes. Then an improved scaling factor can be found that will scale the re-rotated shapes to the new mean of that set

This process is repeated until convergence is achieved for the value of the residual sum-of-squares for the rotated and scaled shapes. As this process is a little difficult to visualise an outline of the computation necessary to achieve this normalisation follows.

3.1.6 Algorithm Outline

1. Initial translation step:
    - Centre each shape, $\mathbf{S}_i \ (i = 0,1,...,m-1)$, on the origin.
2. Initial scaling step:
    - Scale each translated, $\mathbf{S}_i$ by λ such that: -

$$\sum_{i=0}^{m-1} \lambda^2 tr(\mathbf{S}_i \mathbf{S}_i^T) = m \qquad \textbf{(3.9)}$$

Y is equivalent to rotating and scaling **Y** to fit **X**). Again this stage was retained purely because to only way to verify the algorithm was to exactly reproduce Gower's work.

3. Initial rotation step (i.e. calculation of initial candidates for normalised set, $\mathbf{Z}_i$ and the mean of that set, $\mathbf{M}_z$.)

   - Set $\mathbf{Z}_0 = \mathbf{S}_0$
   - Rotate $\mathbf{S}_1$ to fit $\mathbf{Z}_0$, and call the result $\mathbf{Z}_1$
   - Now calculate the mean value, $\mathbf{M}_z$ to be the average of $(\mathbf{Z}_0, \mathbf{Z}_1)$.
   - Rotate $\mathbf{S}_2$ to fit $\mathbf{M}_z$, and call the result $\mathbf{Z}_2$.
   - Re-calculate the mean, $\mathbf{M}_z$, to be the average of $(\mathbf{Z}_0, \mathbf{Z}_1, \mathbf{Z}_2)$
   - Repeat above two steps for $\mathbf{S}_3, \mathbf{S}_4, ..., \mathbf{S}_{m-1}$
   - The results of this stage will be initial values for $\mathbf{Z}_i$ $(i = 0,1,..., m-1)$, and an initial value for $\mathbf{M}_z$ which is the average of $(\mathbf{Z}_0, \mathbf{Z}_1, .., \mathbf{Z}_{m-1})$.
   - Use the results to calculate an initial value for the residual sum-of-squares using :-

$$\mathbf{RSS} = m(1 - tr(\mathbf{M}_z \mathbf{M}_z^T))^{4} \qquad \textbf{(3.10)}$$

   - Finally initialise the scaling factors by setting them all equal to one.

$$\rho_i = 1 \text{ for } (i = 0,1,..., m-1)$$

4. This is the start of the main iterative process, where a rotation step is calculated.

   - For $(i = 0,1,..., m-1)$ rotate $\mathbf{Z}_i$ to fit $\mathbf{M}_z$, and call the result ${}^{Rotated}\mathbf{Z}_i$
   - Calculate a new mean called ${}^{Rotated}\mathbf{M}_z$ that is the average of :-

$$\left({}^{Rotated}\mathbf{Z}_0, {}^{Rotated}\mathbf{Z}_1, ..., {}^{Rotated}\mathbf{Z}_{m-1}\right)$$

   - If desired, as it is not actually used in the calculation, find the residual sum-of-squares of the rotated shapes using:-

$${}^{Rotated}\mathbf{RSS} = \mathbf{RSS} - m * tr\left({}^{Rotated}\mathbf{M}_z \, {}^{Rotated}\mathbf{M}_z^T - \mathbf{M}_z \mathbf{M}_z^T\right) \qquad \textbf{(3.11)}$$

5. Next a scaling step is calculated.

   - For $(i = 0,1,..., m-1)$ evaluate ${}^{new}\rho_i \big/ p_i$, using [5] :-

[4] This equation is a simplified version of Equation 3.3, where the initial scaling step (step 2 in the above description) allows that simplification to be possible.

$$\frac{({}^{new}\rho_i)^2}{(\rho_i)^2} = \frac{tr\left(\rho_i \mathbf{Z}_i \mathbf{M}_z^T\right)}{tr(\rho_i^2 \mathbf{Z}_i \mathbf{Z}_i^T) tr(\mathbf{M}_z \mathbf{M}_z^T)}$$

- Scale the rotated shapes ${}^{Rotated}\mathbf{Z}_i$ to produce scaled and rotated shapes, ${}^{Scaled+Rotated}\mathbf{Z}_i$, by using:-

$${}^{Scaled+Rotated}\mathbf{Z}_i = \frac{{}^{new}\rho_i}{\rho_i} {}^{Rotated}\mathbf{Z}_i$$

- Calculate the new mean, ${}^{Scaled+Rotated}\mathbf{M}_z$.
- Finally calculate the new residual sum-of-squares of the scaled and rotated shapes, using: -

$${}^{Scaled+Rotated}\mathbf{RSS} = \mathbf{RSS} - m * tr\left({}^{Scaled+Rotated}\mathbf{M}_z \, {}^{Scaled+Rotated}\mathbf{M}_z^T - \mathbf{M}_z \mathbf{M}_z^T\right) \quad \textbf{(3.13)}$$

6. Tolerance check:
   - If $\mathbf{RSS} - {}^{Scaled+Rotated}\mathbf{RSS} < tolerance$ then the iteration is complete and the normalised set is ${}^{Scaled+Rotated}\mathbf{Z}_i$.
   - If $\mathbf{RSS} - {}^{Scaled+Rotated}\mathbf{RSS} > tolerance$ then:-
     - Set $\mathbf{Z}_i = \frac{1}{{}^{new}\rho_i} {}^{Scaled+Rotated}\mathbf{Z}_i$
     - Set $\mathbf{M}_z = {}^{Scaled+Rotated}\mathbf{M}_z$
     - Set $\rho_i = {}^{new}\rho_i$
     - Return to step 4.

This ends the description of generalised procrustes analysis.

3.1.7 Principal Component Analysis

The outcome of the normalisation process is a set of aligned training shape-vectors that need to be put in the form: -

[5] The reason for using this equation to calculate the new scaling factors follows from the mathematical analysis in Gower's paper, which is considered beyond the scope of this thesis.

$$\mathbf{x} = \left(x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1}\right)^T \quad \textbf{(3.14)}$$

It is these shape-vectors that are statistically analysed using principal component analysis (PCA) to form the object model.

PCA allows the relationships within a set of correlated variables to be examined by transforming that set of variables into a new set of uncorrelated variables called principal components. These new variables are derived in decreasing order of importance so that the first principal component accounts for as much as possible of the variation of the original data.

Taking the trivial 2-d example, of an original data set such as is shown in Figure 3.5. The result of PCA is to establish a new co-ordinate system whose origin is the centroid of the system (i.e. the mean value of the original data) and whose axes are in the directions that correspond to the most significant modes of variation of the system. As can be seen, the number of principal components obtained is equal to the number of dimensions in the original data set. (In this case that is two.)

The first principal component accounts for the most variation in the data set, and this is reflected by it being associated with the largest eigenvalue in the set. The second principal component accounts for the most variation in the original data that is not described by the first principal component. In addition even if the data had actually been distributed on a straight line, as it is a two dimensional data set two principal components would still be obtained. However, as there would only be variation in one direction the second principal component would be associated with an eigenvalue equal to zero and its direction would be arbitrary other than complying with the orthogonality constraints.

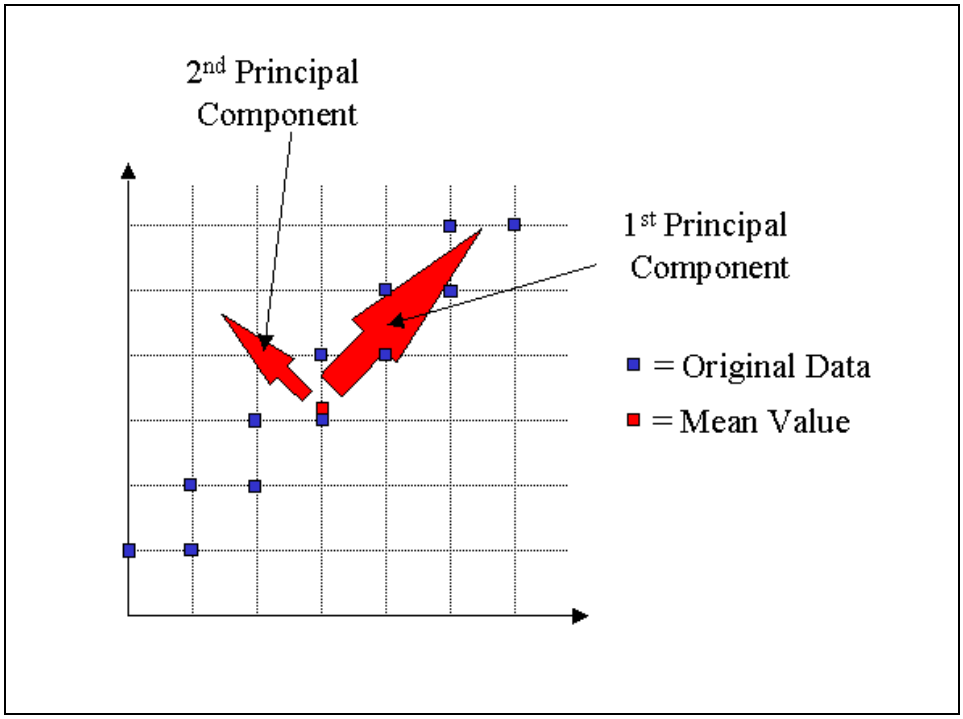

Figure3.5: *A simple illustration of PCA*

In Baumberg's system PCA was not being used to analyse a data set consisting of two-dimensional points. Instead it was applied to the set of normalised shape-vectors extracted from the training images. Given such a set of column vectors, the outcome of PCA is the mean shape of the original set and a number of orthogonal axes that represent the directions in which the initial set of shapes vary from that mean. As will be seen, although the shapes are only two-dimensional the dimensionality of the analysis is actually related to the number of points used to describe the outline of each shape.

The first stage in PCA is to calculate the mean shape vector of the normalised set, using

$$\mathbf{m}_s = E\{\mathbf{s}\} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{s}_i \qquad \textbf{(3.15)}$$

where $E\{\arg\}$ is the expected value of the argument and M is the number of shapes in the set.

Then covariance matrix of the data set is calculated, as defined by.

$$\mathbf{C}_s = E\left\{(\mathbf{s}-\mathbf{m}_s)(\mathbf{s}-\mathbf{m}_s)^T\right\} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{s}_i\mathbf{s}_i^T - \mathbf{m}_s\mathbf{m}_s^T \qquad \textbf{(3.16)}$$

Because each vector $\mathbf{s}_i$ is of order $2n$ the covariance matrix $\mathbf{C}_s$ is of order $2n$ x $2n$. The element c_{ii} of $\mathbf{C}_s$ is the variance of the i^{th} component of the original shape vectors. For example, element c_{00} is a measure of how similar the values of x_0 are across the data set, and element c_{11} is a measure of how similar the values of y_0 are. The element c_{ij} of $\mathbf{C}_s$ is the covariance between the i^{th} and j^{th} components of the original data set. This is a measure of how correlated these different values are. For example, element c_{01} is the covariance of the values for x_0 and y_0 across the original data set. If these components are not correlated then their covariance is zero and, therefore, $c_{ij} = c_{ji} = 0$. The matrix $\mathbf{C}_s$ is both real and symmetric.

Because the covariance matrix, $\mathbf{C}_{s,}$ is both real and symmetric it can be shown (Gonzalez **[14]**) that it is always possible to find a set of $2n$ orthogonal eigenvectors that solve the equation: -

$$\mathbf{C}\mathbf{e}_i = \lambda_i \mathbf{e}_i \qquad \textbf{(3.17)}$$

Where $\mathbf{e}_i$ are the eigenvectors and λ_i are the corresponding eigenvalues arranged in descending order so that $\lambda_0 \geq \lambda_1 \geq \ldots \lambda_{(2n-1)} \geq 0$. It is the eigenvectors that are the new axes of the system and the corresponding eigenvalues that describe how much of the systems total variance lies in the directions denoted by each of these axes.

Now, the eigenvectors are arranged into a *2n* by *2n* matrix, **P** ordered so that the first column of **P** is the eigenvector corresponding to the largest eigenvalue and the last column is the eigenvector corresponding to the smallest eigenvalue.

It is the eigenvector matrix, the eigenvalues and the mean shape-vector that together form the object model.

The eigenvalue matrix **P**, is essentially a transformation matrix that maps any shape-vector, $\mathbf{s}_i$, to a new vector, $\mathbf{b}_i$, in the model's co-ordinate system as follows:

$$\mathbf{b}_i = \mathbf{P}^T \left(\mathbf{s}_i - \mathbf{m}_s \right) \tag{3.18}$$

The vector $\mathbf{b}_i$, is of the form: -

$$\mathbf{b}_i = \left(b_0, b_1, ..., b_{2n-1} \right)^T \tag{3.19}$$

Where the coefficients b_i are known as the shape parameters, and each of these coefficients is a measure of how different the current shape-vector is from the training set's mean value along the direction of the corresponding eigenvector $\mathbf{e}_i$.

To ascertain whether an input shape-vector matches the class of shapes that have been modelled, Equation 3.18 is used to translate the input shape-vector into the model space. The resulting shape parameters are then individually tested to see if they lie within suitable limits. These limits are set by the eigenvalues resulting from the principal component analysis. The variance of the i^{th} shape parameter (b_i) within the training set is simply the eigenvalue λ_i. If the distribution of shape parameters across the training set is assumed to be normal then the two times the standard deviation in any direction will acount for ninety-nine percent of the variation seen in the training shapes along that direction. As the standard deviation of a distribution is simply the square root of the distribution's variance, then the standard deviation for the model's distribution of values for a shape parameter is $\sqrt{\lambda_i}$. Therefore suitable limits for testing to see if an input shape is of the class of shapes that is described by the model, are: -

$$b_i = \pm 2\sqrt{\lambda_i} \tag{3.20}$$

Often it is possible to compress the size of the resulting model by discarding eigenvectors that correspond to insignificant modes of variation. In other words, if any of the eigenvalues are very small (or zero), then there is little (or no) variation in the direction specified by the corresponding eigenvector, and these eigenvectors can be removed from the matrix **P**. Given that the eigenvectors are

arranged in decreasing order of importance, the least significant modes of variation can be removed by simply discarding the last *m* columns of **P.**

The total variation of the original data set can be said to equal: -

$$\sum_{i=0}^{2n-1} \lambda_i \tag{3.21}$$

In other words the total variance of the original data is equal to the sum of the variances in each of the directions denoted by the eigenvectors. Hence it is possible to state that the proportion of the total variance in the original data occurring in the direction of the i^{th} eigenvector is given by: -

$$\lambda_i * \left(\sum_{i=0}^{2n-1} \lambda_i \right)^{-1} \tag{3.22}$$

This gives a convenient method for calculating how many eigenvectors need to be retained for the model to be effective. For example, Baumberg's training data consisted of outlines described by forty two-dimensional control points; hence the outcome of PCA was a matrix containing eighty eigenvectors. However, Baumberg found that the first eighteen eigenvalues accounted for ninety percent of the variation in the training data, hence he compacted his model by discarding the remaining sixty-two insignificant eigenvectors.

3.2 Object Tracking

The aim of Baumberg's system was to track one or more non-rigid objects through a real life outdoor scene. The image data that was to be analysed was simply real-time video footage of a pedestrian scene. The model used to recognise objects of interest within the input data was a two-dimensional statistical model as described in the previous section.

Given the nature of the input data (i.e. real-time video footage of an outdoor scene) there were two main criteria that Baumberg's object tracker needed to meet. First, his tracker needed to be able to analyse the input images at the same rate that they were captured by the video camera. In other words,

thirty images per second needed to be analysed. Second, the tracking mechanism needed to be very robust against noise, not only because the quality of video can be variable but also because objects within the scene could be partly occluded.

Given these criteria there are several reasons as to why it was not enough for Baumberg to implement a tracker that simply extracted the shape-vector descriptions of moving object silhouettes and compared these to the object model.

Looking first at the fact that the live input video stream can be noisy, it may be the case that the segmentation of the entire image could yield silhouettes that are significantly corrupted, and will therefore not match the developed object model. Such a corrupted image is shown in Figure 3.6, where "data drop-out" noise was simulated by randomly adding black or white image artefacts to the result of image differencing. As can be seen from this diagram the existence of such noise can severely alter a shape's outline.

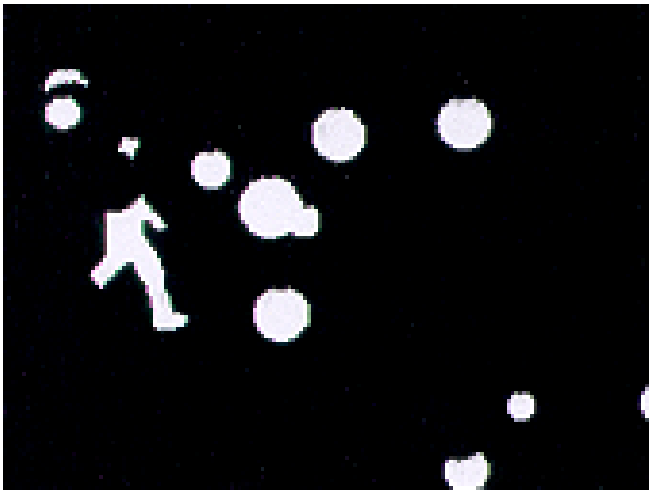

Figure 3.6: *A sample corrupted image that was used as input data for Baumberg's pedestrian tracker*

In addition to containing noise, the input images could be very cluttered and contain many moving objects. In order to be able to track all shapes as they move through a particular image sequence it is necessary to be able to distinguish between the different objects even when they are occluding each other. To achieve this some kind of mechanism needs to be employed to predict where a tracked shape is going to appear in the next image. Applying such a prediction mechanism also reduces the overhead

of searching through the entire image for a particular object, thus helping to keep the computation time down.

Baumberg's predictive tracking mechanism involved the assumption that the objects undergo a uniform two-dimensional motion in image space. Hence the position that a pedestrian is likely to be at in a subsequent frame can be estimated, to within a certain error limit, from its position in the current frame. Similar predictive assumptions can be made about how much the tracked shape was likely to have rotated, changed in size and how the shape parameters are likely to have changed. At this point it was possible to project the mean shape of the model into the image plane at the position, and with the rotation and scaling factors previously estimated using: -

$$\mathbf{S} = Q(s,\vartheta)\left[\mathbf{M}_s + \mathbf{Pb}\right] + \mathbf{T}_c \qquad \textbf{(3.23)}$$

where $Q(s,\vartheta)$ is a rotation by ϑ and a scaling by s and $\mathbf{T}_c$ is a translation by (X_c, Y_c). The shape vector S represents the position of the *n* outline points of the shape in image co-ordinates.

Now an iterative process is entered to refine all of the above estimates. First, suggested movements for each landmark point are calculated. This is done by searching for the strongest edge along the normal to the model boundary at each landmark point. The suggested movements for the landmark points are set to be the distance between the original estimated position and the edge feature. Once new estimates for the values of the state parameters have been found it is possible to go back and improve on the estimates for the translation, rotation and scaling parameters.

The way Baumberg achieved this is in practice was to use the following process. First the shape and alignment (rotation and scaling) parameters were assumed to be fixed and a value for the new origin found. The value for the new origin was calculated by first making an estimate of where the origin was likely to be, subject to some error, and then by searching for edge features in the image to make a "noisy" measurement of were the new actually origin was. These two measurements can then be combined using a Kalman filtering technique (**[15]**,**[16]**) that balances the errors in the estimated value with the errors of the measured value, thus theoretically providing a more accurate result than either of

the two original values. Once a value for the new origin was found this was assumed to be fixed and values for the rotation and scaling factors calculated using a similar mechanism. Finally, again using the same mechanism, each shape parameter could be updated, assuming that the position and alignment factors were constant. This whole process was iterative so the shape contour was actually refined several times for each frame.

This process was applied to each object that the tracker had already identified, so a separate fast segmentation technique was necessary to locate any new moving objects in each frame. This technique was similar to the background subtraction technique used to extract training shapes (as described in Section 3.1.2), except that it did not contain the noise reduction step. Objects that were already being tracked could then be removed from the resulting differenced image by simply clearing the bounding box of the tracked object. Any remaining significantly sized components were then assumed to correspond to new moving objects.

Initial estimates for translation, origin and scaling were calculated from the size and orientation of the bounding box, and initial estimates for the shape parameters were just taken to be the associated eigenvalue, λ_i.

Unfortunately, Baumberg is not explicit about how his tracker distinguishes between moving objects that are pedestrians and moving objects that are not. My assumption on reading his work is that if a moving object does not correspond to the model constructed of pedestrians then there will be no (or few) edge features available in the image within the limits of the feature search. This will result in "lock" being lost and the object not being tracked.

3.3 Summary.

In this long chapter, an overview of the pedestrian tracking system implemented by Baumberg has been presented. Baumberg's system consisted of two main parts. The first of these automatically constructed a point distribution model of pedestrians from a set of training images. This model builder functioned

by first taking live raw images from a static camera and applying a background differencing technique to locate any moving objects within the scenes depicted. Once roughly human-sized moving objects had been located, a reference point was found on each silhouette that roughly corresponded to the top of the pedestrians head. The rest of the points used to describe each outline were calculated by setting them to be the control points of a length-wise uniformly spaced B-spline that described the shape of the outline. The resulting point descriptions of pedestrian outlines were then aligned, to remove any shape variation due to transitional, rotational or scaling differences, by using a normalisation algorithm known as generalised procrustes analysis due to Gower. Finally, the resulting aligned outlines could be analysed, using the multivariate statistical technique principal component analysis, to produce the model.

The model that resulted from this process was comprised of the mean pedestrian shape (i.e. the average of all of the training shapes), and a set of eigen vectors that represent how that mean shape was allowed to vary and still belong to the set of legal pedestrian silhouettes. The eigenvalues associated with the eigenvecotrs give an indication of the degree of variation accounted for by each eigenvector.

The second part of Baumberg's system employed the constructed pedestrian model to track people as they moved through a real life out door scene. To achieve robust tracking in the presence of noise, a predictive mechanism was employed that allowed an initial estimate for where in the image a pedestrian was likely to be, before the silhouette for that pedestrian was identified and compared with the model.

4 Implementation

Given the time constraints of a MSc project it was unfeasible to implement an exact reconstruction of the pedestrian tracker created by Baumberg. However the goal of this project was to evaluate the point distribution modelling techniques used by Baumberg in situations where the objects being modelled were not as complex as human beings, and it was still possible to achieve this goal by implementing only a "stripped down" version of Baumberg's system. The main simplification made was to assume that motion-based image segmentation was possible and that the input data, for both the model-building part of the system and the tracker, was already outline descriptions of the silhouettes of simple three-dimensional objects.

The effect that this has on the model building process is depicted in Figure 4.1 below, given that segmentation of the training data is assumed the remaining model building process consists only of the re-pointing, normalisation and statistical analysis stages.

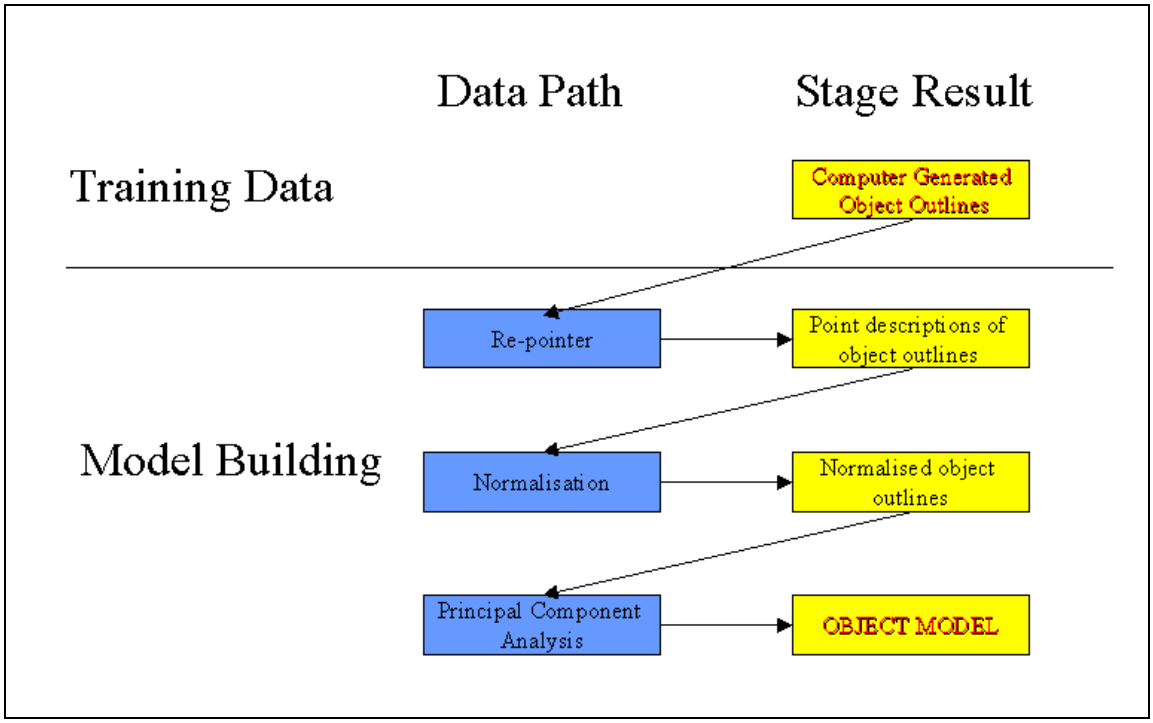

Figure 4.1: *A diagram representing the stages involved in model construction after simplification.*

In addition, given that the input data was known to be clean, noise-free and uncluttered outline descriptions of simple shapes, the tracking part of Baumberg's system could also be greatly simplified. Instead of having to employ the predictive mechanism outlined in the last chapter, all that it was necessary to do was to extract the shape-vector descriptions of moving object silhouettes using the same methods that were used to extract training shape descriptions, and compare these to the object model.

4.1 General Comments

The system implemented here was written in Java 1.2[6] on a 333MHz NT Workstation. Also, given that the normalisation algorithm and principal component analysis both rely heavily on matrix algebra, a Java package that contained common matrix operations was essential to this project. Rather than "re-inventing the wheel" a third party package (Jama **[17]**) was found and used. This package includes the eigenvalue decomposition necessary for principal component analysis and the singular value decomposition necessary for normalisation, as well as most common matrix operations.

4.2 Input Data

The data used in this project, both to construct a model and as input for the tracker, were synthetic two-dimensional outline descriptions of simple convex three-dimensional objects. The sequences are intended as realised results of contour finding applied to the results of Baumberg's image-differencing motion detection, performed on image sequences resulting from a moving camera continually aimed at a stationary object.

[6] Java may not operate fast enough to be a suitable choice for implementing an object tracking system that needs to work at video frame rate. However the system implemented here is intended as a proof of concept rather than as a system that could function in the "real" world, so constraints such as speed of operation do not apply.

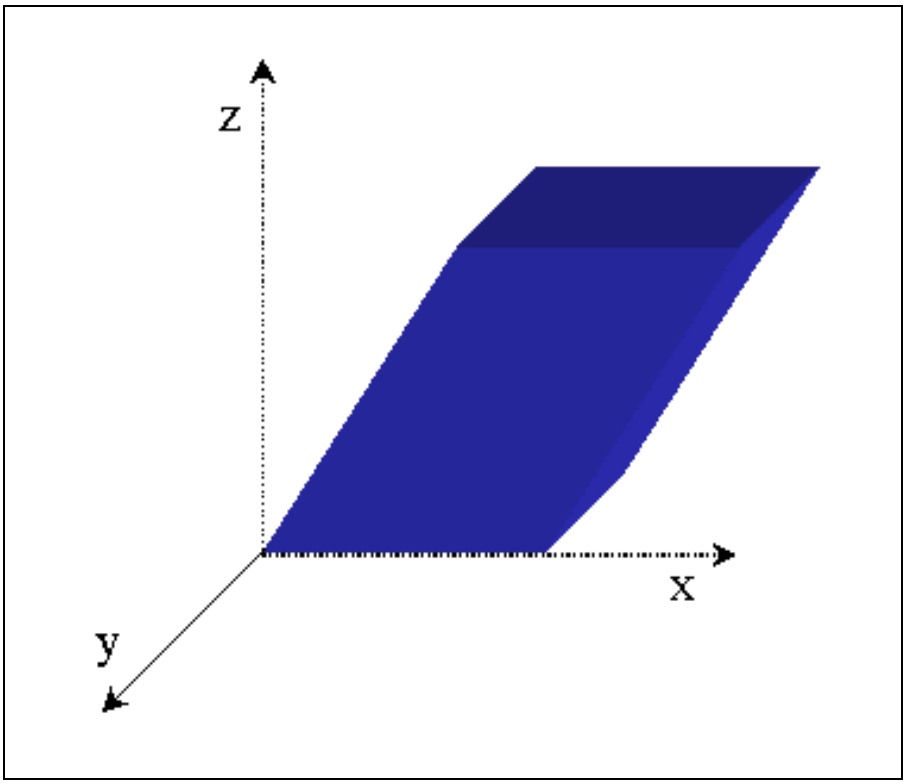

Figure 4.2a: *An example of the kind of simple convex prismatic objects being modelled in this project.*

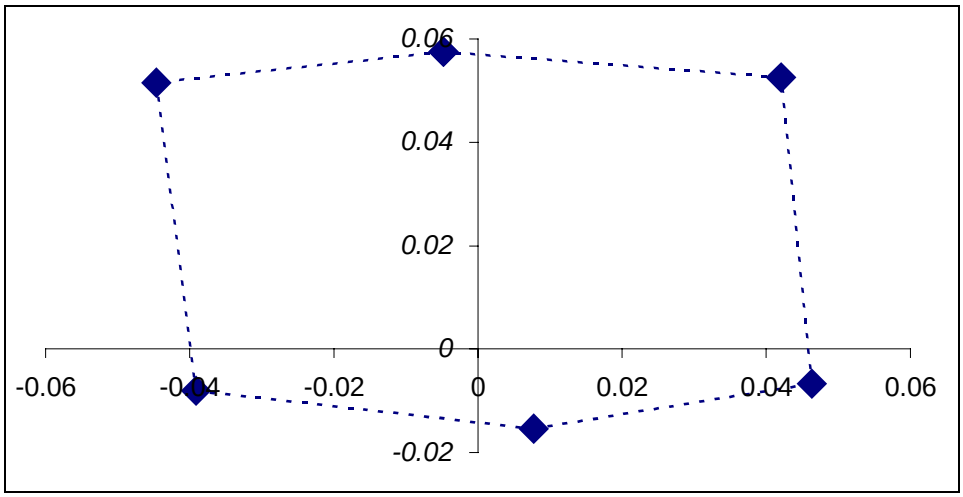

Figure 4.2b: *An example of the outline descriptions produced by the program used to generate the input data for this project.*

To generate this a third party C program was used. This program allows the user to describe the shape of simple convex three-dimensional objects, state from which direction the objects are being viewed, and to specify how the camera position and direction varies over time. The output from the program is a set of matrices that describe the positions of each object's outline silhouette as the camera view changes. The silhouette itself is described as a list of x, y co-ordinates that represent the vertices of the

silhouette. All 3-D objects used here are prisms formed by extruding a planar (2-D) convex polygon 'template' along an axis perpendicular to the template's plane. Figure 4.2a above, shows an example of the kind of 3-D objects specified, and Figure 4.2b an example of the output produced from the program.

4.3 Building the Object Model.

4.3.1 The Training Data.

The accuracy and therefore usefulness of the model is largely dependent on the training data that was used to generate it. For example, if a system being built to recognise and track people in a bus station is trained only with images containing standing human figures then it will to fail match against people who are seated, or walking.

As the type of training data required is clearly dependent on the end-use of the system, it is very difficult to be prescriptive about exactly what constitutes a "good" training set. Indeed, there is no in-depth discussion of training data in any of the referenced literature. However, even though there is no quantitative analysis of training data requirements there are at least two qualitative guidelines. First, the training data needs to contain images that cover the range of object aspects that are likely to exist in the situation the system is being built for. Second, there is a desire to be able to train the model automatically. For example, Baumberg used as training data 15 minutes of live video images of the pedestrian scene that his system was designed to observe. In addition to this, once the human silhouettes had been extracted from this data, each was reflected about its principal axis to produce even more valid training data.

For this piece of work the second guideline is of less relevance, as this system is only intended to be a "proof of concept" rather than a system that is designed to work in the real world. However, even if real rather than synthetic data were being used, it is doubtful that the model building phase could be

entirely automatic. Even Baumberg carefully selected which video sequences to use as training data such that they would not be too cluttered with moving objects.

Given the irrelevance of the second guideline, the training data simply needs to contain the range of silhouettes that the objects are likely to exhibit in the field.

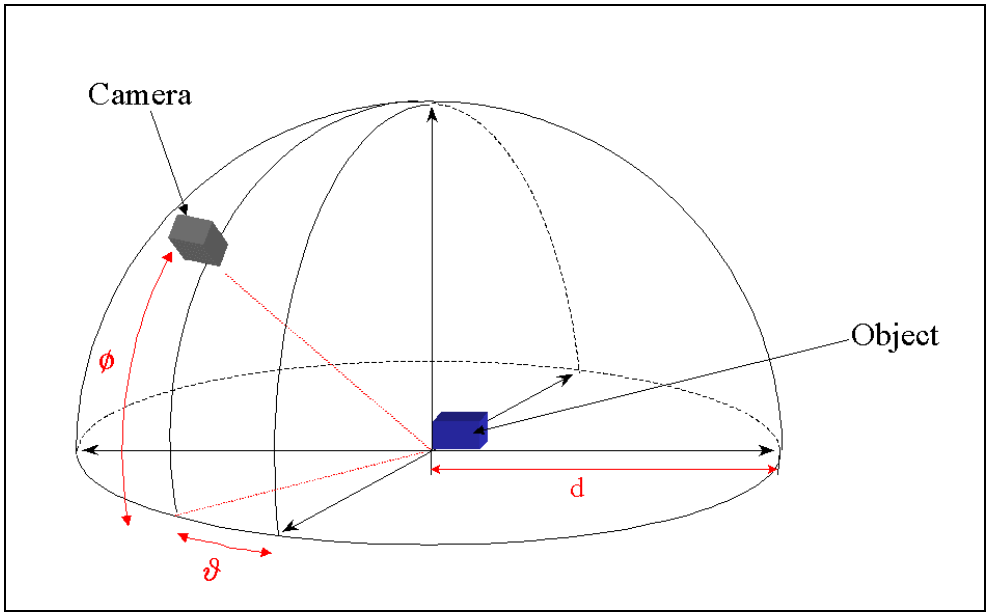

Figure 4.3a: *A diagram depicting the upper view hemisphere of the object camera.*

To achieve this it is worth thinking about a real world application area. For example, if a system were being built to identify and track vehicles one could argue that the best way to collect training data would be to film a number of stationary cars by moving the camera around those cars. To achieve good coverage of the set of all possible viewing positions and angles using this system there are at least four[7] independent variables that need to be considered, as depicted in Figure 4.3a & Figure 4.3b. These variables are: -

- θ. The angle the camera makes with the principal horizontal axis of the object.
- φ. The 'tilt' angle the camera makes with the ground plane.
- d. The distance between the camera and the origin of the object.

- α. The angle of acceptance of the lens ($\alpha \to 0$ for a telephoto lens and $\alpha \to \infty$ for a wide-angle)

In practice the number of possible views of an object is actually constrained by the circumstance of the viewing. For example, for a standard surveillance camera the angle of acceptance and the tilt angle are liable to remain fixed. In addition any variability in the distance from camera to object is going to be removed by the normalisation part of the model building process. So in this work training sets of different objects were generated by keeping the camera's angle of acceptance, camera tilt and distance from the object constant, then taking a sequence of images of the object at suitable intervals of θ.

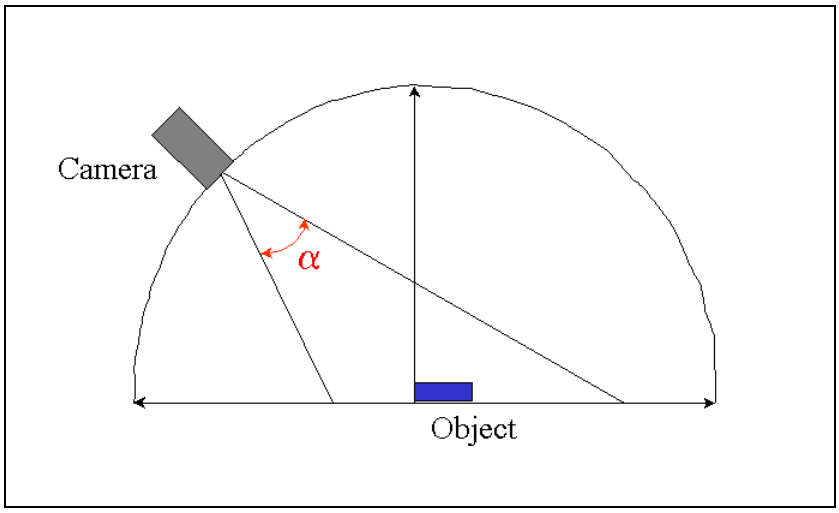

Figure 4.3b: *A diagram depicting the side elevation of the camera's upper view hemisphere. Showing the camera's angle of acceptance, α*

4.3.2 Re-Pointing.

As the input data is simply a description of the vertices of each shape a re-pointing algorithm is required to change this input into a form that complies with the criteria discussed in Chapter 3.1.4. Certainly, a vertex description will not guarantee that the same number of points is used to describe each shape, even if on similar shapes the location of the points correspond to the same object feature.

[7] If the object were large, or the camera was close to the object with a small angle of acceptance, then the position of the origin in relation to the object would be another variable that would need to be

The re-pointing algorithm that has been implemented places the first landmark point at the highest point of the contour in the image. If the top of this contour is a horizontal line then the first landmark point is placed at the right extremity of that line. Then the rest of the desired number of points are placed equidistantly around the outline. By always ensuring that the first point is located in the same place it is guaranteed that for two almost identical shapes, any variation will be due to the difference in the shape rather than to a difference in the way that the points were allocated.

In fact, using the highest point of the shape to place the first landmark point was suggested by Baumberg as an alternative to his method of finding where the principal access cuts the image boundary, for objects such as cars.[8]

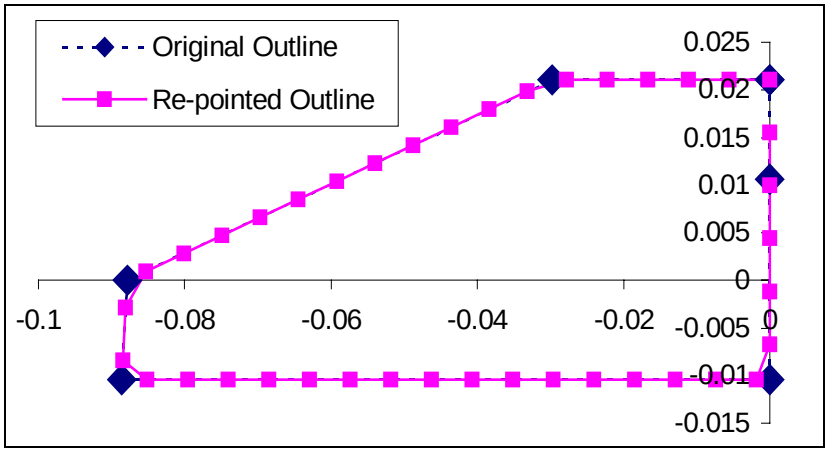

Figure 4.4: *A training shape before and after re-pointing*

The re-pointing algorithm was tested by visually comparing input and output shapes. An example can be seen in Figure 4.4 above.

4.3.3 Normalisation Algorithm

The normalisation algorithm implemented was generalised procrustes analysis as outlined in section **3.1.5**.

---

considered.

There were several pieces of the normalisation algorithm as derived by Gower **[13]** that were not strictly necessary for the algorithm to function correctly in this circumstance. For example, the initial scaling step where each of the original shapes, $\mathbf{S}_i$, is scaled by λ such that: -

$$\sum_{i=0}^{m-1} \lambda^2 tr(\mathbf{S}_i \mathbf{S}_i^T) = m \qquad \textbf{(4.1)}$$

was included by Gower simply to ensure that the numbers generated by the algorithm were "friendly" enough for the computers available in 1975 to be able to handle them accurately. In addition, the initial rotation step was only included in an attempt to make the normalisation algorithm symmetrical (i.e. in the case of two matrices, rotating and scaling **X** to fit **Y** is equivalent to rotating and scaling **Y** to fit **X**). For this work it would have been acceptable to have a non-symmetrical algorithm that scaled and rotated all of the shapes to fit the first shape of the set.

However, I decided that the algorithm should be implemented as described by Gower for two reasons. In the first place, Baumberg did not discuss his implementation of the algorithm at all and hence provided no indication of whether or not he simplified Gower's work. Given that this work is a rational reconstruction of Baumberg's system, it was felt that the "safest" option was to implement the algorithm exactly as Gower had defined it.

In addition Gower provided test data for a worked example in his paper (Please see Appendix B for this test data), and this test data proved to be the most convenient way to verify that the algorithm was working correctly. However, to gain identical results to Gower the algorithm had to be implemented exactly as Gower had intended.

The operation of the algorithm was also tested on three triangles as depicted in Figure 4.5. Although this was a very simple test case it allowed easy visual verification of all the stages during debugging.

---

[8] Although Baumberg did not specify what to do should the upper most part of the shape be a horizontal line rather than a single point.

Although it may not be clear from the level of description found here, identifying and dealing with the errors and omissions in Gower's article describing generalised procrustes analysis caused significant problems and incurred heavy time delays.

4.3.4 Principal Component Analysis.

The statistical analysis of the training data was achieved by implementing principal component analysis (PCA) as outlined in section **3.1.7**.

The operation of the algorithm was visually inspected on some simple distributions of two-dimensional points, an example of which can be seen in Figure 3.5

To verify that the algorithm was functioning correctly in more than two dimensions the effect of PCA on a set of ellipses with equal height but varying width was examined. (See Figure 4.6 for a depiction of the input data)

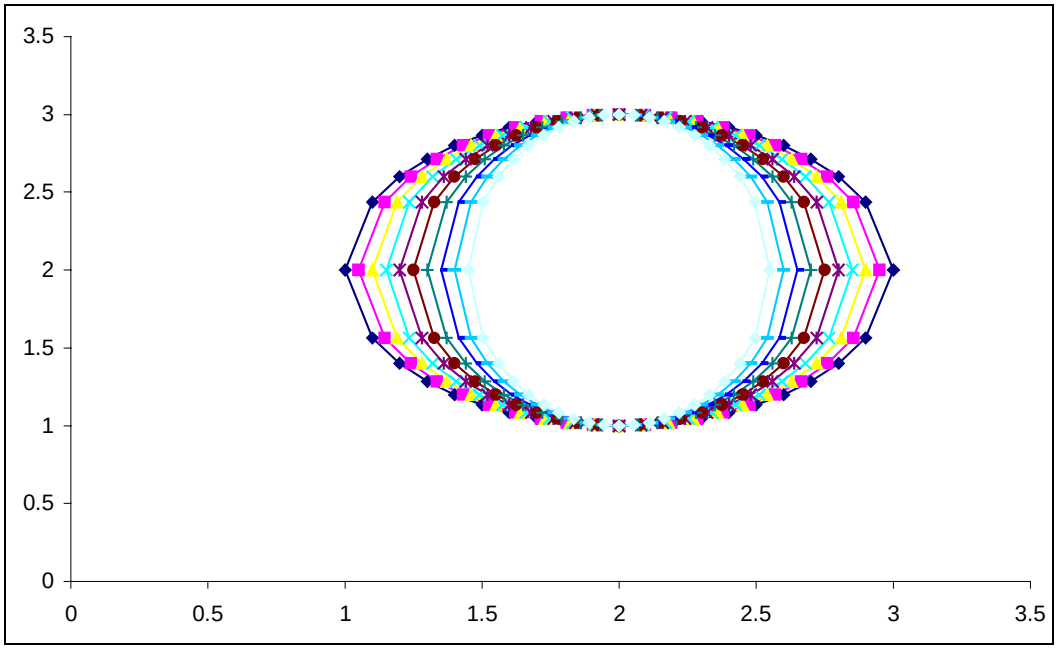

Figure 4.5: *A graph representing the set of ellipses used to verify the operation of the principal component analysis algorithm.*

When the PCA algorithm was applied to the above mentioned shapes only one of the resulting eigenvectors had a corresponding eigenvalue that was not equal to zero. In other words there was only one "direction" in which all of the variation of the original data occurred. This was the expected outcome, as to produce a set of ellipses with different widths only one variable in the equation for an ellipse is altered. This means that there is actually only one linearly independent variable that describes the variation of the shapes, and therefore there can only be one direction of variation resulting from the principal component analysis of that set of shapes.

As is it difficult when there are more than two dimensions to visualise the "directions" that eigenvectors resulting from PCA correspond to, a method was needed to view and verify the outcome of the algorithm when sets of shape vectors, rather than sets of two-dimensional points, were analysed.

The method used to visualise a mode of variation, as represented by an eigenvector and eigenvalue pair, was to produce a set of shapes that were the result of varying the mean shape, within suitable limits along the "direction" denoted by the eigenvalue for that mode. Explicitly, for the i^{th} mode of variation, shape-vectors $\mathbf{S}^{(j)}$ are calculated using: -

$$\mathbf{S}^{(j)} = \mathbf{S}_m + step * j * (\sqrt{\lambda_i}) * \mathbf{e}_i \qquad \textbf{(4.2)}$$

Where $\mathbf{S}_m$ is the mean shape, and j varies between $-k$ and k (e.g. j = -2, -1, 0, 1, 2). The variable *step* multiplied by j controls how much the mean shape is varied along the direction denoted by the eigenvalue $\mathbf{e}_i$. For example, if *step* is set to be 0.5 and the values of j are as above, then five shapes will be produced. The first shape will differ from the mean by –1 standard deviation where the standard deviation of the distribution of the possible shapes for that mode is given by the square root of the corresponding eigenvalue, λ_i. The second, forth and fifth shapes will differ from the mean by –0.5, 0.5, and 1 standard deviations respectively and the third shape will be the mean shape of the input set.

The visualisation of the only mode of variation resulting from the analysis of the ellipse shapes is shown below in Figure 4.6.

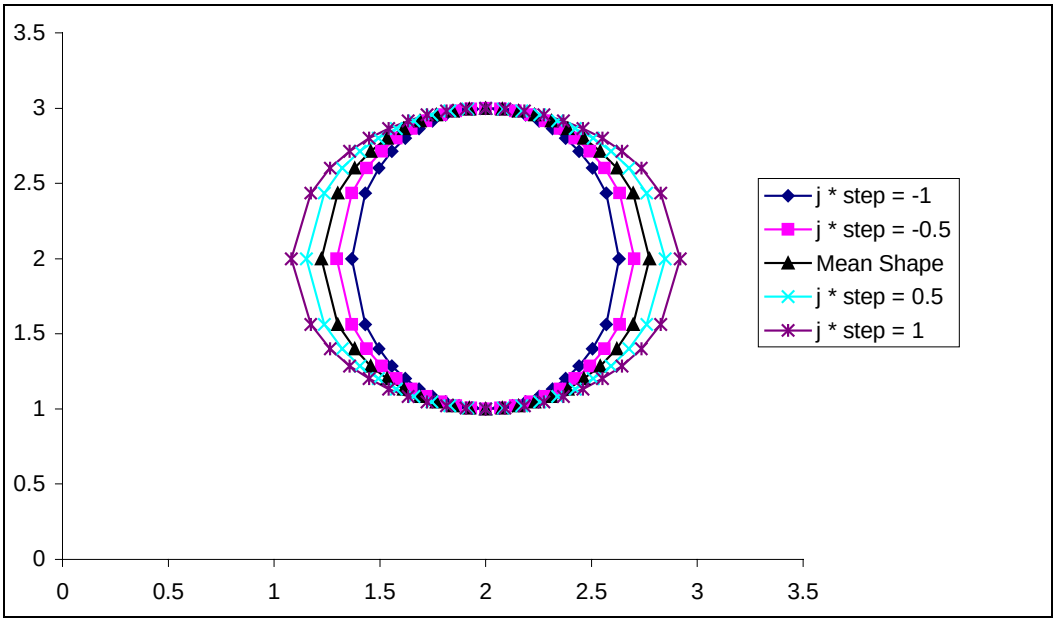

Figure 4.6: *A visualisation of the single mode of variation produced by applying principal component analysis to a set of ellipses of varying width.*

4.4 Object Tracker.

Given that real image data was not being analysed the tracking mechanism implemented by Baumberg was reduced to a much simpler object recognition mechanism. This mechanism consisted of three principal parts. First the input shape was re-pointed using exactly the same re-pointing algorithm that was employed in the model building stage, as is described above in Section 4.3.2. Then the re-pointed shape was aligned to mean shape of the model at which point the resulting shape could be compared to the model to see whether or not it was within the set of shapes allowed.

As the re-pointing algorithm implemented here was identical to the algorithm describe above, only the normalisation and identification processes will be discussed here.

4.4.1 Normalisation.

When applied to two shapes and implemented as described above (Section 4.3.3), generalised procrustes analysis will align those two shapes *to each other*. Hence it was not possible to employ the

above algorithm to rotate and scale the input shape to fit the model's mean shape whilst keeping the mean shape fixed.

To align the input shape to the model's mean shape a non-generalised procrustes analysis was implemented as follows.

First the model's mean shape, $\mathbf{M}_s$, and the input shape, $\mathbf{S}$, were aligned to each other using the generalised procrustes analysis algorithm described above, to produce to new shapes, ${}^{new}\mathbf{M}_s$ and ${}^{new}\mathbf{S}$. The new rotated and scaled mean[9], ${}^{new}\mathbf{M}_s$, is given by: -

$$ {}^{new}\mathbf{M}_s = \lambda * \rho_{mean}\mathbf{M}_s\mathbf{H}_m \qquad \textbf{(4.3)} $$

Where λ is the initial scaling factor, and $\lambda * \rho_{mean}$ is the total amount that the mean shape has been scaled by, and $\mathbf{H}_m$ is the matrix describing how the mean has been rotated.

The new rotated, scaled and translated input shape, ${}^{new}\mathbf{S}$, is given by: -

$$ {}^{new}\mathbf{S} = \lambda * \rho_{shape}\mathbf{S}\mathbf{H}_s + \mathbf{T}_s \qquad \textbf{(4.4)} $$

Where, $\mathbf{T}_s$, describes the translation that the input shape undergoes, $\lambda * \rho_{shape}$ is the total scaling factor, and $\mathbf{H}_s$ is the matrix describing how the input shape has been rotated.

Next, the newly aligned input shape was re-aligned to the original mean shape. This was achieved by calculating what the scaling and rotation factors applied to the mean shape in the above operation were, and applying the inverse of these to the previously aligned input shape, ${}^{new}\mathbf{S}$. In other words, the same

[9] The model's mean shape is not translated during the generalised procrustes analysis, as it was placed at the origin when calculated originally.

rotation and scaling operations that would translate the new mean, ${}^{new}\mathbf{M}_s$, to be identical to the original mean, $\mathbf{M}_s$, needed to be applied to the new input shape, ${}^{new}\mathbf{S}$.

Equation 4.3 indicates that the scaling factor applied to the original mean shape was equal to $\lambda * \rho_{mean}$ and both of these scaling factors are available as results of the generalised procrustes analysis algorithm. However the rotational matrix $\mathbf{H}_m$, also from equation 4.3 above was not available as a result, and therefore needed to be calculated.

Rather than find the rotation matrix that rotates the original mean to the new version of the mean, and then finding the inverse of that matrix, the necessary rotation factor was found directly by using principal value decomposition, as described previously in section **3.1.5**. The matrix that rotates the new mean, ${}^{new}\mathbf{M}_s$, back to fit the original mean, $\mathbf{M}_s$, is given by: -

$$\mathbf{H} = \mathbf{V}^T\mathbf{U} \tag{4.5}$$

where $\mathbf{U}$ and $\mathbf{V}$ are given by: -

$$\mathbf{M}_s^T\,{}^{new}\mathbf{M}_s = \mathbf{U}^T\Gamma\mathbf{V} \tag{4.6}$$

Once the rotational matrix had been calculated the previously aligned input shape, ${}^{new}\mathbf{S}$, is re-aligned to fit the original mean using: -

$${}^{Final}\mathbf{S} = \frac{1}{\rho_{mean} * \lambda}\left({}^{new}\mathbf{SH}\right) \tag{4.7}$$

The above described non-generalised procrustes analysis algorithm was tested by visually examining the shapes before and after they were normalised.

4.4.2 Object Identification.

Once the input shape has been re-pointed and aligned to the model's mean shape, it is possible to project the input shape into the model space and ascertain if the resulting shape parameters lie within acceptable limits.

The input shape is projected into the model space, as derived in Section **3.1.7**, by using: -

$$\mathbf{b}_i = \mathbf{P}^T \left(\mathbf{s}_i - \mathbf{m}_s \right) \tag{4.8}$$

Where **b** is the column vector of shape parameters (i.e. $\mathbf{b}_i = \left(b_0, b_{1,}, ..., b_{n-1} \right)$ where *n* is the number of principal components retained by the model), $\mathbf{P}^T$ is the model's eigenvector matrix, $\mathbf{s}_i$ is the input shape-vector and $\mathbf{m}_s$ is the models mean shape. As discussed in Section **3.1.7**, suitable limits for testing to see if an input shape as an allowable view of the object that is described by the model, are: -

$$b_i = \pm 2\sqrt{\lambda_i} \tag{4.9}$$

where λ_i is the corresponding eigenvalue.

4.5 Summary

In this section a description of the system implemented to explore the effectiveness of point distribution models when used to model simple convex prismatic objects has been presented.

The system implemented here was a simplified version of Baumberg's work, where the initial image segmentation stages were assumed to be possible and therefore the input data, into both the object builder and the tracker, was already available in the form of outline descriptions of simple object silhouettes. This meant the object builder implemented here needed to consist only of re-pointing, normalisation and principal component analysis stages.

In addition, by cutting out the use of raw image data, the tracking part of the system used for testing the operation of the built models was also simplified. Instead of needing to employ Baumberg's predictive mechanism, it was enough to use methods similar to those used in training the model to extract outline descriptions of the input data, and compare the resulting descriptions directly to the model.

5 Results: Preliminary Exploration

In order to investigate the performance of point distribution models where the objects being modelled are simple convex rigid bodies, several models of different simple objects were constructed and these models were then applied to object recognition tasks.

For a preliminary exploration, models were constructed for the four objects shown below in Figure 5.1. For the rest of this document the naming convention adopted will be to refer to these objects by their distinguishing faces, i.e. square, parallelogram-1, parallelogram-2, and parallelogram-3.

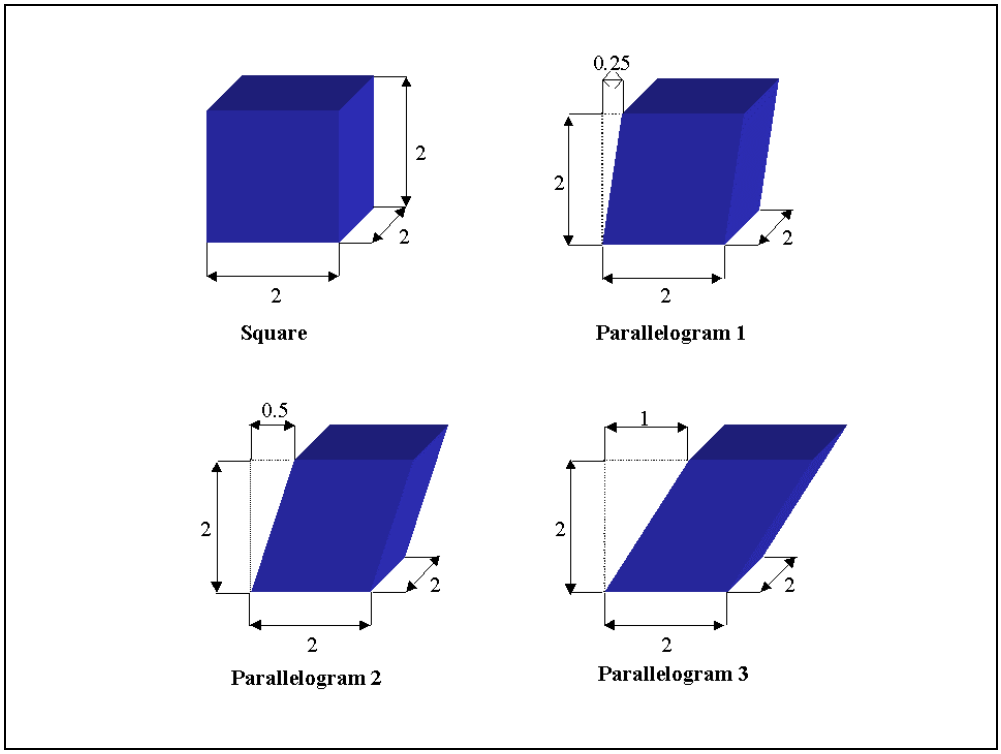

Figure 5.1: *A representation of the three-dimensional objects being modelled. (Not to scale)*

The training data used to construct an object model comprised of outlines of the object as seen by a camera as it rotated around that object. In other words, the camera's angle of acceptance (α), tilt (ϕ) and distance from the object (d) were held constant, while the angle the camera made with the principal horizontal axis of the object (ϑ) was varied, in suitable steps, between 0 and 360 degrees. (As depicted in Figure 5.2 below)

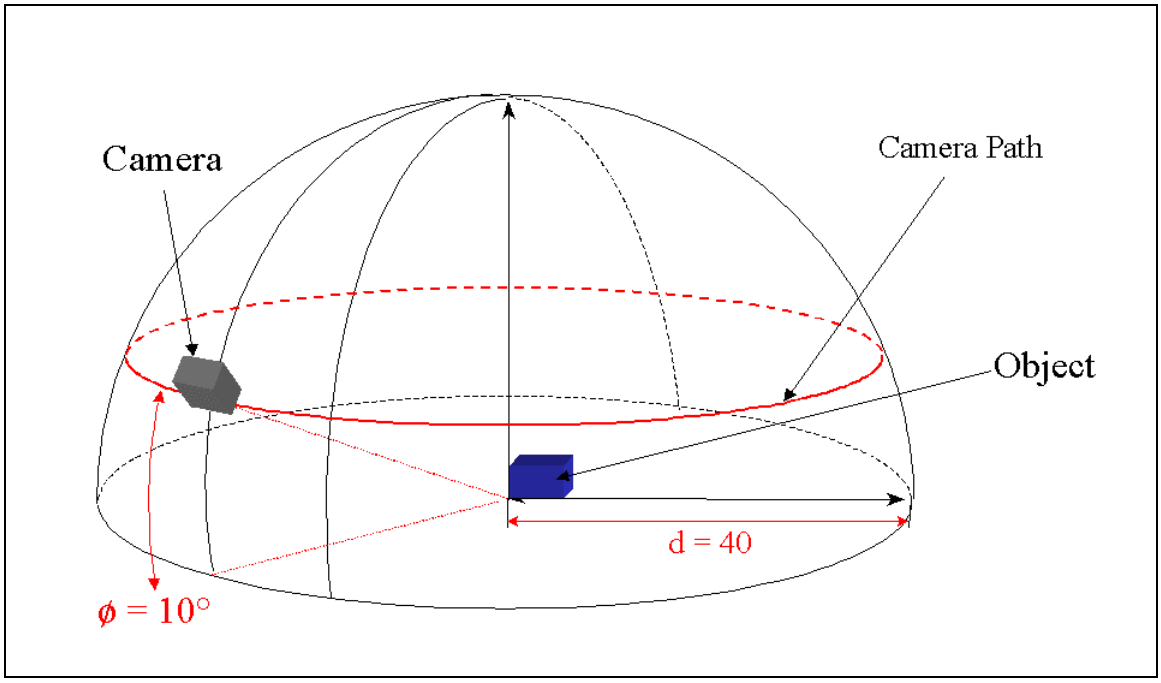

Figure 5.2*: A representation of the camera path used to produce a sequence of training images.*

The values for α, ϕ, d, and $\Delta\vartheta$ (the amount by which ϑ varied between "frames") were chosen to be: -

- $\alpha = 45^{o}$
- $\phi = 10^{o}$
- d = 40[10]
- $\Delta\vartheta = 10^{o}$, where ϑ ranges from 0^{o} to 360^{o}

These values were chosen (reasonably arbitrarily) such that the training set produced was of a nature that allowed easy visual inspection of the object silhouettes. Given this choice each training set contained 36 outlines, some of which are displayed below in Figure 5.3.

[10] Where distance in measured in some arbitrary unit.

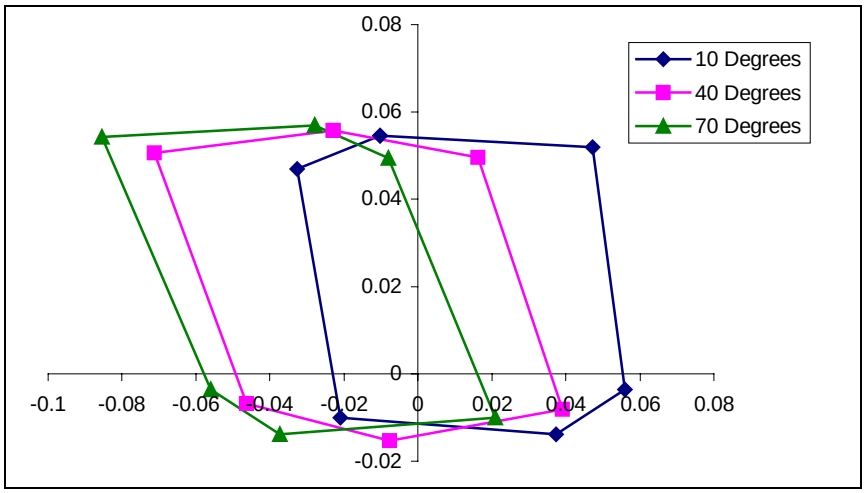

Figure 5.3: *Examples from the set of training data generated for parallelogram-3*

Several versions of model were built for each object, where the compactness of the model was varied in two ways. First, different models of the objects were constructed by varying the number of points added by the re-pointer to describe each shape in the training set. Second, the amount of amount of variation of the training data that the model finally accounted for was varied. In other words, the number of eigenvectors (resulting from the principal component analysis of the training shapes) retained by the model was varied.

When a model is presented with an input shape there are four different identifications that the model can make depending on whether or not the input shape is of the class of shapes being modelled. These are: -

- True positive: The shape is correctly identified as fitting with the model.
- True negative: The input shape is correctly rejected.
- False positive: The input shape is incorrectly identified.
- False negative: The input shape is incorrectly rejected.

The overall accuracy of a model can be examined by looking at what percentage of input shapes are correctly classified, i.e. how often a model produces a true positive or a true negative when it should. However, a model can be incredibly good at positively identifying shapes of its own class without necessarily being good at rejecting shapes corresponding to a different object. Therefore it is also

useful to look at what percentage of input shapes that do correspond to the model are correctly positively identified in isolation from what percentage of input shapes that do not correspond the model are correctly rejected.

The data presented to the models here, to see how well they performed at identification tasks, was actually the training data described above. So for each set of training data the corresponding model should always be able to positively identify input shapes as belonging to the class of shapes described by that model. However, it was expected that given how similar the four objects being modelled were, that the models would falsely identify input shapes that are views of objects which were not the object being modelled.

The first set of models was constructed such that they retained the modes of variation that accounted for 99% of the variance of the training data. Several of these models were built for each object where the number of points used in the re-pointer to describe the training shapes varied between 10 and 80.

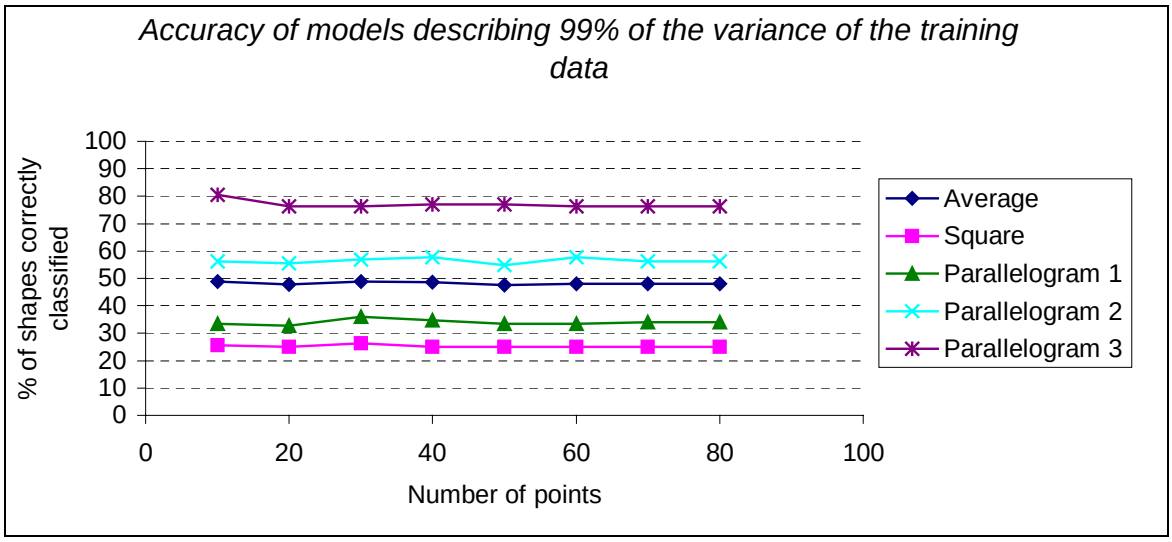

Figure 5.4: *A graph showing the accuracy of classification against number of points used to describe the training shapes for each of the four models and the overall accuracy of identification for all four models.*

Looking first at the overall performance accuracy (Figure 5.4), it was found that the combined accuracy of the models when presented with all four sets of input data was slightly below 50%. In other words, whilst working in parallel, the models misclassified (i.e. either produced a false negative or a false positive identification) input shapes over 50% of the time. Out of the four models, the model of

parallelogram-3 had the best overall performance and correctly classified input shapes just under 80% of the time.

In addition the accuracy of the individual and overall performances were not affected to any significant degree by changing the number of points (within the limits tried here) that were used to describe each training shape.

Looking at how accurately the models could positively identify shapes of its own class (Figure 5.5), it was found that over all four models input shapes were only correctly positively identified around 90% of the time. By this measure the square model had the best performance, being the only model to achieve 100% recognition of its own shapes. However this result coupled with the fact that the square model only achieved an overall performance of around 25% indicates that the model for the square was too general and allowed all shapes belonging to the three other objects to be incorrectly identified. Conversely, the parallelogram-3 model achieved good overall results, and yet was the model least able to identify shapes of its own class.

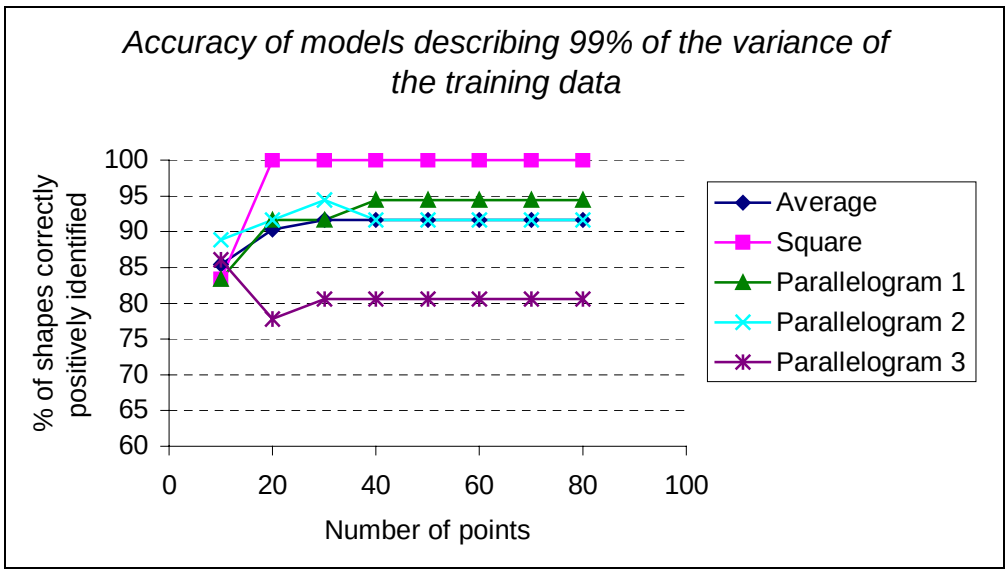

Figure 5.5*: A graph showing the accuracy of positive identification against number of points used to describe the training shapes for each of the four models and the overall accuracy of identification for all four models.*

Again, changing the number of points used to describe the training shapes had little effect on improving the accuracy of the models.

Next, a set of models was constructed such that the models all used 30 points to describe the training shapes, while the amount of variation of the training data that the models accounted for was varied.

As can be seen from Figure 5.6, only the overall performance of the models was not significantly effected by changing the number of modes of variation retained by the model.

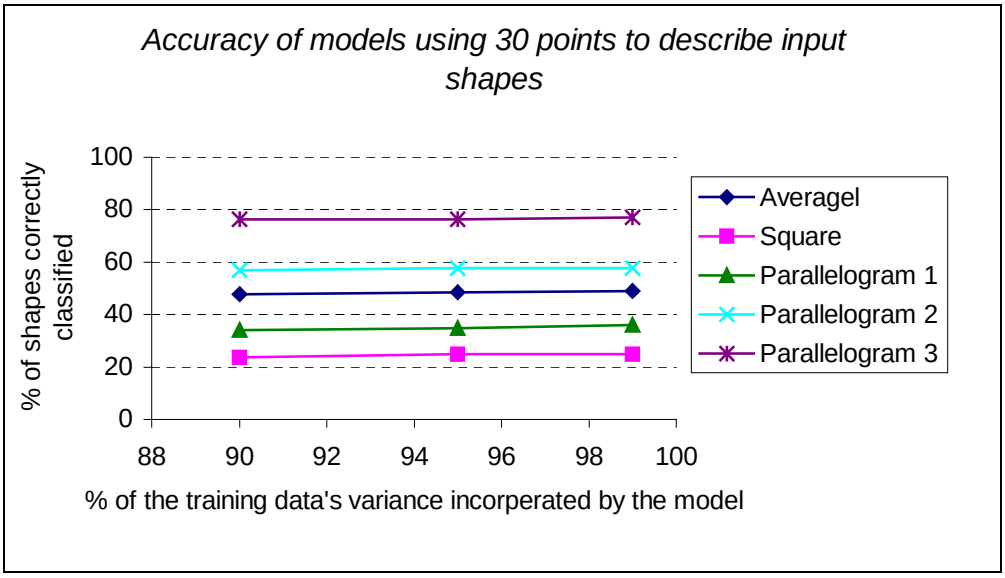

Figure 5.6: A graph showing the accuracy of classification against the percentage of the variance of the training data retained by the model.

Looking at how well models identified shapes of their own class (Figure 5.7), as before the square model had the best performance, and indeed was the only model to achieve 100% accuracy on positive identification tasks. All of the models remained largely unaffected by changing the number of modes of variation retained by the models.

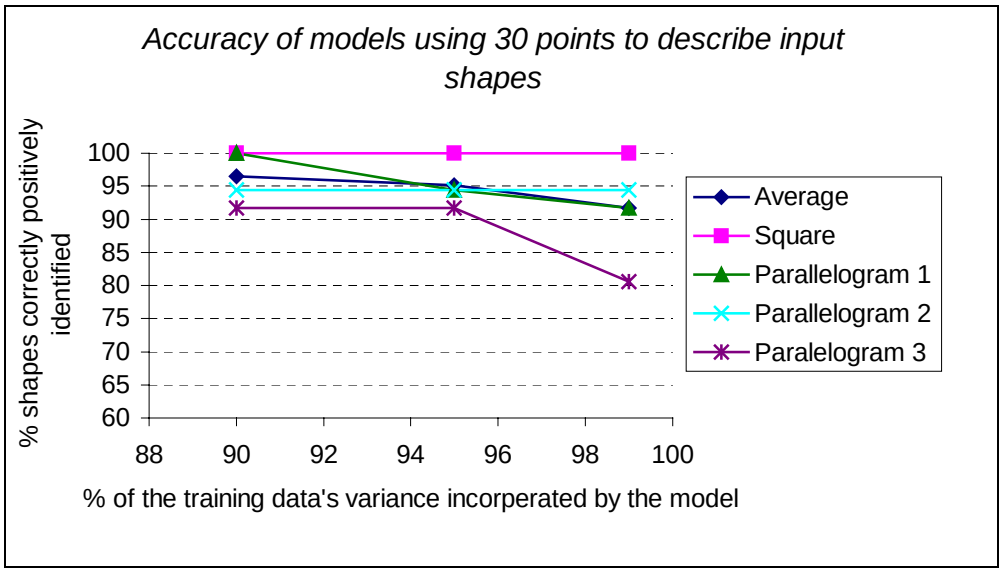

Figure 5.7: *A graph showing the accuracy of positive identification against the percentage of the variance of the training data retained by the model.*

6 Discussion and Further Exploration

6.1 Discussion of Initial Results

The most significant result gained from the initial exploration was the fact that, apart from the square model, all of the models failed some of the time to correctly identify shapes of their own class. This was particularly worrying, as the input data used to test the models was identical to the data used to train them. When a training shape is projected into the model space the outcome is a column vector of shape parameters given by: -

$$\mathbf{b}_i = \mathbf{P}^T(\mathbf{s}_i - \mathbf{m}_s) \qquad \textbf{(6.1)}$$

where $\mathbf{P}$ is the eigenvector matrix and $\mathbf{m}_s$ is the mean shape. Across the whole training set, the variance of the i^{th} shape parameter is given by the corresponding eigenvalue λ_i. The distributions of the shape parameters is assumed to be normal and an input shape is considered to belong to the object described by the model if the shape parameters (corresponding to the input shape) all lie within 2 standard deviations of the mean (i.e. $b_i \pm 2\sqrt{\lambda_i}$). In other words, given the assumption that the distribution of the each shape parameter is normal, a model should correctly positively identify input shapes at least 99% of the time when the input shapes are identical to the set of training shapes.

This level of accuracy in positive identification tasks was clearly not achieved by three of the four models tested in the initial exploration phase, leading to the hypothesis that the shape parameters of the training shapes are not normally distributed. Further investigation confirmed this hypothesis. In addition it was found that not only are the distributions of the shape parameters not normal, but they also contain multiple peaks. It is these unexpected extra peaks in the distributions that cause the models to be unstable and therefore degrade their performance at positive identification tasks. This is discussed in more detail below.

The unexpected nature of the shape parameter distributions is caused by the way that the re-pointing algorithm works. Point descriptions of all of the input shapes were gained by taking the highest point[11] of the input shape (this is known as the reference point) as the position of the first point, and then distributing the rest of the points around the outline with equidistant spacing. For the most part, this approach results in the same point on the 3D object being identified as the location for the first point to be placed on the 2D image space projection of that object. However, as the view of the object changes, there are times when over a small change in view the first point positioned on the 2D object outline stops corresponding to one part of the object and 'jumps' such that it then corresponds to an entirely different part of the object. For example, for a 3-D parallelogram when the camera angle ϑ varies between 1° ~ 89° the first point of the outline description will correspond to the top left-hand corner of the face at the rear of the object as this is the highest point of the object within the image plane. As the camera view varies from 89° ~ 90° the highest point of the object (and therefore the position of the first point) flips to be what was originally the top left-hand corner of the object's front face. (Please see Figure 6.1)

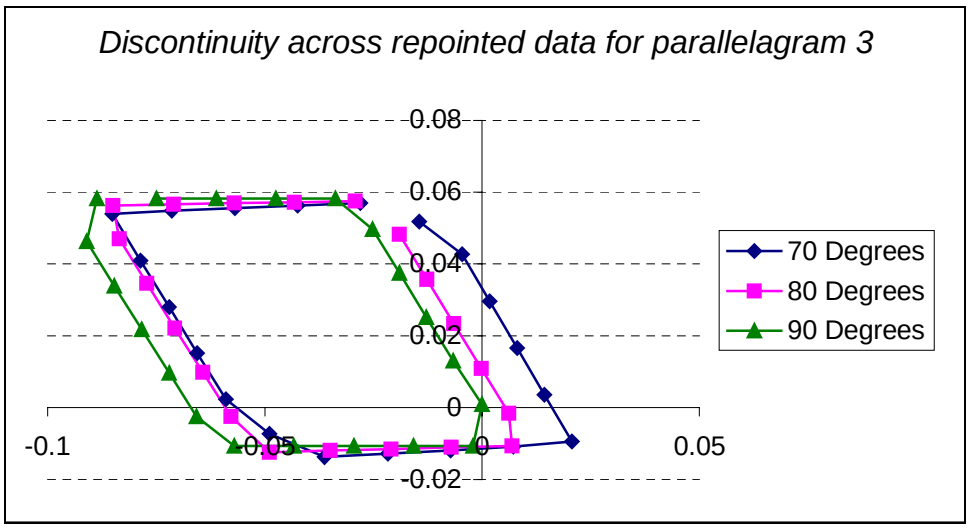

Figure 6.1: *A diagram showing the "jump" of the positioning of the first outline point as the view of parallelogram-3 passes through 90 degrees.*

As can be seen from Figure 6.2, these discontinuities in the shapes descriptions are preserved even after normalisation. The result of the normalisation process is clusters of shapes, where each cluster corresponds to views of the object where the first point of each outline description is constant, and the

[11] If the highest part of the image is a horizontal line then the right most extremity of that line is chosen

discontinuity between clusters corresponds to a shift in what features of the object the points correspond to.

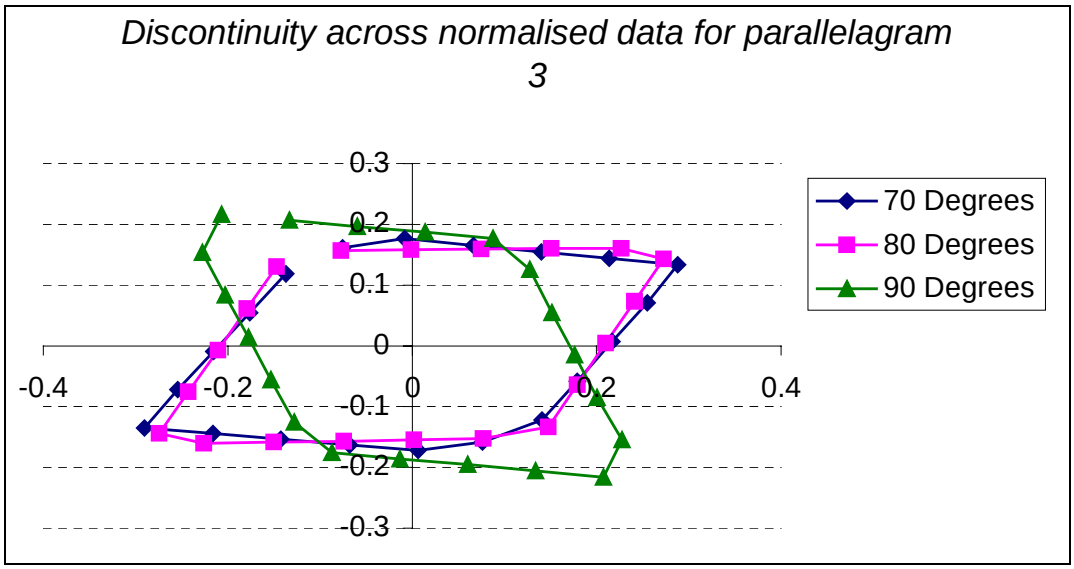

Figure 6.2: *A diagram showing 'jump' in the positions of the outline points after normalisation has occurred.*

The reason that the square model behaved better (at positive identification tasks) than the other three models tested was because of the symmetrical nature of the cube object being modelled. When the reference point jumps to another location on the cube, because of rotational symmetry, the new outline description produced will match an outline description produced when the reference point was at a different object location. For example, the outline description of a cube centred on the origin and viewed at 91° exactly matches the outline description of the same cube when viewed at an angle of 1°. The slight degradation of the square model performance can therefore be explained by the fact that the program that created all of the input data did not allow the objects to be centred on the origin. Instead the objects were placed in the view hemisphere such that front left-hand bottom corner of the object corresponded to the system's origin. This shift in origin is enough to mean that although shape descriptions before and after a shift in the positioning of the first outline point are similar they are not identical, thus leading to a slight deformations in the distributions of the model's shape parameters.

---

as the position for the first point.

6.2 Further Results

To confirm that the incorporation of discontinuities in the outline descriptions was the only thing that caused the bad performance of the models when presented with positive identification tasks, a further set of experiments were run. Here, the models were constructed by constraining the object's view space such that the first point of the outline description of the training shapes always corresponded to the same location on the object.

The same four objects were modelled as before (see Figure 5.1). To produce the training data used to build the models, the camera path used before was constrained such that the angle the camera made with the objects only ranged between 10° ~ 80°, with a 10° variation per "frame". Discontinuities in the positioning of each outline's reference point only occur every time the camera angle passes through 90° (i.e. at 90°, 180°, 270° & 360°), hence by limiting the range of views used as above, the data was known to contain only smooth transitions of the outline points. The training sets constructed contained 8 object views each.

As before, the first set of models was constructed such that the models retained the modes of variation that accounted for 99% of the variance of the training data. Several of these models were built for each object where the number of points used to describe the training shapes varied between 10 and 80. The data presented to the models, to see how well they performed at identification tasks, was again identical to the training data used to create the models.

As can be seen from Figure: 6.3, the accuracy of each model at correctly identifying shapes, as belonging to the class of shapes described by that model, was 100%. And this level of accuracy was totally unaffected by how many points were used to describe each shape. In addition, the overall accuracy of the models performance (i.e. how well the models could correctly identify shapes of their own class in addition to how well the models could correctly reject shapes that did not belong), improved. With the new set of models, correct classification is achieved just under 70% of the time, rather than before when the accuracy was below 50%. (Figure 6.4)

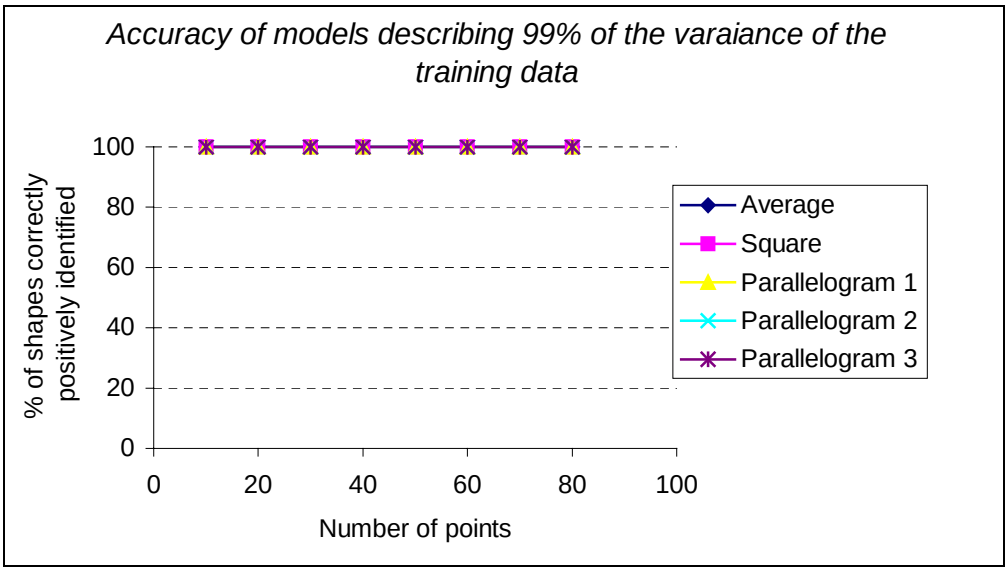

Figure 6.3: *A graph showing the accuracy of positive identification against number of points used to describe the training shapes for each of the four models.*

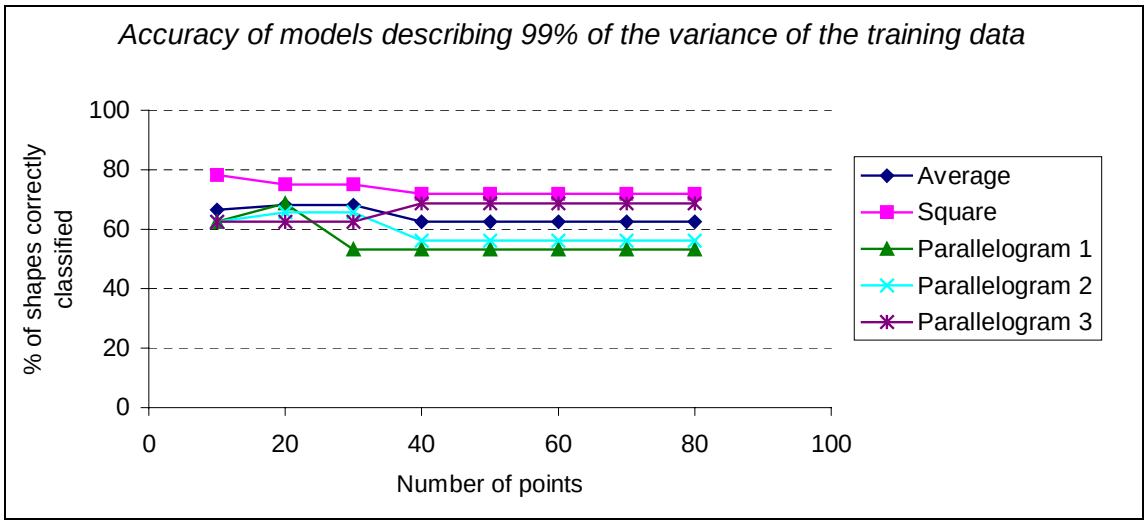

Figure 6.4: *A graph showing the accuracy of classification against number of points used to describe the training shapes for each of the four models.*

Again as before, a second set of models were constructed such that the models all used 30 points to describe the training shapes, while the amount of variation of the training data that the models accounted for was varied.

For the new set of models, regardless of how much variance of the original data was retained by the model, the performance for correct positive identification of shapes remained at 100%. However the overall accuracy for general shape classification dropped from just under 70% to 25% as the amount of

possible shape variation described by the models varied from 99% to 90% (Figure: 6.5). This, in combination with the results concerning the models' ability to perform positive identification tasks, indicates that as the possible shape variation retained by a model is decreased, that model becomes less good at correctly rejecting shapes that do not belong to its class of shapes.

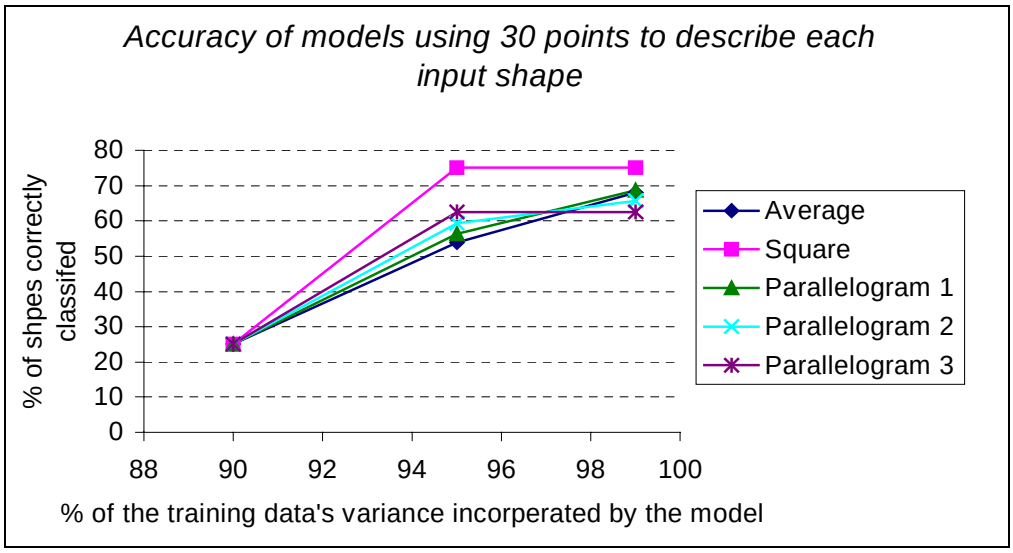

Figure 6.5: *A graph showing the accuracy of classification against the percentage of variance of the training data retained by the model.*

6.3 Further Discussion

The second set of results show that the incorporation of discontinuities in shape outline descriptions was the only thing that caused the bad performance of the models at correctly recognising shapes that were used to construct the model in the first place.

This indicates that for this kind of automatically-constructed point distribution model to be effective in anything other than highly constrained circumstances there needs to be a mechanism for accurately identifying a stable reference point on the object outline. In addition, this fact is true for any objects being modelled regardless of how simple or complex they are.

To a certain extent, it would appear that Baumberg was lucky that he did not encounter this difficulty. Wherever a camera is placed on the surface of the view hemisphere (Figure 6.6), it becomes clear that

(unless the camera angle of elevation (α) is at 90°) the upper most point of the pedestrian silhouette is always going to correspond roughly to the top of a persons head. However, using the same technique to identify a reference point on a silhouette would not work for humans if it was applied in situations where people are either not in an erect posture or where it is likely that they should be holding their hands above their head.[12]

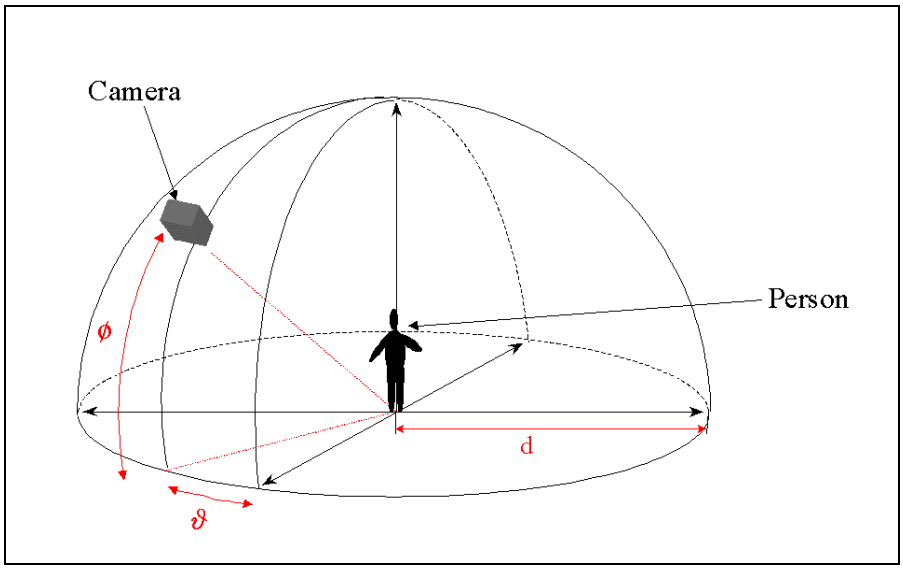

Figure 6.6: *A diagram representing the possible viewing angles of an upright human being.*

For the kinds of objects discussed in this thesis it is not obvious how it would be possible to systematically identify a single stable reference point that lies on the outline of the object. This is because, for the objects studied here, there is no single point of reference on any of the objects that remains in view regardless of where the camera is on the view sphere.

One suggestion for overcoming this problem would be to find a systematic way of automatically identifying a series of reference points, such that for any one view of an object there was at least one of the reference points visible. Then it would be possible to write a normalisation algorithm that takes these reference points into consideration when aligning the input shapes, thus avoiding discontinuities being incorporated into the model.

[12] Thus providing any potential criminals reading this thesis with two ways of fooling the Reading/Leeds automated surveillance system whilst stealing cars.

However, until such a solution to the reference point problem can be devised it is clear that this kind of automated point distribution modelling technique is not going to applicable to a wide range of objects, unless the circumstance of operation is severely constrained.

Once the discontinuities were removed from the shape descriptions of the training data, it was possible to see that the models were still failing to correctly classify input shapes. As the models performance at positive identification was one hundred percent, it is clear that the remaining classification errors were due to the models falsely identifying shapes that did not belong to the sets of shapes being modelled. In other words the remaining source of errors was due to the fact that the models could not systematically distinguish between the different shapes.

Unfortunately, due to time constraints, not enough experimentation was possible to ascertain more than an intuitive feel for how this remaining error rate was affected by varying some of the model parameters. For example, the ability for the models to be able to correctly distinguish between shapes is clearly dependant on how different the objects being modelled are, however the work done here does not provide any quantitative description of how different the shapes need to be. In addition, the models were not tested on data that was not part of the training data, so this work sheds no light on how comprehensively the training data needs to cover the view space for the model performance to be optimised.

However, even from these limited results it is clear that, where simple objects are concerned, a model that only incorporates data concerning how the silhouette of an object varies is going to encounter shapes that it cannot correctly classify. This is because the spaces of allowable outline shapes defined by different models are going to overlap in certain circumstances. In other words, failures to correctly reject shapes will occur when an outline of an object also corresponds to a valid view of a different object. In cases where confusion is bound to occur (simply because there is not enough variation between views of different objects to allow a model of this nature to operate), some secondary characteristics are going to need to be modelled to help resolve conflicts when they occur. For example, in the above case, including information about the positions of all visible object edges, rather than just the edges forming a silhouette, is going to improve the chances of being able to distinguish between

views of the square and views of the parallelograms. However, this kind of additional modelling is a non-trivial problem, as years of research in the field of machine vision will testify.

The question that this project was attempting to answer was whether or not point distribution modelling techniques are still valid in areas where the objects being modelled are simple rigid objects. Certainly this project has shown that, in anything other that very constrained circumstances, not being able to automatically identify a stable object reference point when extracting outline descriptions causes this kind of system to fail badly. Even if it was possible to fix this, the early indications are that it is doubtful whether or not there is enough variation in simple object silhouettes for such a system to be able to systematically correctly classify such objects.

7 Further Work

Before this type of model building technique becomes appropriate for use in a broad range of applications some way needs to be found to extract stable outline descriptions of the objects being modelled. Without the ability to do this, automatically generated point distribution models are not going to operate effectively unless either the circumstance of operation is severely constrained, or the nature of object itself is such that an object-specific solution can be found.

One potential solution that could be investigated would be to find a systematic way of automatically identifying a series of reference points, such that for any one view of an object there was at least one of the reference points visible. Then it would be possible to write a normalisation algorithm that takes these reference points into consideration when aligning the input shapes, thus avoiding discontinuities being incorporated into the model.

In addition, much more experimental work can be done to gain more insight into how the performance of this kind of model can be optimised. Examples of experiments that can be done to achieve this include: -

- Investigation into how well the models behave when presented with data that is not identical to the data used to generate the model. (i.e. so called “out-sample” testing) This idea can also be extended further to see how gracefully the model’s behaviour decays when the data presented to it falls outside of the range of the training data.
- Investigation of how sparsely the training data can cover the possible view space of an object before the performance of the model falls below an acceptable level.
- Investigation that yields qualitative (or preferably quantitative) results concerning how different the objects being modelled need to be in order to minimise the occurrences of false identification. This work could also be extended to explore what secondary characteristics should be modelled in order to allow distinction in cases where silhouette information is not enough on its own.

Certainly some if not all of the above-mentioned investigations need to be undertaken before any evidence is produced that has the potential to shed light on whether or not the human visual system functions by statistically analysing visual data.

8 Conclusions

The goal of the piece of work described in this thesis was to ascertain if it is valid to use new model building techniques involving the automatic generation of two-dimensional point distribution models, for image interpretation tasks where the objects being viewed are much less complicated than have hitherto been tried.

This goal was achieved by implementing a rational reconstruction of an image interpretation system that generated and used a point description model to recognise complex bodies, and then to use this system to build and test the function of models describing simple convex prismatic objects.

The system chosen as the basis for this work was a pedestrian tracking system that was created by Baumberg for his PhD thesis. This work has two main parts. The first of these is the portion of the system that is responsible for creating the model of pedestrians. The model builder functioned by taking live raw images from a static surveillance camera and automatically extracting point descriptions of any pedestrian-sized moving objects within the scene. This data was then statistically analysed using principal component analysis to create a model describing how a silhouette of a pedestrian is allowed to vary.

The second part of Baumberg's system used the resulting model to track pedestrians through a real life outdoor scene. The image data that was to be analysed was simply real-time video footage of a typical street scene.

Given the time constraints of a MSc project it was unfeasible to implement an exact reconstruction of the pedestrian tracker done by Baumberg. Therefore the system implemented here was simplified by assuming that raw image segmentation was possible and that the input data, into both the object building part of the system and the tracker, was already available in the form of outline descriptions of the silhouettes of simple three-dimensional objects.

The outline descriptions of the simple three-dimensional objects were created using a third party C program. A re-pointing algorithm was then used to change the input shape descriptions to point descriptions similar to those that Baumberg constructed from raw images. These point descriptions were then normalised and statistically analysed, using the same methods as Baumberg, to produce an object model.

The resulting object models were then tested to see how well they could recognise input shapes. This was achieved by presenting the models with the same images that were used to train them in the first place (i.e. in sample testing), and gaining a measure of how accurately the models could classify these shapes.

It was found that the method Baumberg used to automatically generate point-descriptions of object outlines did not produce descriptions in which the positions of the points move smoothly as the silhouette of the objects change. This in turn caused the models constructed by statistically analysing the movement of these points to be unstable and to fail to systematically recognise shapes even though those shapes were part of the training set used to create the model in the first place.

In addition, preliminary evidence was found that suggested that, even if a method can be found that will automatically generate stable object outline point-descriptions, simple convex prismatic objects do not display enough variation in outline for this kind of model to be systematically effective at distinguishing between similar objects.

Further possible exploration that extends the work done here includes investigation into a method for automatically generating point descriptions of object outlines, that will allow this kind of system to function correctly in a boarder range of situations than is as yet possible. In addition, much more experimental work can be done to gain more insight into how the performance of point distribution models describing simple rigid bodies can be optimised.

References in order of citation

[1] Baumberg A. “Learning deformable models for tracking human motion”, *PhD Thesis, University of Leeds*, 1995.

[2] Johnson N. “Learning object behaviour models”, *PhD Thesis, University Of Leeds*, 1998.

[3] Haykin, S. "*Neural Networks: A Comprehensive Foundation*" Macmillan, 1994.

[4] Johansson G. “Visual motion perception”, *Scientific American*, 73-88, June 1975.

[5] Cliff, D & Noble, J. “Knowledge-based vision and simple visual machines”, *Phil. Trans. R. Soc. Lond. B*, **352**, 1165-1175, 1997.

[6] Taylor, C.J., Cootes, T.F., Lanitis, A., Edwards, G., Smyth, P. & Kotcheff, A.C.W. “Model-based interpretation of complex and variable images. *Phil. Trans. R. Soc. Lond. B*, **352**, 1267-1274, 1997.

[7] Cootes, T.F., Hill, A., Taylor, C.J. & Lanitis, A. “Active shape models: evaluation of a multi-resolution method for improving image search. *5th British Machine Vision Conference*, 372-336, 1994.

[8] Robinson, D. “Differential geometry as an approach to automatic landmark point generation” *MSc Thesis, University Of Manchester*, 1996.

[9] Lanitis, A., Cootes, A. F. & Taylor, C.J. “A unified approach to coding and interpreting face images.” *5th Int. Conf. On Computer Vision*, 368-373, 1996.

[10] http://www.scs.leeds.ac.uk/imv/index.html

[11] http://www.cvg.cs.reading.ac.uk/

[12] Hogg, D. C. “Model based vision. A program to see a walking person” *Image Vision Computing.* **1**(1), 5-20. 1983.

[13] Gower, J. C. “Generalised Procrustes Analysis”. *Psycometrika* (40):33-51, 1975.

[14] Gonzalez, R. & Woods, R. “*Digital Image Processing”*. Addison-Wesley Publishing Co., 1992.

[15] Gelb, A., editor. “*Applied Optimal Estimation.”* MIT Press. 1974.

[16] Joseph, P. D. “*Peter D. Joseph's Home Page Containing Material on Chess and on Kalman Filters*”, http://ourworld.compuserve.com/homepages/PDJoseph/

[17] *Jama: A Java matrix package.* http://math.nist.gov/javanumerics/jama/

Bibliography of unquoted references.

[1] Bruce, V., Green, P., & Georgeson, M., *"Visual Perception: Physiology, Psychology, and Ecology"*, Psychology Press. 1996.

[2] Chatfield, C, & Collins, A., *"Introduction to Multivariate Analysis"*, Chapman and Hall, 1980.

[3] Cootes, T.F. & Taylor, C. J., "Using Grey-Level Models to Improve Active Shape Model Search", *IEEE*, 1994.

[4] Heap, T. & Hogg, D., "Improving Specificity in PDMs Using a Hierarchical Approach", *University of Leeds.*

[5] Johnson, N. & Hogg, D., "Learning the Distribution of Object Trajectories for Event Recognition", *University of Leeds*

[6] Johnson, N., Galata, A. & Hogg, D., "The Acquisition and Use of Interaction Behaviour Models", *University of Leeds.*

[7] Koosis, D. *"Statistics: A Self teaching Guide"*, John Wiley & Sons, 1985.

[8] Lipschutz, S., *"Linear Algebra"*, McGraw-Hill Book Company, 1968.

[9] Vetterling, W., Press, W., Teukolsky, A. & Flannery, B. *"Numerical Recipes in C"*, Cambridge University Press, 1998.