

# Learning Deformable Shape Models for Object Tracking

by

*Anthony James Heap*

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



The University of Leeds  
School of Computer Studies

September 1997

The candidate confirms that the work submitted is his own and the appropriate credit has been given where reference has been made to the work of others.

# Abstract

The use of computer vision to locate or track objects in images has applications in a diversity of domains. It is generally recognised that the analysis of objects of interest is eased significantly by making use of *models* of objects. In many cases, the strongest visual feature of an object is its shape. Also, many objects of interest are non-rigid, or have a non-rigid appearance with respect to a particular viewpoint. For these reasons, there is much interest in the construction of, and tracking with, *deformable shape models*.

A common approach to building such a model is to apply statistics to a set of real-life training examples of an object in order to *learn* shape and deformation characteristics. Such methods have proved successful in many specific applications; however, they can experience inadequacies in the *general* case. For example, objects which exhibit non-linear deformations give rise to models which are not *compact* and not *specific*: in the process of capturing the range of valid shapes, *invalid* shapes also become incorporated into the model. This effect is particularly pronounced when building models from automatically-gathered training data. Also, in tracking, smooth movement and deformation is generally assumed, but is not always the case: the apparent shape of an object can change discontinuously over time due to, for example, rotations in 3D.

The work in this thesis addresses the above problems.

Two extensions to current statistical methods are described. The first makes use of polar coordinates to improve the modelling of objects which bend or pivot. The second uses a hierarchical approach to model more general complex deformations; non-linearities are broken down into smaller linear pieces in order to improve model specificity. In particular, this greatly improves the modelling of objects from automatically-gathered training data.

A new approach to tracking which complements the latter of these models is also described. Learned object shape dynamics are combined with stochastic tracking to produce a system which can track from automatically-generated models, as well as being able to handle discontinuous shape changes.

Examples are given of the use of these techniques, predominantly in the domain of *hand tracking*. In particular, it is shown how it is possible to track 3D objects purely from 2D models of their silhouettes.

Also described is the construction of 3D deformable models, and the use of such models in tracking. This approach eases the task of object pose inference, but is much less robust than the 2D approach.

# Acknowledgements

I would like to thank John Ridgeway and St James' Hospital in Leeds for help in the acquisition of Magnetic Resonance images, and also Andreas Lanitis and the Wolfson Image Analysis Unit, University of Manchester, for kind provision of hand model training data.

I would also like to thank my colleagues in the School of Computer Studies Vision Group at the University of Leeds, for three years' supply of friendly, productive working environment and much stimulating discussion, and particularly David Hogg for expert supervision, including help, ideas, proofreading and a fair modicum of enlightenment.

Finally, I would like to thank Lucy for a plentiful supply of perception and perspective.

# Declarations

Some parts of the work presented in this thesis have been, or are due to be, published in the following articles:

**Heap, A. J. and Hogg, D. C.**, “Extending the Point Distribution Model using Polar Coordinates”, *Image & Vision Computing*, 14 (1996) 589–599.

**Heap, A. J. and Hogg, D. C.**, “3D Deformable Hand Models”, *Progress in Gestural Interaction* (Proceedings of Gesture Workshop, York, April 1996)

**Heap, A. J. and Hogg, D. C.**, “Towards 3D Hand Tracking using a Deformable Model”, *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, Killington, October 1996.

**Heap, A. J. and Hogg, D. C.**, “Improving Specificity in PDMs using a Hierarchical Approach”, to appear in *Image & Vision Computing* during 1998.

**Heap, A. J. and Hogg, D. C.**, “Wormholes in Shape Space: Tracking through Discontinuous Changes in Shape”, to appear in *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, January 1998.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Domain . . . . .	1
1.1.1	Models of Objects . . . . .	2
1.1.2	Deformable Shape Models . . . . .	3
1.2	Focus of This Work . . . . .	4
1.2.1	Overview of Thesis . . . . .	5
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Overview of Previous Work . . . . .	7
2.1.1	Deformable Shape Models . . . . .	7
2.1.2	Tracking Paradigms . . . . .	10
2.1.3	Hand Tracking . . . . .	13
2.2	Relevant Techniques Studied in Depth . . . . .	14
2.2.1	The Point Distribution Model (PDM) and Derivatives . . . . .	14
2.2.1.1	Limitations of the PDM . . . . .	15
2.2.1.2	Adaptations to the PDM . . . . .	16
2.2.2	Shape Space Constraint Surfaces . . . . .	18
2.2.3	Active Shape Models . . . . .	19
2.2.4	The CONDENSATION Algorithm . . . . .	21
<b>3</b>	<b>Shape Models for Objects that Pivot</b>	<b>24</b>
3.1	Introduction . . . . .	24
3.2	The Cartesian-Polar Hybrid PDM . . . . .	25
3.2.1	Removing Polar Bias . . . . .	27
3.2.2	Coping with the Angle Discontinuity Problem . . . . .	27
3.3	Designing a Mapping . . . . .	28
3.3.1	The ‘Compacter’ Mapping Algorithm . . . . .	28
3.3.2	The ‘Annotator’ Mapping Algorithm . . . . .	30

3.3.2.1	Finding Rigid Sets of Landmarks . . . . .	30
3.3.2.2	Finding Potential Pivots Between Sets . . . . .	31
3.3.2.3	Constructing the Pivotal Structure . . . . .	34
3.4	Results . . . . .	34
3.4.1	Comparing the Different Modelling Techniques . . . . .	34
3.4.2	Comparing Mapping Generation Algorithms . . . . .	37
3.5	Tracking . . . . .	40
3.6	Extension to 3D . . . . .	41
3.7	Conclusions . . . . .	42
<b>4</b>	<b>Hierarchical Shape Models</b>	<b>44</b>
4.1	Introduction . . . . .	44
4.2	Constraints in Shape Space . . . . .	46
4.3	A Hierarchical PDM . . . . .	47
4.3.1	Implementation Details . . . . .	48
4.3.2	Using the Hierarchical PDM . . . . .	49
4.4	Evaluation . . . . .	52
4.4.1	Synthetic Anglepoise Lamp . . . . .	52
4.4.2	Other Synthetic Examples . . . . .	58
4.4.3	Manually Collected Real Data . . . . .	59
4.4.4	Automatically Collected Real Data . . . . .	61
4.5	Tracking using HPDMs . . . . .	64
4.6	Conclusions . . . . .	66
<b>5</b>	<b>Models of Shape Dynamics</b>	<b>68</b>
5.1	Introduction . . . . .	68
5.2	Modelling Discontinuous Changes in Shape . . . . .	70
5.3	Tracking . . . . .	71
5.4	Evaluation . . . . .	74
5.5	Improving CONDENSATION . . . . .	77
5.6	Conclusions . . . . .	79
<b>6</b>	<b>3D Shape Models</b>	<b>81</b>
6.1	Introduction . . . . .	81
6.2	3D Model Construction . . . . .	82
6.2.1	Training Image Acquisition . . . . .	82
6.2.2	Training Mesh Capture . . . . .	84

6.2.3	Physically-Based Models . . . . .	85
6.2.4	Simplex Meshes . . . . .	86
6.2.5	Initial Mesh Construction . . . . .	87
6.2.6	Mesh Deformation . . . . .	87
6.2.7	Example: 3D Hand Model . . . . .	89
6.3	Tracking . . . . .	90
6.3.1	Gathering image evidence . . . . .	92
6.3.2	Updating the model position . . . . .	93
6.3.3	Handling Self-occlusion . . . . .	96
6.4	Evaluation . . . . .	97
6.5	Extensions . . . . .	100
6.5.1	Stereo . . . . .	100
6.5.2	Polar Coordinates . . . . .	102
6.6	Discussion and Conclusions . . . . .	104
<b>7</b>	<b>Conclusions</b>	<b>106</b>
7.1	Summary of Work . . . . .	106
7.2	Discussion . . . . .	107
7.3	Future Work . . . . .	108

# List of Figures

1.1	Examples of inadequacies in existing models of deformation. . . . .	5
2.1	Training a Point Distribution Model. . . . .	15
2.2	Learning a non-linear 1-dimensional surface . . . . .	19
2.3	Generating suggested movements for landmarks . . . . .	20
2.4	Propagation of a shape population with only one shape parameter using the CONDENSATION algorithm. . . . .	22
3.1	Landmark point mappings used for an anglepoise lamp . . . . .	26
3.2	Illustration of terms used in the ‘annotator’ algorithm. . . . .	32
3.3	Modes of variation for an anglepoise lamp model under the standard PDM. . . . .	35
3.4	Modes of variation for an anglepoise lamp model under the quadratic PDM. . . . .	35
3.5	Modes of variation for an anglepoise lamp model under the hybrid PDM. . . . .	35
3.6	Graph showing how variation is accounted for under the standard, quadratic, cubic and hybrid PDMs. . . . .	36
3.7	Modelling a golf swing. . . . .	37
3.8	Automatic mapping generation for a hand model using the ‘com- pacter’ algorithm. . . . .	38
3.9	Automatic pivot generation for a hand model using the ‘annotator’ algorithm. . . . .	38
3.10	A synthetic model of an object with a single pivot. . . . .	39
3.11	Pivot location accuracy under varying conditions. . . . .	40
3.12	Cylindrical and spherical polar coordinates for the 3D Hybrid PDM. . . . .	42
3.13	First mode of variation for a 3D Hybrid PDM of a human hand. . . . .	42
4.1	Example training data for a three gesture hand model. . . . .	44



4.2	Projections of the ‘three gesture’ training data in PDM shape space. .	45
4.3	Example shapes produced by a standard PDM on the ‘three gesture’ model. . . . .	45
4.4	Constraining shape using hyperplanes and hyperellipsoids. . . . .	48
4.5	Constraining a general point to lie within a hyperellipsoid-bounded region. . . . .	50
4.6	The valid shape regions produced under two different constraint al- gorithms. . . . .	52
4.7	Three examples from the synthetic anglepoise lamp training set. . . .	53
4.8	Two views of the lamp model training data transformed into global PCA space. . . . .	53
4.9	The three most significant modes of variation of the <i>linear</i> lamp PDM. 53	
4.10	The lamp model global PCA space (2D projection), showing training data and principal component axes for the constraint patches. . . . .	54
4.11	Three traversals through the VSR for the lamp HPDM. . . . .	54
4.12	Accuracy and specificity error graphs for anglepoise lamp HPDMs with varying numbers of linear patches and degrees of overlap. . . . .	56
4.13	Accuracy and specificity error graphs for anglepoise lamp models built under the various modelling techniques. . . . .	56
4.14	Effect of training set size on the accuracy of the lamp HPDM. . . . .	57
4.15	Comparison of the HPDM with constraint surface models. . . . .	58
4.16	Manually annotated training examples for a hand model. . . . .	59
4.17	The manually annotated hand training data in global PCA space and the HPDM linear patches. . . . .	60
4.18	Modes of variation for the manually annotated hand PDM and equiv- alent HPDM shape space traversals. . . . .	60
4.19	Maximally non-valid shapes for the manually-annotated hand model. 60	
4.20	Automatically collected training examples for a hand model. . . . .	61
4.21	Several projections of the automatically collected hand training data in global PCA space and HPDM linear patches. . . . .	61
4.22	Modes of variation for the automatically trained hand PDM and equivalent HPDM shape space traversals. . . . .	62
4.23	Maximally non-valid shapes for the manually-annotated hand model. 63	
4.24	Graph showing how the specificity of the hand HPDM compares with a linear PDM, and how it varies with the number of clusters used. . .	63
4.25	Tracking examples using manually-trained hand models. . . . .	64

4.26	Tracking examples using the automatically-trained HPDM of the hand.	65
4.27	Tracking examples using the automatically-trained linear PDM of the hand. . . . .	66
5.1	Discontinuous changes in object boundary shape due to deformation and rotation. . . . .	69
5.2	A typical transition matrix. . . . .	71
5.3	The automatically captured training examples (projected into the two principal dimensions of global PCA space) and the HPDM linear patch principal axes. . . . .	74
5.4	Some example conditional pdfs generated from single seeds via both the (untrained) Fokker-Planck and Markov model algorithms. . . . .	75
5.5	Graph showing the fitness scores for the different tracking algorithms on an image sequence containing sudden shape changes. . . . .	76
5.6	Coping with a sudden change. . . . .	77
5.7	Propagation of a 1D shape population using the hybrid tracker. . . . .	78
5.8	Graphs comparing the hybrid tracker to a CONDENSATION tracker and an Active Shape Model tracker, on moderate and high speed hand movements. . . . .	79
6.1	Slices from a Magnetic Resonance scan of a human hand. . . . .	83
6.2	The Simplex Angle. . . . .	86
6.3	Deforming a Simplex Mesh from an initially random position to fit MRI data of a human hand. . . . .	90
6.4	Deforming the first model to fit a second training image. . . . .	90
6.5	The first and second modes of variation of the 3D hand PDM. . . . .	91
6.6	Suggested landmark movements. . . . .	93
6.7	Self occlusion of a vertex by a facet. . . . .	97
6.8	Snapshots from hand tracking experiments using the 3D PDM. . . . .	98
6.9	Tracking against a cluttered background. . . . .	100
6.10	The stereo tracking environment. . . . .	101
6.11	Calibration image for the stereo environment. . . . .	101
6.12	Snapshots from a 3D stereo tracking sequence. . . . .	103
6.13	Three views of the line drawing output from the 3D stereo tracker sequence. . . . .	104

# List of Tables

3.1 Machine cycles used to generate a model instance under different types of PDM. . . . . 37

# Chapter 1

## Introduction

---

### 1.1 Problem Domain

The use of computer vision to locate or track objects in images has applications in a diversity of domains.

In some cases the goal is to assist humans in tedious or repetitive visual tasks. In surveillance, human body tracking warns night watchmen of potential intruders; in manufacturing, electronic component location is used to spot defects in printed circuit boards on the production line.

Alternatively, there may be a need for accurate measurement or classification. In medicine, x-rays of children's hands are analysed to spot defects in bone growth; in sports, whole-body tracking is used to analyse and improve the technique of javelin throwers; in robotics, feedback from vision guides autonomous vehicles around obstacles; in animal welfare, livestock tracking is used to monitor behaviour under varying living conditions.

Otherwise, the application might simply be to improve the interaction between man and machine. Hand tracking is the first step towards gesture analysis and the interpretation of sign language; gaze analysis can be used to determine the user's point of focus on-screen.

In all these cases it is generally recognised that analysis of the objects of interest is eased significantly, especially in the presence of noise or background clutter, by

making use of *a priori* knowledge of object shape and appearance. In short, we require *models* of objects.

### 1.1.1 Models of Objects

“If we are to identify objects, we must have some *a priori* representation for those objects.” Rodney Brooks, 1981 [13].

A model of an object is merely a description of its features. This very general statement hides a number of underlying questions. Firstly, which features are to be modelled? In the context of computer vision these should certainly be *visual* features, such as surfaces, edges, corners or distinctive markings. Secondly, which *attributes* of these features are to be described? Shape, colour or texture are all possibilities. Thirdly, what is the nature of these descriptions? For models of rigid objects, a single description of appearance is required. For models of classes of objects, deformable objects or 2D projections of 3D objects, there might be either a continuous range or discrete set of valid appearances. For models of dynamic objects, a description of how appearance changes over time may also be necessary.

Only after these questions have been answered can an object model be constructed (i.e. have its features described). Simple models can be hand-crafted, with the descriptions of appearance, deformation and dynamics all being hard-coded. However, for more complex models, and also to provide a general framework, it is useful to look to *learning* models from real-world examples.

The very fact that an object model is being constructed as an aid to object location or tracking seems to negate the possibility of making object feature measurements automatically from suitable images. However, it is sometimes possible to automate or semi-automate the process by providing object examples under constrained conditions. For instance, it is very easy to locate the *boundary* of an object placed against a homogeneous background.

It is perhaps interesting to note that in order to automate fully the construction of object models, one must automate the design decisions (i.e. the questions posed above) as well as the measurement process. Otherwise one can only ever build models of objects which lend themselves to the particular chosen features and attributes.

Models of object shape and appearance are used for object *location* in images by performing a match between model and image. Given a set of parameters describing the model state (e.g. position, orientation or size), the problem reduces to one of finding values for these parameters which result in the best match. If we select image

features and look for matching model features we are said to be using a *bottom-up* approach. Conversely, if we select model features and search for matching image features we are taking a *top-down* approach.

At the simplest level, object *tracking* can be thought of as repeated object location. However, it would be foolish to ignore the object pose constraints which exist from frame to frame in an image sequence: models of object *dynamics* can be used. The most basic dynamic constraint is proximity; given a sufficient sample rate, an object's appearance in one frame will be very similar in the next frame. Additionally, any object with mass will experience momentum and inertia, so velocity constraints can be used. Higher order assumptions can also be made; for example, acceleration models can predict oscillatory behaviour, but beyond that they become less powerful.

Although certain object models lend themselves to particular tracking paradigms, it is useful to draw a distinction between *models* and *tracking*. Tracking should be thought of as an *application* of a model; there are other equally worthy applications such as simulation or visualisation. Also, it is certainly possible for two different tracking algorithms to make use of the same underlying object model—indeed such trackers could possibly be combined to produce a hybrid tracker which exhibits the best qualities of the two individuals.

Two of the most important factors when considering modelling or tracking techniques are performance and reliability. Most of the applications outlined above require real-time tracking, so any successful algorithm must be able to process several frames per second on technology that is currently available or will be available within, say, the next ten years. Also, a system should be robust enough such that tracking failures rarely occur: humans have a very low tolerance for such failures. Achieving either real-time performance or reliability in isolation is relatively simple; achieving both together is much harder.

### 1.1.2 Deformable Shape Models

In many cases, the strongest visual features of an object are its contours (edges), especially those that divide it from its surroundings (i.e. its boundary), and the most useful, if not only, attribute of a contour is its shape. Also, many objects of interest are either non-rigid, or else exhibit non-rigid changes in appearance with respect to a particular viewpoint. For these reasons there has been much work published on the subject of models of shape and deformation. The features used in such

work are almost exclusively contours, allowing the modelling of a very wide class of objects (i.e. any object that has a reasonably distinctive boundary). Tracking with such models is also generally very successful - edge features are easy to locate in images, and a top-down approach with sparse image analysis can be used, resulting in high-speed systems which are relatively robust.

One of the most successful approaches to modelling shape and deformation has been to use training data to *learn* an object's shape and how it can vary. This generally gives realistic, compact models which perform well in many tracking tasks.

## 1.2 Focus of This Work

Although the use of deformable shape models has been demonstrated to be successful when applied to certain specific examples, current models of deformation and dynamics still experience inadequacies in the general case. Some common problems are:

- Models of deformation are not optimally compact, often representing non-linear deformations as a combination of two or more linear ones (Figure 1.1a).
- Subsequently, non-valid shapes can be generated via inappropriate combinations of these linear deformations — the model is not *specific* (Figure 1.1b).
- Automatically gathered training data can produce very poor quality models (Figure 1.1c).
- Smooth, continuous dynamics are generally assumed. However, in some cases, *sudden* shape changes can occur. When tracking a silhouette this can be due to object deformations, or changes in view of a 3D object (Figure 1.1d).

The work in this thesis addresses the above problems. New techniques are developed which improve the specificity of models, and an approach to tracking is described which complements these models and caters for a wide array of dynamic behaviours, including non-smooth deformation. The development of 3D object models for tracking from 2D images is also detailed, and their relative merits are discussed.

Throughout this work, human hands have been used as the main case study for the presented modelling and tracking techniques, with the rationale that the ability to locate hands and recognise gestures would be of great practical use, for example

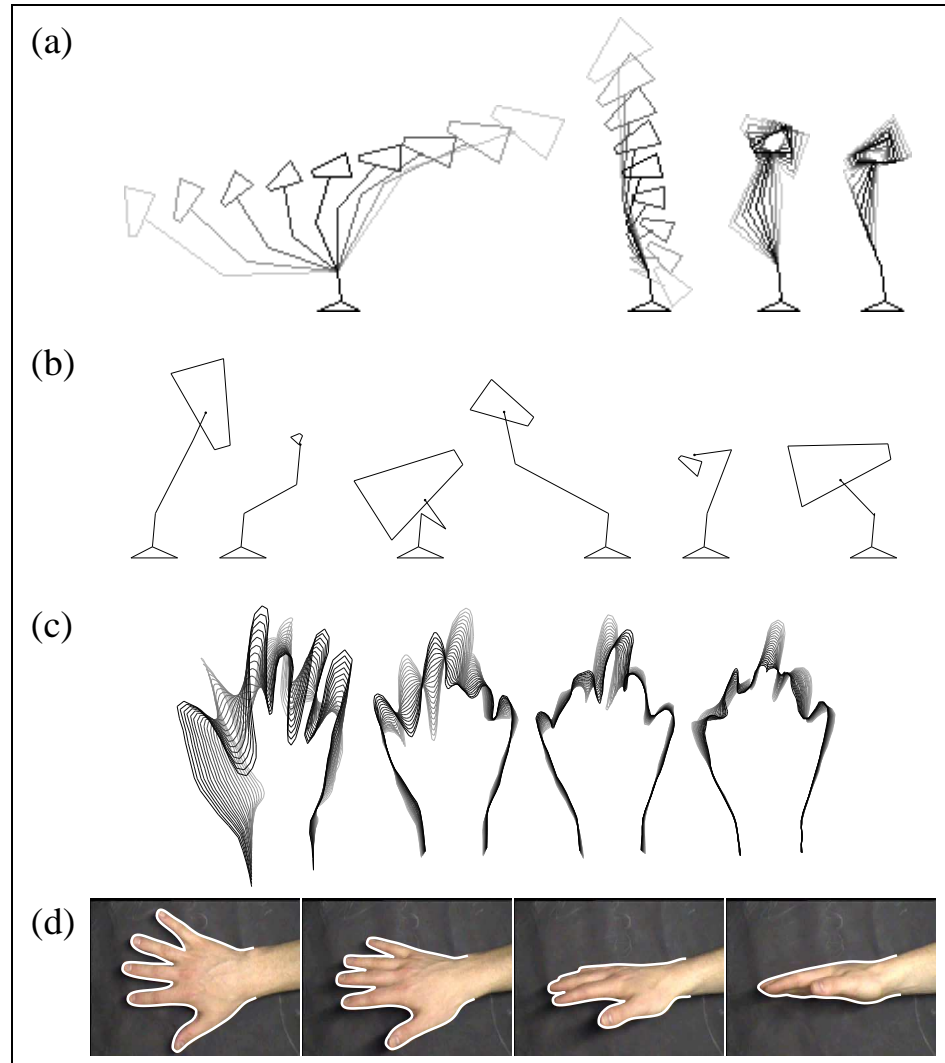


Figure 1.1: Examples of inadequacies in existing models of deformation: (a) non-compact linear deformation modes, (b) subsequent non-valid shapes, (c) a model generated from automatically-gathered training data and (d) a discontinuous shape change.

in human-computer interaction and also because it is a difficult and challenging task. The methods described also apply generally to other objects.

### 1.2.1 Overview of Thesis

The thesis is set out as follows: this chapter has been a general introduction and overview of the subject; Chapter 2 is a review of background material; the remainder constitutes the original work and is organised in the following way:



- Chapter 3 describes an extension to current statistical models: polar coordinates are used to improve the modelling of objects which bend or pivot.
- Chapter 4 describes an alternative extension which uses a hierarchical approach to model more general non-linear deformation. This improves model *specificity*, and also enables construction of models from automatically gathered training data.
- Chapter 5 describes a new approach to tracking: a model of object shape dynamics is combined with stochastic tracking to produce a system which can cope with a wide array of dynamic behaviours, including discontinuous shape changes. This facilitates the tracking of 3D objects from silhouettes and also tracking from automatically generated shape models.
- Chapter 6 describes the construction of 3D deformable models and the use of such models in tracking from 2D images. The relative merits and drawbacks compared with 2D models are discussed.

Finally, some general conclusions are drawn, and there is a discussion of potential future work.

# Chapter 2

## Background

---

This chapter is divided into two sections. Firstly, there is an overview of previous work on deformable shape models and their use in tracking. Secondly, there is a more in-depth study of techniques upon which the work in this thesis is based.

### 2.1 Overview of Previous Work

#### 2.1.1 Deformable Shape Models

Deformable shape models date back at least to 1965 and the work of Roberts [64], in which simple geometric primitives such as cubes and prisms are permitted to stretch and skew in order to explain observed image features.

Many other authors make use of hand-crafted deformable shape models. In the ACRONYM [13] system of Brooks, models are built from sets of generalised cones with variable shape parameters, joined together via translation and orientation constraints. Hogg's 'Walker' system [39] uses a 3D articulated 'tin-can' model for tracking human motion, and Rehg and Kanade take a very similar approach towards hand modelling [62]. Both systems incorporate a complex set of hard-coded joint constraints. Lowe uses a similar system but for more general objects [52]. Yuille describes *deformable templates* [75]: parameterised deformable shape models, constructed from geometric primitives such as circles and parabolas. Prior probabilities are also defined for each parameter to indicate the possible range of values.

All of the above modelling techniques require the investment of time and expertise to produce a model. Deformable models require much more thought than rigid models because a whole range of possible shapes must be described, preferably in terms of a small number of parameters. In some cases, there are dependencies amongst these parameters which must also be modelled (e.g. inter-dependent finger joint angles). Additionally, the approximation of object shape by geometric primitives, such as cylinders, lines, boxes or simple mathematical curves, is limited in its expressiveness. For example, biological objects can rarely be effectively modelled in this way.

The *snake* [45] of Kass *et al* provides a more appropriate framework for modelling biological shapes. A snake is a flexible contour (represented by a set of control points) which behaves like an elastic stick with internal physical properties. It can take on almost any form, but it has a potential energy term which rises as it deforms away from the rest shape. Snakes have no *a priori* knowledge of object shape; the only device by which they may be customised to a particular shape is by relaxing or tightening the stiffness constraints at particular nodes.

Deformable meshes are the 3D equivalent to snakes. They are surface meshes consisting of a number of nodes, each connected via virtual springs to a number of neighbouring nodes. Delingette [25] and Bulpitt and Efford [14] describe two such meshes.

Curwen and Blake introduce *coupled contours* [24]—snakes coupled to fixed-shape templates—to allow models to have a specific preferred rest state. Additionally, they represent the contour as a cubic B-spline; this requires fewer control points and also has implicit smoothness constraints. As with snakes, deformation is based on an assumption of elastic properties.

Grenander *et al* [30] adopt a statistical approach to incorporating *a priori* knowledge of shape into contours. Several example contour shapes are extracted manually from training images and the distributions of the angles between adjacent contour segments are learned and modelled as a Markov chain.

Much work has been published on modelling techniques which can be loosely classed as ‘base-shape-plus-linear-deformations’ approaches. These generally consist of a base shape  $\bar{\mathbf{x}}$  (usually coded as the  $(x, y)$  co-ordinates of a number of control points) and a number of linearly-independent deformations  $\mathbf{v}_1 \dots \mathbf{v}_t$  which can be added to the base shape in various proportions to produce all possible valid shapes:

$$\mathbf{x} = \bar{\mathbf{x}} + \sum_i^t b_i \mathbf{v}_i \quad (2.1)$$

The weightings  $b_i$  are linear deformation parameters and form a  $t$ -dimensional *shape space*. The advantages of such approaches are low deformational dimensionality and the use of linear (and hence fast) algebra. The differences between the approaches are really to do with the determination of  $\bar{\mathbf{x}}$  and the  $\mathbf{v}_i$ .

The simplest example of this is the set of affine-invariant deformations introduced by Blake *et al* [8]. The model incorporates three degrees of affine deformation from the base shape, allowing the modelling of *planar* objects viewed at any orientation.

Pentland and Horowitz [57] describe how it is possible to produce sensible linear deformations from a single base shape by way of the Finite Element Method (FEM). An FEM model is a physical model, treating the control points as point masses, and including stiffness and damping coefficients between every pair of points. Such models undergo free vibration when perturbed. Eigenanalysis of the FEM model components can be used to extract the *free modes of vibration*. These are used directly as the model deformation vectors. Pentland and Horowitz demonstrate the use of the technique on generic object shapes [57, 58], but it can also be applied to *specific* object shapes [66].

*Key frames* [10], described by Blake *et al*, are representative training examples of an object being modelled. Deformation modes can be constructed as interpolations from the base shape to each of the key frames, hence each key frame gives rise to one degree of freedom. Key frames must be chosen carefully and control points located with precision to ensure that a good model is produced. Ullman and Basri [73] show how key frames can also be used in some cases to reconstruct (a limited range of) unseen views of 3D objects.

The *Point Distribution Model*, or PDM [19] of Cootes *et al* (described in Section 2.2.1) combines the idea of multiple training examples with statistical analysis to produce models with good specificity. The use of *principal component analysis* (PCA) [44] on the deviations of a large number of training examples from the mean shape gives rise to a small number of deformations which are representative of the training data. The accompanying eigenvalues provide a measure of significance for each deformation mode. The main advantages over the use of key frames are that training examples do not need to be chosen quite as carefully and the location of control points need not be as accurate. Both are important factors when considering the possibility of automated training [3], which generally gives rise to large volumes of noisy training data. PDMs have found many practical applications in computer vision, including face recognition [49], person tracking [4], medical image analysis [38] and car tracking (using 3D models) [67, 74].

The similarity of the above techniques means that they can easily be combined to produce richer models. Blake *et al* combine the affine-invariant deformations with key frames in a single model [10], and Cootes and Taylor show how PDMs and FEM models can be combined to produce useful hybrid models in the case of having only a few training examples [16].

In some cases the base-shape-plus-linear-deformation approach can fail. Valid object shapes might form regions in shape space which are non-linear, cyclic, disjoint or of variable dimensionality. Bregler and Omohundro describe *constraint surfaces* [12] (see Section 2.2.2) which provide a means for learning *arbitrary* regions within high dimensional spaces from training data. They go on to show how this technique can be applied to shape modelling, using human lips as an example. Ahmad *et al* [1] describe work along similar lines whereby training shapes of human hands are manually categorised into five separate gestures and a *local* PCA is performed on each one to produce a piecewise-linear hand gesture model.

### 2.1.2 Tracking Paradigms

There has been much work on *location* of objects; this is generally treated as a global search problem and, as such, is computationally intensive. Most techniques operate on a hypothesis/validation (generate-and-test) principle; promising candidate solutions are generated and then validated via image analysis. Efficiency is highly dependent on the heuristics for the hypothesis stage. The simplest strategy is *pose sampling* (effectively an exhaustive search), but this is only feasible for very constrained search spaces. Techniques such as *alignment*, the *generalised Hough transform* and *geometric hashing* [40] all depend on the reliable location of ‘interest points’ in the image, such as edges or corners (i.e. they adopt a ‘bottom-up’ approach). This is time-consuming, and also not robust because low-level mistakes affect high-level performance. Conversely, the Genetic Algorithm approach used by Hill *et al* [34, 33] is entirely top-down: hypotheses are generated without reference to the image; image analysis is only used in the verification stage.

Object *tracking* has the added advantage of the availability of temporal information. The simple fact that object movement is generally small between consecutive frames reduces the global search problem to a local one, implying that any object location technique that is based on local optimisation is also useful for tracking.

One such technique is described by Lowe [52], using parameterised, articulated models. The error between projected model edges and nearby image edge features is

measured, and Newton non-linear least-squares minimisation is used to optimise the model parameters iteratively for this error. In Lowe's implementation, however, the speed benefits of local optimisation are cancelled out to some extent by the global computation of image edges.

The *snake* of Kass *et al* [45] is the forerunner to a whole host of work on physics-based tracking. As mentioned above, a snake is a flexible contour with certain internal stiffness properties. It tracks by being 'attracted' to various image features. The scenario is formulated in terms of energy: the image is abstracted as an energy landscape, with desirable features (usually edges) having low energy. A snake, when placed on such a landscape, locks onto features by sliding down into these energy minima whilst simultaneously minimising its internal potential energy. In practical terms, the energy gradient is evaluated (via image analysis) at a set of control points along the snake (the image first undergoes a Gaussian blur in order to widen the energy wells in the landscape) and the snake is deformed iteratively until it reaches a stable position. The whole process can alternatively be thought of in terms of force-based tracking: external gravity-like forces pull the snake downhill in the energy landscape and internal forces maintain its smoothness. This is a local optimisation process and so extends naturally from object location to object tracking. In addition, the physical properties of the snake can be extended to momentum, thus providing some form of temporal prediction. Terzopoulos and Szeliski reformulate the snake dynamics within a probabilistic framework and introduce the Kalman snake [72] (based on a Kalman filter) which, as well as predicting the snake's position, can provide confidence limits for such predictions.

The principles of snake deformation apply to 3D surface meshes in a very similar manner, with force-based tracking predominantly being used. There is the additional aspect of structural reorganisation of the mesh to cater for object topography; mesh nodes can be added (refinement) or removed (decimation) in order to provide a more even tessellation.

Curwen and Blake [24] show how snake technology can be used with B-spline contours, and also introduce a more efficient method for feature search, whereby image edges are sought along contour normals using a divide-and-conquer strategy. This avoids the need for the Gaussian blur and 2D gradient calculations. In further work [8], Blake *et al* also combine their approach with the Kalman Filter, which affords several advantages. One benefit is that the spatial search scale is controlled automatically according to certainty; if no feature is found, the search scale is increased. Also, the temporal scale (i.e. memory) is adaptive; inertia is effectively

reduced when features are lost, allowing fast recovery. When features are found, the memory is extended to exploit motion coherence.

Cootes and Taylor describe Active Shape Models (ASMs or ‘Smart Snakes’) [15, 20]: the application to tracking of the PDM. The approach is similar to Lowe’s in that image measurements are projected into the model parameter space and parameter errors are then minimised. However, in this case the minimisation is *linear* least-squares, which has a closed form solution and is thus faster to calculate. The maths involved is further simplified by the fact that the PDM’s deformation modes are orthonormal. Also, because there are generally only a few model parameters, this approach is faster than previous snake-like techniques. Performance and speed can be improved further still by employing a multi-resolution search [21] whereby earlier iterations proceed at lower image resolution and fewer shape parameters are allowed to vary, with refinement being permitted in the later stages.

Baumberg and Hogg show how ASMs can be coupled with a Kalman filtering framework to produce a more robust system [4]. This method is very efficient because the filters for each shape parameter can be decoupled, allowing independent filtering of each parameter and thus avoiding large matrix computations.

All of the approaches described so far have either non-specific or hard-coded dynamical systems. Some of them (e.g. Kalman filters) are adaptive, but none have any *learned* dynamics.

Hogg’s ‘Walker’ model [39] is an early example of a non-trivial temporal model. The kinematics are coupled to a pre-learned periodic walk sequence, modelled via a series of cubic B-splines, which is used to derive predictions for plausible object states in each successive frame.

Blake *et al* describe how dynamics can be learned from training sequences [10] within a Kalman filter framework. Dynamics up to second order (such as constant speed, oscillation or decay) can be learned using a simple tracker with default dynamics in a constrained environment. The trained tracker then improves robustness on similar motions.

Baumberg and Hogg [6] show how to construct temporal models from training sequences using FEM model analysis [57]. The models produced exhibit a number of independent modes of vibration which reflect the motions experienced in the training sequences. These motions can then be used directly as prediction models for tracking, again, within a Kalman filtering framework. The use of modal analysis means that, unlike Blake *et al*’s model, the Kalman filter can be decoupled for extra speed.

Isard and Blake’s CONDENSATION algorithm [43] (described in Section 2.2.4) provides a much richer environment for temporal prediction. The model state is represented not as a single, deterministic set of model parameters, but as a probability density function over the whole parameter space. This allows for non-Gaussian (arbitrary, in fact) uncertainty and multiple hypotheses. Propagation dynamics are learned from training sequences; Isard and Blake demonstrate the construction of second order models which can predict constant velocity, oscillatory and decaying dynamics.

### 2.1.3 Hand Tracking

As the focus application of this thesis is hand tracking, it seems appropriate to give a survey of previous work in this area. Pavlovic *et al* [56] have recently produced a very thorough review of much of this work, also including details on *gesture analysis*, which involves ascribing meaning to the movements of the hand(s).

All (noteworthy) previous work makes use of some form of model of the hand. The types of models used vary enormously but can be categorised broadly as being either full 3D models or 2D appearance models.

3D approaches have been based almost exclusively on skeletal models of the hand. These manually-crafted models are fairly accurate in that the constraints amongst joint angles and relative bone lengths are coded explicitly from biological data, but in each case a ‘generic’ hand model is created: the differences from person to person are not incorporated. They are also very tedious and time-consuming to build. Tracking with such models involves collecting pieces of image evidence and calculating changes to the hand position and joint angles which best satisfy the evidence. Both Dorner [27] and Lee and Kunii [50] simplify the image analysis task by using colour-coded gloves. Rehg and Kanade’s *DigitEyes* [62], however, can track unadorned hands; the skeletal model is appended with jointed cylinders, and edge detection is used to drive the deformation process. In later work by the same authors [63] the subject of self-occlusion is discussed.

Systems using 2D appearance models have been more varied. Deformable contour-based approaches are popular [46, 1, 9]; these use either key frame or statistical shape models, and tracking proceeds in a snake-like manner, with various adaptations as described in Section 2.1.2.

Silhouette shape models have also been used extensively [47, 54, 23, 29, 60, 53]. These generally involve a fixed number of static hand poses, indexed into a ‘gesture



library' via some set of measured features, such as silhouette width, height, centroid position, first and second order moments, or even eigenimage decomposition. Under such systems, hands are not so much tracked through video sequences as *located* in every frame. A bottom-up approach is used; background subtraction, image thresholding or colour segmentation techniques are applied to segment the hand silhouette, which is then compared to the library to find a match. This comparison in itself can be non-trivial; Kjeldsen, for example, makes use of a neural network [47] for matching. Such approaches can be very susceptible to background noise and changes in hand scale or orientation.

## 2.2 Relevant Techniques Studied in Depth

### 2.2.1 The Point Distribution Model (PDM) and Derivatives

A PDM is a model of shape built purely from the statistical analysis of a number of examples of the object to be modelled [19]. Given a collection of training images of an object, the Cartesian coordinates of  $N$  strategically-chosen landmark points are recorded for each image. Training example  $e$  is represented by a vector  $\mathbf{x}_e = (x_{e1}, y_{e1}, \dots, x_{eN}, y_{eN})$  (for a 2D model).

The examples are aligned (translated, rotated and scaled) using a weighted least-squares algorithm, and the mean shape  $\bar{\mathbf{x}}$  is calculated by finding the mean position of each landmark point. The modes of variation are found using principal component analysis (PCA) [44] on the deviations of examples from the mean, and are represented by  $N$  orthonormal 'variation vectors'  $\mathbf{p}_1 \dots \mathbf{p}_N$ . A deformed shape  $\mathbf{x}$  is generated by adding linear combinations of the  $t$  most significant variation vectors to the mean shape:

$$\mathbf{x} = \bar{\mathbf{x}} + \sum_{j=1}^t b_j \mathbf{p}_j \quad (2.2)$$

where  $b_j$  is the weighting for the  $j^{\text{th}}$  variation vector. Figure 2.1 illustrates the training procedure.

Generally, the significant deformations are captured by only a handful of variation vectors; the rest represent noise in the training data. By choosing  $t \ll 2N$ , we extract only the important deformations, discarding noise, and thus we can compactly capture object shape and variation.

An instance  $\mathbf{X}$  of the model can then be generated in the image frame by speci-

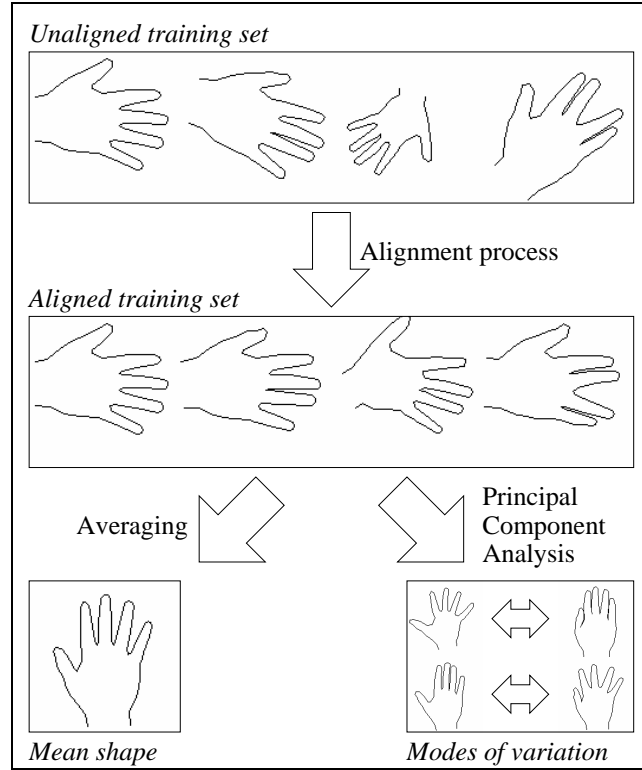


Figure 2.1: Training a Point Distribution Model.

fying translation, scale and orientation:

$$\mathbf{X} = M(s, \theta)[\mathbf{x}] + \mathbf{X}_c \quad (2.3)$$

where  $M(s, \theta)[\cdot]$  performs a rotation by angle  $\theta$  and a scaling by factor  $s$ , and  $\mathbf{X}_c = (X_c, Y_c, X_c, Y_c, \dots, X_c, Y_c)$ , where  $(X_c, Y_c)^T$  are the coordinates of the centroid of the model instance.

The PDM affords several advantages over other shape modelling techniques. The use of real-life training data makes for an accurate model of shape. Because a whole *set* of these training examples is used, a realistic model of deformation is also obtained, and without the need for hand-crafting. The statistical nature of the approach means that a degree of noise in training data annotation is tolerable; this noise is filtered by the PCA.

### 2.2.1.1 Limitations of the PDM

A good deformable model should be *accurate*, *specific* and *compact*. An accurate model includes all valid shapes. A specific model excludes all invalid shapes. A compact model uses the smallest number of parameters possible to describe a shape

(i.e. its dimensionality approaches the natural deformational dimensionality of the object being modelled).

Model shape can be thought of in terms of points in an  $n$ -dimensional *shape space*. In the case of a PDM the dimensions are the  $x$  and  $y$  coordinates of every landmark point (so there are  $2N$  dimensions). Within the shape space there is generally a continuous region which corresponds to *valid* shapes; in this thesis this is referred to as the valid shape region, or VSR.

The standard PDM assumes that the set of all valid shapes forms a Gaussian distribution normally about some mean point in the shape space. By setting a maximum allowable Mahalanobis distance from the mean, the VSR can be thought of as being bounded by a hyperellipsoid. In some cases, especially when model landmarks have been chosen strategically, this approximation is sufficient to produce a satisfactory model which is both compact and specific. However, in many real objects, non-linear deformations (such as bending or pivoting) are a natural occurrence. The PDM is forced to model non-linear deformations by the combination of two or more linear deformations. Such models are not compact because the dimensionality is increased, and not specific because invalid shapes can be produced via an inappropriate combination of linear deformations (see Figure 1.1a for examples). This corresponds to the PDM *over*-approximating the VSR, and covering a larger region of shape space than is required. Poor accuracy is the result of the opposite problem, whereby the PDM *under*-approximates the VSR, resulting in some valid shapes not being included in the model.

### 2.2.1.2 Adaptations to the PDM

There has been some work on adaptations to the PDM to improve its accuracy, specificity or compactness.

Accuracy is often poor in the case where not enough training data has been provided; generally this results in only a subset of the true VSR being modelled (i.e. the PDM is too highly constrained). Improvements can be made by somehow allowing extra degrees of deformational freedom in the model.

Kervrann and Heitz [46] show how this is possible by way of a first-order Markov process, similar to that used by Grenander *et al* [30]. Landmark points are allowed to move slightly from their current position, but with neighbouring points being encouraged to move in unison to some extent.

Cootes and Taylor show how a similar effect can be achieved via FEM model analysis [16]; statistical and physical deformations are combined to produce a single

model, as described in Section 2.1.1. In later work they describe a simpler alternative which attempts to introduce extra variability via direct tampering with the shape covariance matrix [17].

All of these processes can be used to ‘bootstrap’ the construction of a PDM by automatically locating new training examples in unseen poses.

Poor specificity arises generally in the case where the VSR is non-linear, forcing the PDM to include non-valid shapes in the model. This also results in non-optimal compactness (see Section 2.2.1.1 above). In such cases it is sometimes possible to improve specificity and compactness by performing a non-linear mapping of the shape space into some new space, in the hope of making it more linear. If this can be done, fewer modes of variation are needed and the model is more compact. Also because non-linearities are not being modelled with two or more modes, specificity is improved.

Sozou *et al* describe one such approach, the Polynomial Regression PDM [68], in which the second and subsequent shape parameters,  $b_2 \dots b_t$ , are determined by fixed polynomial functions of the first parameter,  $b_1$ :

$$b_j = a_{j0} + a_{j1}b_1 + a_{j2}b_1^2 + \dots + a_{jd}b_1^d \quad (2 \leq j \leq t) \quad (2.4)$$

where  $d$  is the degree of the polynomial being used. The values of the  $a_{jk}$  coefficients are determined by finding polynomials of best fit through the training data. Because, in general, not all legitimate variation can be captured with only one shape parameter, it is necessary to perform this process iteratively on residual deviation from the polynomial curves (hence the term ‘regression’), until the only remaining variation is due to noise in the training data.

In later work, Sozou *et al* describe the use of a multi-layer perceptron (MLP) to perform a non-linear mapping of the shape space [69]. A 5-layer perceptron is constructed with a ‘bottleneck’ in the middle layer. In training, the MLP is forced to code training shapes in a number of dimensions equal to the number of neurons in the middle layer, thus effecting the required non-linear dimensional reduction. The process works well for one-dimensional deformations but less well in the presence of two or more modes of deformational freedom.

In the case of modelling continuous contours, an alternative to constructing a non-linear mapping is to look at the possibility of altering the training data itself. Landmarks can be moved *around* a contour without altering its overall shape. If done intelligently, this can improve model linearity. Hill and Taylor describe a *greedy* algorithm which attempts to achieve this by minimising the overall model

variance [35]. Baumberg and Hogg [5] describe the use of an alternative component analysis (i.e. *not* PCA) along with stochastic model perturbations in order to refine training examples.

### 2.2.2 Shape Space Constraint Surfaces

*Constraint surfaces* (as described by Bregler and Omohundro [12]) provide a method for learning an arbitrary *surface* within an  $n$ -dimensional space, using samples taken from it. In this context, a surface is defined to be a geometric entity — embedded within an  $n$ -dimensional space — which can have any topology and whose dimensionality can vary arbitrarily over the space (but is generally much less than  $n$ ).

The key to the approach is that a complex, non-linear surface approaches linearity locally under magnification, and hence can be approximated by a combination of a number of smaller linear ‘patches’ (i.e. a *piecewise linear* approach).

To build the linear patches, a  $k$ -means cluster analysis is performed on the training data in shape space to find a number of prototypes. For each prototype, a number of nearest neighbours are taken from the training set and a PCA is performed on them. This produces the desired lower-dimensional linear sub-space in the region of each cluster (the dimensionality of which is determined by the number of *significant* principal components, according to an arbitrary cut-off point). The whole non-linear surface is represented via a combination of these linear patches. For example, a constraint function  $C$  can be applied to a general shape  $\mathbf{x}$  as follows:

$$C(\mathbf{x}) = \frac{\sum_i G_i(\mathbf{x})P_i(\mathbf{x})}{\sum_i G_i(\mathbf{x})} \quad (2.5)$$

where  $P_i(\mathbf{x})$  is the shape  $\mathbf{x}$  as projected into the subspace of linear patch  $i$  and  $G_i$  is the *influence* function for patch  $i$ .  $G_i$  is a Gaussian and is centred on the  $k$ -means prototype for patch  $i$ , with a variance “determined by the local sample density” (no rigorous mathematics is given to explain this exactly). Figure 2.2 illustrates the learning process.

This technique applies directly to object shape modelling. If the training samples are examples of valid object shapes then the surface produced is representative of all valid shapes. Bregler and Omohundro demonstrate modelling the shape of human lips in this way [12].

Note that in building such a constraint surface there are various design choices to be made: the number of linear patches to build, how many nearest neighbours to use for each local PCA (this is related to how much the patches should overlap)

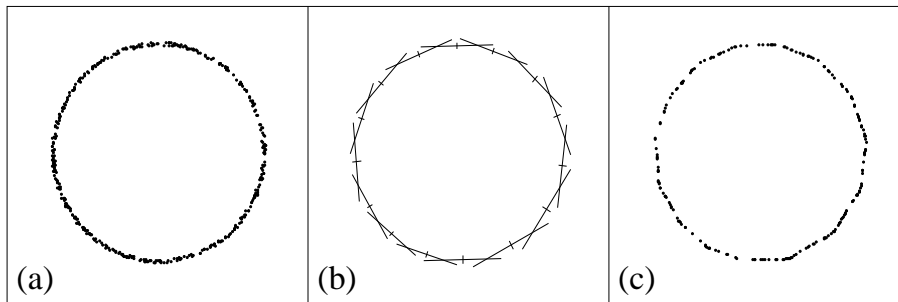


Figure 2.2: Learning a non-linear 1-dimensional surface; (a) the training data, (b) the local linear patches, (c) the learned surface.

and, crucially, the cut-off ratio for deciding the dimensionality of each patch (i.e. how significant a dimension must be for it to be included).

Constraint surface models suffer some disadvantages, mostly related to the use of non-bounded hyperplanes to represent each linear patch. This is discussed in more detail in Chapter 4.

### 2.2.3 Active Shape Models

Active Shape Models [15, 20] are the application of the PDM to object location and tracking. The approach is similar to the snake algorithm, but with the shape constraints coming from the PDM.

The starting point is a PDM instance  $\mathbf{X}$  in the image frame, defined in terms of centroid coordinates  $(X_c, Y_c)$ , orientation  $\theta$ , scale  $s$  and deformation weights  $\mathbf{b} = (b_1, \dots, b_t)^T$ :

$$\mathbf{X} = M(s, \theta)[\bar{\mathbf{x}} + \mathbf{P}\mathbf{b}] + \mathbf{X}_c \quad (2.6)$$

where  $\mathbf{X}_c = (X_c, Y_c, X_c, Y_c, \dots, X_c, Y_c)$  and  $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_t)$  is the matrix of variation vectors (this is a combination of equations 2.2 and 2.3). For each control point  $i$ , a suggested movement  $d\mathbf{X}_i = (dX_i, dY_i)$  is discovered by analysing a line of pixels normal to the model boundary and locating the strongest edge (see Figure 2.3).

The aim is not to update the positions of the control points directly, but to find changes to the model parameters  $X_c$ ,  $Y_c$ ,  $s$ ,  $\theta$  and  $\mathbf{b}$  which move the control points as close as possible to their desired locations. The changes in pose  $(dX_c, dY_c, ds$  and  $d\theta)$  are calculated as follows:

$$dX_c = \frac{1}{N} \sum_{i=1}^N dX_i \quad dY_c = \frac{1}{N} \sum_{i=1}^N dY_i \quad (2.7)$$

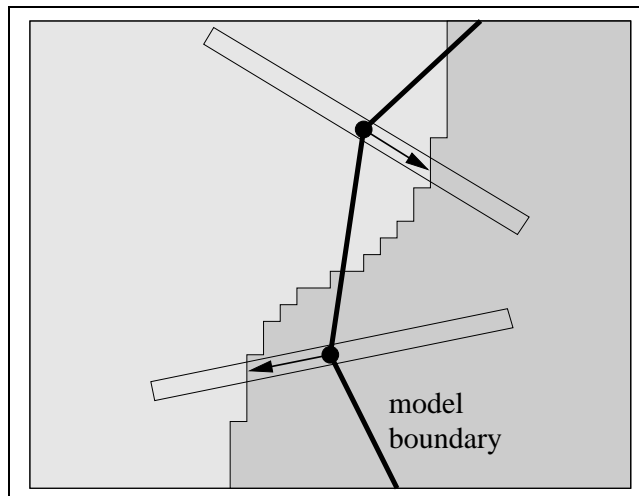


Figure 2.3: Generating suggested movements for each landmark by searching for edges along a line normal to the model boundary.

$$ds = \frac{1}{N} \sum_{i=1}^N \frac{(d\mathbf{X}_i - d\mathbf{X}_c) \cdot (\mathbf{X}_i - \mathbf{X}_c)}{\|\mathbf{X}_i - \mathbf{X}_c\|^2} \quad (2.8)$$

$$d\theta = \frac{1}{N} \sum_{i=1}^N \frac{(d\mathbf{X}_i - d\mathbf{X}_c) \cdot \mathbf{R}(\mathbf{X}_i - \mathbf{X}_c)}{\|\mathbf{X}_i - \mathbf{X}_c\|^2} \quad (2.9)$$

$$\text{where } \mathbf{R} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad (2.10)$$

The *residual* suggested movements are then projected into the model frame:

$$d\mathbf{x} = M((s(1 + ds))^{-1}, -(\theta + d\theta))[M(s, \theta)[\mathbf{x}] + d\mathbf{X} - d\mathbf{X}_c] - \mathbf{x} \quad (2.11)$$

Subsequently the change in shape weights  $d\mathbf{b}$  is given by:

$$d\mathbf{b} = \mathbf{P}^T d\mathbf{x} \quad (2.12)$$

It can be shown that equation 2.12 is equivalent to a least-squares approximation.

The process is applied iteratively to ensure a good fit to the image data. As is the case for other local optimisation-based object location techniques, this one extends to tracking by initialising the model in an image frame at the final rest position from the previous frame.

Improved performance can be achieved in some cases using a multi-resolution

search [21], whereby a coarse scale is used in the initial stages and finer resolution is gradually introduced to allow for refinement in the later stages. More accurate model fitting is also achieved if use is made of grey-level information around the landmark points [22].

Hill *et al* [36] describe another extension which uses *directional constraints* to help combat the *aperture problem*, which relates to the uncertainty in the correct position of a landmark *along* an image edge. The task of finding the best changes to the shape parameters is reformulated in terms of errors in normal and tangential directions separately. This allows landmarks to ‘slide’ along image edges more freely, and model fitting subsequently proceeds in fewer iterations.

### 2.2.4 The CONDENSATION Algorithm

In the CONDENSATION (Stochastic *Conditional Density Propagation*) tracking algorithm [43], the location of an object in an image is represented not by a single set of model parameters, but by a probability density function (pdf) over the model parameter space. A model of conditional probability (learned from training sequences) is used to propagate the pdf over time. In other words, given the pdf at time  $t$  there is a mechanism to predict the pdf at time  $t + 1$ , based on a simple model of object motion. The new pdf is consolidated and refined using measurements from the current image frame. Specifically, a *fitness* function is used to determine the goodness-of-fit of model to image for any given set of model parameters.

In practice, the pdf is represented by a population of samples drawn from the parameter space. Each one has its fitness calculated and *factored sampling* is used to choose seeds for propagation. The algorithm proceeds as follows:

1. To initialise, generate a population of  $N$  candidate model shapes  $\mathbf{s}_1 \dots \mathbf{s}_N$  at random positions in the shape parameter space.
2. Iterate:
  - (a) Calculate the fitness  $f_i = F(\mathbf{s}_i)$  of each shape.
  - (b) To produce each shape in the *new* population:
    - i. Select a member of the old population randomly, with probability of selecting shape  $j$  being equal to  $f_j / \sum_i f_i$
    - ii. Apply the propagation function  $P(\mathbf{s})$  to produce the new shape. Isard and Blake use a ‘Fokker-Planck’ (drift-and-diffuse) stochastic differential equation [65]:



$$P(\mathbf{s}) = \mathbf{A}\mathbf{s} + \mathbf{B}\omega \quad (2.13)$$

where  $\omega$  is a vector of independent standard normal random variables, and  $\mathbf{A}$  and  $\mathbf{B}$  are constant matrices learned from training sequences of characteristic movements;  $\mathbf{A}$  defines the deterministic ‘drift’ in the model and  $\mathbf{B}$  is used to scale and orientate  $\omega$ .

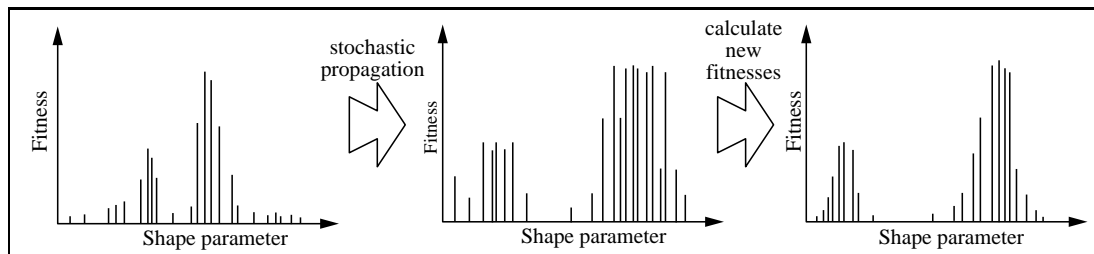


Figure 2.4: Propagation of a shape population with only one shape parameter using the CONDENSATION algorithm. Each vertical line represents a candidate shape in the population. The fitness peaks indicate promising solutions.

Figure 2.4 illustrates the algorithm, and a more detailed explanation is given in [43]. A major issue is the question of what the ‘correct’ object shape is, given a pdf represented by a discrete population of candidate shapes. One possibility is to take the *modal* (highest fitness) shape, but this can sometimes produce a noisy output. Alternatively the statistical mean can be found, using:

$$\bar{\mathbf{s}} = \frac{\sum f_i \mathbf{s}_i}{\sum f_i} \quad (2.14)$$

However, this has the undesirable feature of interpolating between peaks in the pdf. Neither solution is perfect but both are simple to calculate.

The fitness function  $F(\mathbf{s})$  should be designed to produce a value which indicates the ‘goodness of fit’ of a shape to an object in the image. For a contour model, a simple such algorithm examines image pixels along a line normal to the contour at each control point and returns a score proportional to the number, proximity and strength of edges found in the image. More details can be found in [32].

The benefits of CONDENSATION are as follows:

- It can support multiple hypotheses; this is represented by a pdf with multiple peaks.
- It recovers well from failure; the stochastic nature of the algorithm allows it to escape from local maxima.

- It incorporates a level of prediction, which improves the speed of convergence and the quality of results over, for example, a Genetic Algorithm.

The prediction aspect is embedded in the propagation equations. Currently these have two elements; a deterministic term which allows for simple drifting of the pdf, and a stochastic term which encourages spreading of the pdf. Although the tracker *can* escape from local maxima (due to the stochastic term), the underlying dynamical model is still based on an assumption of smooth, continuous object movement. Such an assumption is not always valid.

# Chapter 3

## Statistical Shape Models for Objects which Bend or Pivot

---

### 3.1 Introduction

As discussed in Section 2.2.1.1, the standard PDM is based purely on linear statistics: for any particular mode of variation, the positions of landmark points can vary only along straight lines. Non-linear variation is achieved by combining two or more modes. This situation is not ideal, firstly because the most compact representation of shape variability is not achieved, and secondly because implausible shapes can be generated, due to the incorrect assumption that the variation modes are independent (i.e. models are not *specific*). The specificity is particularly poor when the object being modelled can bend or pivot.

Since bending and pivoting are such major features of so many classes of object, it is reasonable that these actions should be modelled directly. This can be achieved within the framework of a PDM by the use of polar coordinates. Objects with both linear and angular deformation are best served by a hybrid model in which each landmark can exist either in Cartesian or polar space.

There has been much work in the past describing the use of *articulated* models for vision tasks [62, 39, 27]. The approach described here has two advantages over such systems: it is more general, not being limited to purely pivotal motion, and also

the constraints amongst the pivots are captured automatically during the training process. Overall there is less hard-coding of parameters to be done.

The remainder of this chapter gives more details on these ideas. The construction of a Cartesian-Polar Hybrid PDM is explained and an experimental comparison of the new technique with both the standard PDM and the Polynomial Regression PDM of Sozou *et al* (see [68] and Section 2.2.1.2) is made. In addition, we present two different algorithms for automatically determining pivot positions within a model, via analysis of the training set, and we test them on both real and synthetic data.

## 3.2 The Cartesian-Polar Hybrid PDM

The Cartesian-Polar Hybrid PDM attempts to overcome limitations of the standard PDM by allowing angular movement to be modelled directly. This is achieved by a reparameterisation of coordinates according to some predefined mapping function. Landmarks which appear to pivot about some other landmark in the model are transformed into polar coordinates, with the suspected pivot as origin and some other landmark chosen strategically to define the polar reference axis. Landmarks which have no such angular behaviour remain in the Cartesian domain.

A shape  $\mathbf{x} = (x_1, y_1, \dots, x_N, y_N)$ , where  $(x_i, y_i)$  are the Cartesian coordinates of landmark  $i$ , is reparameterised by a mapping  $P$  into a parameter vector  $\mathbf{q} = (q_1, q_2, \dots, q_{2N-1}, q_{2N})$  whereby each  $(x, y)$  pair in  $\mathbf{x}$  is either mapped in a Cartesian or polar fashion as follows:

$$\begin{array}{ll}
 (x_p, y_p) & \xrightarrow{\text{Cartesian map}} \quad \begin{array}{l} q_{2p-1} = x_p \\ q_{2p} = y_p \end{array} \\
 (x_p, y_p) & \xrightarrow[\text{centre landmark = } c]{\text{polar map}} \quad \begin{array}{l} q_{2p-1} = r_p = \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} \\ q_{2p} = \theta_p = \phi_p + \phi_a \end{array} \\
 & \text{where } \phi_i = \tan^{-1} \frac{y_i - y_c}{x_i - x_c}
 \end{array}$$

By allowing angles to be measured relative to axis reference landmarks, it is possible to model objects which have *series* of jointed parts (such as in an angle-poise lamp), or continuous bending regions (such as the tadpoles and chromosomes described by Sozou *et al* [68]) to a greater degree of accuracy.

The axis reference landmark  $a$  for a polar-mapped landmark  $p$  should be chosen

with care. If  $p$  is pivoting off some locally rigid structure within the object (about a centre landmark  $c$ ) then  $a$  should be a landmark within this structure. For maximum stability,  $a$  should be as distant from  $c$  as possible (averaged over the training set), and also cyclic dependencies must be avoided. Figure 3.1 gives an example labelling.

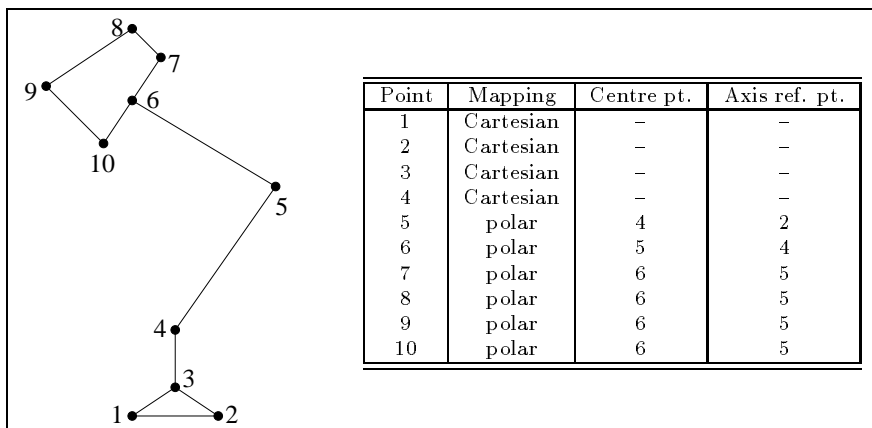


Figure 3.1: Landmark point mappings used for an anglepoise lamp

In training, the parameter mapping is applied to every training example. The mean shape in parametric form  $\bar{\mathbf{q}}$  is found simply by averaging each parameter over the training set. PCA is used on the training parameter vectors in order to find the major modes of variation (after removing any polar bias — see Section 3.2.1 below).

Generating a shape from shape parameters  $b_1 \dots b_t$  is a two-stage process. In the first stage the parameter vector  $\mathbf{q}$  is generated as for the standard PDM, using (2.1). In the second stage,  $\mathbf{q}$  is mapped into Cartesian coordinates:

$$\begin{array}{lll}
 q_{2p-1} & \xrightarrow{\text{Cartesian map}} & x_p = q_{2p-1} \\
 q_{2p} & & y_p = q_{2p} \\
 q_{2p-1} \quad (= r_p) & \xrightarrow{\text{polar map}} & x_p = x_c + r_p \cos(\theta_p + \psi) \\
 q_{2p} \quad (= \theta_p) & \text{centre landmark} = c & y_p = y_c + r_p \sin(\theta_p + \psi) \\
 & \text{axis landmark} = a & \text{where } \psi = \tan^{-1} \frac{y_a - y_c}{x_a - x_c}
 \end{array}$$

The *order* in which landmarks are mapped is important: because the position of landmark  $p$  is dependent on the positions of landmarks  $c$  and  $a$ , the latter two landmarks must be mapped *before* landmark  $p$ .

### 3.2.1 Removing Polar Bias

A successful PCA requires all dimensions in the shape space to be *comparable* in that they have equal levels of significance. For example, angles span a total range of  $2\pi$  radians, whereas displacements could have a range of up to, say, 50 pixels. Quantities measured in these two domains are not directly comparable. Also, a polar-mapped landmark which is close to its pivot gives rise to a large angular movement for a relatively small displacement, so angles measured at different radii are not directly comparable.

We combat both of these problems with a single solution. Using the relationship  $s = r\theta$  (where  $s$  is the arc length of an angle  $\theta$  at radius  $r$ ) we can convert angles into arc lengths, which are comparable with displacements. For the conversion of landmark  $p$  into polar coordinates, a constant scale factor  $R_p$  is used.  $R_p$  is calculated as being the average distance of landmark  $p$  from its centre landmark over the training set; this ensures that every training example is scaled by the same factor.

Recent work by Sumpter *et al* [70] describes a more general approach to finding suitable linear scalings for non-comparable parameters prior to performing a PCA. Therein, the information content of the model is maximised via the calculation of *eigen-entropy*.

### 3.2.2 Coping with the Angle Discontinuity Problem

Another problem encountered with the use of polar coordinates is that the PCA gives unexpected results when the range of angles measured crosses the  $0^\circ/360^\circ$  angle discontinuity. For example, the samples  $\{1^\circ, 2^\circ, 359^\circ, 358^\circ\}$  have a mean of  $180^\circ$  and not the desired  $0^\circ$ .

This problem can be avoided in all but the most pathological of cases. For any polar-mapped point, the angle is calculated twice for each training example — once in the range  $0 \leq \theta < 360^\circ$  and once in the range  $-180^\circ \leq \theta < 180^\circ$ . The standard deviation of the angles over the training set is calculated separately for both ranges. If the standard deviation is the same for both ranges then no boundary crossing has occurred. If it is larger for one range, it is assumed that this involved a boundary crossing and angles from the *other* range are used when performing the PCA.

### 3.3 Designing a Mapping

Of chief importance to the Hybrid PDM is the choice of a suitable parameter mapping. Each landmark must be classified as either Cartesian- or polar-mapped; in the latter case, a choice of centre and axis reference is also needed. This can be done by hand when the pivots are obvious and the model has relatively few landmarks; however, for larger models, automation is desirable.

Two different algorithms are proposed for this. The first is a simple approach, but is limited in that it assumes that all desired pivots on the object correspond to existing landmarks. This is not always the case; the second algorithm has no such restriction, and is consequently more complex.

#### 3.3.1 The ‘Compacter’ Mapping Algorithm

For a given set of training data, the ideal mapping would provide the most *compact* model. Compactness can be measured in terms of the total variance existing in the model. The modes of variation produced by the PCA are in fact eigenvectors of a covariance matrix and the corresponding eigenvalues provide a measure of the variance captured by each individual mode. The sum of the magnitudes of these eigenvalues thus gives a measure of the *overall* variance present in the model. The smaller this value, the more compact the model. Hence the best mapping is the one which minimises this eigenvalue sum. Trying every possible combination of mappings takes exponential time and is thus not feasible for large models. A heuristic alternative is suggested below.

1. Construct a *base set* of relatively stable landmarks as follows:
  - (a) Align the training set to a common axis, as for standard PDM training.
  - (b) Find the mean *Cartesian* shape  $\bar{\mathbf{x}}$ .
  - (c) For each landmark, find its average displacement (over the training set) from its mean position.
  - (d) Include the landmark in the base set if this average distance is less than a specified threshold value<sup>1</sup>.
  - (e) Assign an *axis reference partner* to each base set landmark. This can be any other base set landmark; choosing the one which maximises the

---

<sup>1</sup>0.05 times the total model size has been used in this work.

average distance from a landmark to its partner provides the most stable axes.

2. Assign a Cartesian mapping to all base set landmarks. This is acceptable as there is little deformation in the base set.
3. Attempt to assign mappings to the remaining landmarks; the strategy is to select potential mappings intelligently and use the compactness measure to find the best one:
  - (a) Choose the next unassigned landmark. Call this landmark  $a$ .
  - (b) Assign a Cartesian mapping to landmark  $a$  and perform PCA on the base set plus landmark  $a$ . The sum of the eigenvalues produced gives a measure of the total variance of the model when  $a$  is mapped in the Cartesian domain.
  - (c) Take each base set landmark in turn; call it landmark  $b$ . Calculate the distance between  $a$  and  $b$  for every training example. Find the variance of these distances. If this variance is below a fixed threshold<sup>2</sup> then it is possible that  $a$  pivots about  $b$ . To test this hypothesis, assign a polar mapping to landmark  $a$ , with landmark  $b$  as centre landmark and using landmark  $b$ 's axis reference partner. Perform PCA on the base set plus landmark  $a$ , and record the sum of the eigenvalues produced.
  - (d) Once all base set landmarks have been tested against landmark  $a$ , find the lowest recorded eigenvalue sum. Assign the corresponding mapping to landmark  $a$  (be it Cartesian or polar), and add it to the base set. However, if *no* base set landmarks were tested against landmark  $a$  then do *not* add it to the base set. This allows for cases where  $a$  is best pivoted around an as-yet unassigned landmark.
  - (e) Iterate steps (a) to (d) until no more mapping assignments are possible.

4. Assign a Cartesian mapping to all remaining unassigned landmarks.

This process is computationally intensive, taking several minutes to run even for a relatively small model (60 landmarks). However, it is performed off-line, so speed is not critical. The technique is heuristic, so is not guaranteed to find the best solution. Also, the algorithm can only produce sensible results if all pivots present in the object have been landmarked.

---

<sup>2</sup>0.04 times the total model size has been used in this work.



### 3.3.2 The ‘Annotator’ Mapping Algorithm

The above algorithm is restricted in that landmarks may only pivot about other existing landmarks. The technique suggested below applies a geometric approach to overcome this problem. Statistical analysis of the training data is used to spot trends in the movement of landmarks; broadly speaking, if a landmark appears to move in an arc then it should be classified as polar and its centre of rotation should be marked. To allow for chains of pivots this idea is extended: polar movement *relative to other landmarks or sets of landmarks* is detected. The method employed is to construct sets of landmarks which appear to represent different roughly-rigid parts of the object, and look for pivotal relationships between pairs of sets.

#### 3.3.2.1 Finding Rigid Sets of Landmarks

Sets of landmarks are to be found which appear to be rigid in the sense that they appear to ‘move together’ over the training set. In order to achieve this, the  $N$  by  $N$  normalised distance variance matrix  $\mathbf{D}$  for the training set is first calculated:

$$[\mathbf{D}]_{ij} = \text{Var}_{e=1}^E \left( \frac{\sqrt{(x_{j,e} - x_{i,e})^2 + (y_{j,e} - y_{i,e})^2}}{S_e} \right) \quad (3.1)$$

where  $E$  is the number of training examples,  $(x_{i,e}, y_{i,e})$  are the coordinates of landmark  $i$  in training example  $e$ ,  $\text{Var}$  is the variance operator, and  $S_e$  is a measure of the overall size of example  $e$ , given by:

$$S_e = \frac{1}{E} \sum_{i=1}^N \sum_{j=1}^N \sqrt{(x_{j,e} - x_{i,e})^2 + (y_{j,e} - y_{i,e})^2} \quad (3.2)$$

A small value for  $[\mathbf{D}]_{ij}$  indicates that landmarks  $i$  and  $j$  move together over the training set. Hence  $\mathbf{D}$  can be used as a basis for ‘growing’ rigid sets as follows:

1. Start with each landmark as a singleton set.
2. For each set, attempt to add a landmark. If  $G$  is the set under consideration, then choose the landmark  $k \notin G$  which minimises  $S_k$  in:

$$S_k = \max([\mathbf{D}]_{ij} | i, j \in G \cup \{k\}) \quad (3.3)$$

If  $S_k$  is below a fixed threshold then add landmark  $k$  to the set, otherwise no landmark is added.

3. Remove any duplicate sets formed as a result of step 2.
4. Iterate steps 2 and 3 until sets have stopped growing.
5. Finally, discard sets with two landmarks or less. It follows that any landmarks which do not form part of a rigid body are not assigned to a set, and are excluded from the subsequent pivot search.

The threshold used for set inclusion is deliberately quite stringent, making sets smaller than they might otherwise be: two sets which should ideally be one can be merged later when they are discovered to have a similar pivot, but one set which should ideally be two can give poor results at the pivot-finding stage.

The use of thresholding can, in some cases, cause problems since different model components often require different thresholds for good results. For example, in objects such as the human hand, which consist of a large main body (i.e. the palm) and smaller, pivoting sections (i.e. the fingers, and in particular the thumb, which has two separate rigid portions), a larger threshold is generally required for the main body than for the smaller sections. The method used to combat this involves a three-stage process:

1. The training set is aligned using a weighted least-squares technique [19] (the same alignment process is used prior to PCA in the training process). Landmarks whose standard deviation from their mean position is less than a fixed threshold (1.5% of the model size) are deemed fairly static and are thus included in a *base set* of landmarks.
2. Set-growing is then performed as described above, but using only the landmarks not in the base set.
3. When the iteration has converged, base set landmarks are reintroduced, and the iterative process is again allowed to converge. This final step allows overlap between the base set and other sets.

### 3.3.2.2 Finding Potential Pivots Between Sets

Having identified rigid sets of landmarks, the next step is to find plausible pivots and augment training examples with additional landmarks at the pivots. A pivot will generally have different coordinates in each training example – even if the examples are globally aligned, a pivot in a chain will not occupy a fixed position. It is therefore

necessary to construct a local coordinate frame for each rigid set in each training example, such that landmarks in the set are roughly static *in local coordinates* over all training examples. Static pivot positions can be located in the *local* coordinate frames, and then mapped back into global coordinates. The details of the procedure are as follows:

1. For each set  $g$  in each training example  $e$ , it is necessary to find a suitable coordinate frame transformation  $C_{g,e} = (s_{g,e}, t_{g,e}, p_{g,e}, q_{g,e})$ , where a point  $(x, y)$  in local coordinates is transformed onto  $(x', y')$  in global coordinates by:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} p_{g,e} & -q_{g,e} & s_{g,e} \\ q_{g,e} & p_{g,e} & t_{g,e} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (3.4)$$

The transformation matrix in (3.4) is labelled  $\mathbf{R}_{g,e}$ .

An easy way to produce the  $C_{g,e}$  is to perform a least-squares alignment of set  $g$ 's member landmarks over the training set (this is the same process used for training set alignment in PDM construction [19]). We can then say that  $\mathbf{R}_{g,e}$  is the inverse of the transformation required to align example  $e$ .

2. For each pair of sets  $a$  and  $b$ , the best pivot is found. Define this point to be at  $(u_a, v_a)$  in  $a$ 's local coordinates and at  $(u_b, v_b)$  in  $b$ 's local coordinates. Ideally, for each training example, these two points would both transform onto the same global position, i.e. if  $\mathbf{R}_{g,e}(u_g, v_g, 1)^T = (u_{g,e}, v_{g,e}, 1)^T$  then  $\forall e.(u_{a,e}, v_{a,e}) \approx (u_{b,e}, v_{b,e})$ . Figure 3.2 illustrates.

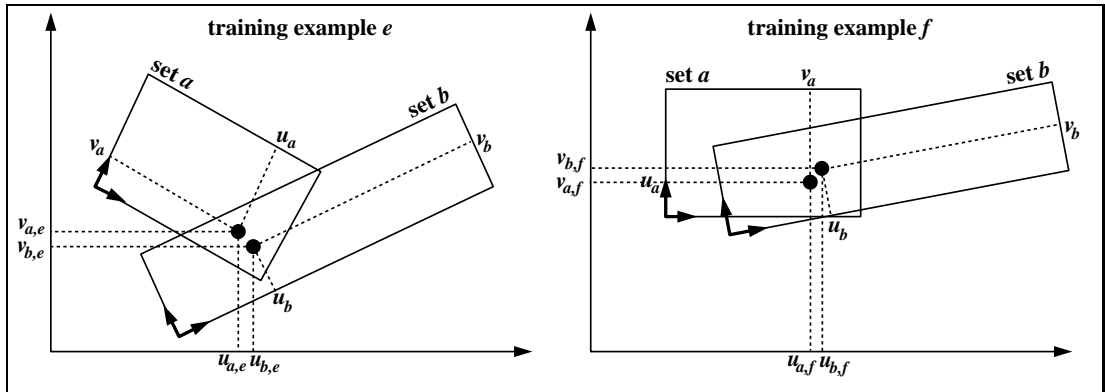


Figure 3.2: Illustration of terms used in the ‘annotator’ algorithm.

Hence, we find the values of  $u_a$ ,  $v_a$ ,  $u_b$  and  $v_b$  which minimise  $\mathcal{E}$ , the sum of the squares of the distances between the pairs of global coordinates:

$$\mathcal{E} = \sum_{e=1}^E (u_{b,e} - u_{a,e})^2 + (v_{b,e} - v_{a,e})^2 \quad (3.5)$$

Using  $(u_{g,e}, v_{g,e}, 1)^T = \mathbf{R}_{g,e}(u_g, v_g, 1)^T$ , we obtain:

$$\mathcal{E} = \sum_{e=1}^E \left( ((p_{b,e}u_a - q_{b,e}v_b + s_{b,e}) - (p_{a,e}u_a - q_{a,e}v_a + s_{a,e}))^2 + \right. \\ \left. ((q_{b,e}u_a + p_{b,e}v_b + t_{b,e}) - (q_{a,e}u_a + p_{a,e}v_a + t_{a,e}))^2 \right) \quad (3.6)$$

To minimise  $\mathcal{E}$ , equate the partial derivatives  $\partial\mathcal{E}/\partial u_a$ ,  $\partial\mathcal{E}/\partial v_a$ ,  $\partial\mathcal{E}/\partial u_b$  and  $\partial\mathcal{E}/\partial v_b$  to zero, and hence obtain four simultaneous linear equations, the solution of which simplifies to:

$$(u_a, v_a, u_b, v_b)^T = \sum_{e=1}^E (J_e^T J_e + K_e^T K_e)^{-1} \sum_{e=1}^E (s_{b,e} - s_{a,e}) J_e + (t_{b,e} - t_{a,e}) K_e \quad (3.7)$$

where  $J_e = (p_{a,e}, -q_{a,e}, p_{b,e}, -q_{b,e})$  and  $K_e = (q_{a,e}, p_{a,e}, q_{b,e}, p_{b,e})$ .

From here, the pivot point in global coordinates  $(c_{a,b,e}, d_{a,b,e})$  can be found for each training example  $e$  by transforming the two points into global coordinates and averaging them:

$$\begin{pmatrix} c_{a,b,e} \\ d_{a,b,e} \end{pmatrix} = \frac{1}{2} \left( \begin{pmatrix} u_{a,e} \\ v_{a,e} \end{pmatrix} + \begin{pmatrix} u_{b,e} \\ v_{b,e} \end{pmatrix} \right) \quad (3.8)$$

The variability  $V_{a,b}$  of the pivot can be measured as the average separation of these two points over the training set, when mapped into global coordinates:

$$V_{a,b} = \frac{1}{E} \sum_{i=1}^E \sqrt{(u_{b,e} - u_{a,e})^2 + (v_{b,e} - v_{a,e})^2} \quad (3.9)$$

3. Declare a pivot between sets  $a$  and  $b$  if *all* of the following are true:

- The variability  $V_{a,b}$  of the pivot is less than 5 pixels.
- The standard deviation of the distribution of angles between the coordinate frames for  $a$  and  $b$  over the training set is at least  $5^\circ$ .
- The normalised deviation (standard deviation divided by mean) in the distribution of the size ratios of the two coordinate frames is less than 0.1 (i.e. the relative size of the two sets remains fairly constant).

The thresholds have been chosen to be tolerant, as false pivots can still be rejected in the final stage.

### 3.3.2.3 Constructing the Pivotal Structure

Once a set of potential pivots has been found, a mapping for use by the Hybrid PDM must be constructed, noting that the formulation of polar mappings relies on having no cyclic dependencies amongst landmarks. Hence, sets are organised into one or more tree-like structures, where child sets pivot off parent sets, and the root set(s) provide(s) a relatively stable reference.

The algorithm that has been used in this work caters for most cases. A single tree is constructed: the root is taken to be the largest rigid set found, and a breadth-first search is used to attach child sets which have a common pivot with the parent set. Using the breadth-first search ensures that the structure found has the shortest possible pivotal chains.

The mapping is then constructed in an obvious fashion: landmarks which are in a set with no parent (i.e. at the tree root or not included in the tree), and landmarks not in a set, are assigned a Cartesian mapping.

Landmarks in parented sets are assigned a polar mapping. New landmarks are generated in all training examples to represent the pivots, as described in 5.2.2 above. If a new pivot is *consistently* sufficiently close (5 pixels) to any existing landmark, it is removed and the existing landmark used instead. A suitable landmark from the parent set is chosen as an axis reference.

## 3.4 Results

### 3.4.1 Comparing the Different Modelling Techniques

In this section the performance of the Cartesian-Polar Hybrid PDM is compared with that of the standard PDM and the Polynomial Regression PDM (PRPDM) of Sozou *et al* (as described in [68] and Section 2.2.1.2). Thirty training images of an anglepoise lamp (chosen for its multiple jointed pivots) were captured in various positions. The lamp was positioned such that all rotations were parallel to the image plane i.e. no 3D effects were experienced. The locations of 10 landmarks were recorded for each training image, as defined in Figure 3.1. Models were then generated using each of four methods: a standard PDM, a quadratic PRPDM, a cubic PRPDM and a Hybrid PDM. Figures 3.3, 3.4 and 3.5 show the first four modes

of variation for three of these models. A range of  $\pm 2$  s.d. is shown, with the darkest central figure being the mean in each case.

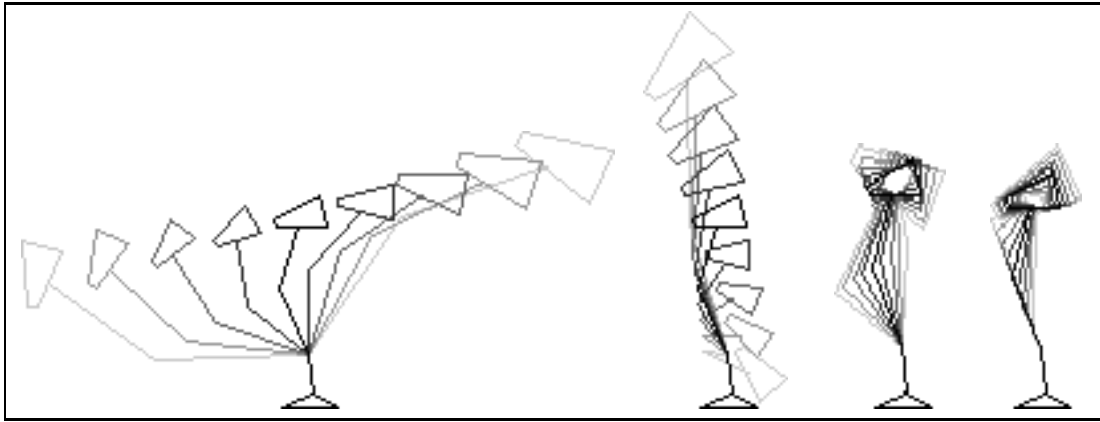


Figure 3.3: The first four modes of variation under the standard PDM.

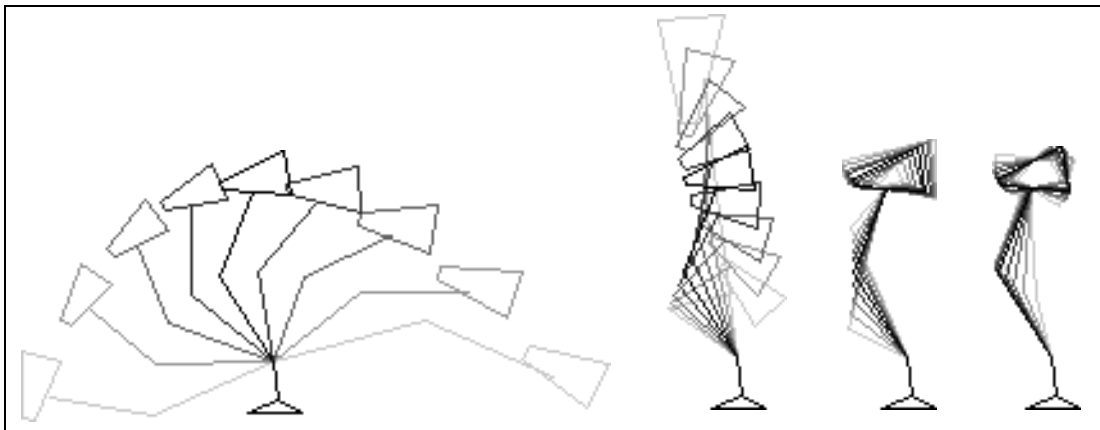


Figure 3.4: The first four modes of variation under the quadratic PRPDM.

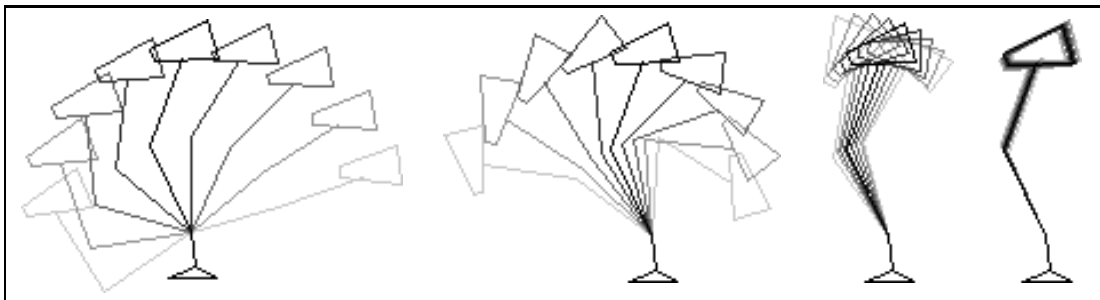


Figure 3.5: The first four modes of variation under the hybrid PDM.

Figure 3.6 gives a statistical comparison of the modelling techniques. The graph shows, for each model, what proportion of the total standard deviation is captured

with respect to the number of modes used. Unlike the other models, the curve for the hybrid model does not start at the origin; this is because the reparameterisation process captures (or eliminates) a percentage (58.5%) of the deviation as experienced by the other models (indicated by a lower value for the sum-of-eigenvalues in the case of the hybrid model).

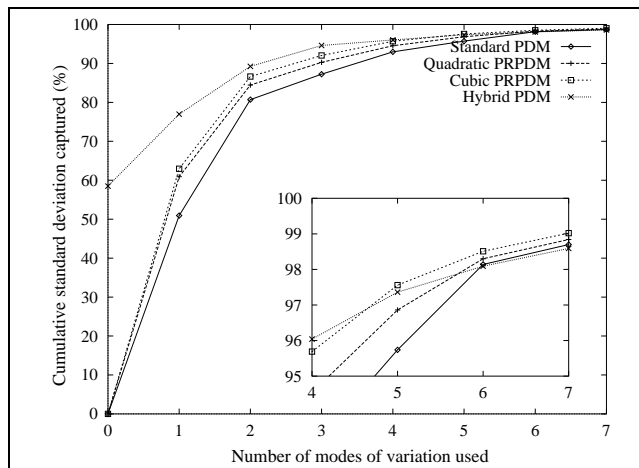


Figure 3.6: Graph showing how variation is accounted for under the various models.

As can clearly be seen from Figure 3.3, the standard PDM fails to capture the pivotal nature of the object. Landmarks move in straight lines for each variation mode, the lamp head changes size and the pivotal arms are stretched and compressed, thus implausible shapes can be generated. The anglepoise has only three pivots, so any variation seen in the fourth mode must be acting purely to compensate for inaccuracies in earlier modes. Statistically, this model is the least compact, the cumulative deviation captured being the lowest at every stage.

The Quadratic PRPDM improves on the standard PDM; the most significant mode of variation captures *some* bending, but the circular arc is approximated by a parabola. Again, some stretching is seen, but not to the same degree as for the standard PDM. There is still a fair amount of compensatory variation in the fourth mode. The Cubic PRPDM performs better statistically than the Quadratic version, and in general, performance increases with polynomial degree.

The Cartesian-Polar Hybrid PDM captures the pivotal nature of the object precisely. The modes of variation show no changes in lamp head size or stretching of arms. It is also interesting to note that the major mode captures the fact that the lamp head generally remains facing in the same direction as the lamp body is moved. Statistically, the mere reparameterisation of the model explains over 50% of the standard deviation. The Hybrid PDM is most compact up until the introduction

of mode 5, by which time the remaining variation is due purely to noise.

Table 3.1 shows how the techniques compare in terms of computational complexity. The values shown in the middle column are numbers of machine cycles required to generate a model instance (this affects the speed of feature location/tracking). The tests were performed on a MIPS R4400 processor, with a MIPS R4010 FPU. A machine speed of 25MHz is assumed; statistics were captured using the code-profiler, *pixie*. As expected, the standard PDM is the fastest, but note that the hybrid model is not much slower. Also, and most importantly, the two PRPDMS are substantially slower (about 6 times).

Model	Cycles	Time/ms
Standard PDM	215621	8.6
Quadratic PRPDM	1326126	53.0
Cubic PRPDM	1417455	56.7
Hybrid PDM	221105	8.8

Table 3.1: Machine cycles used to generate a model instance under each technique.

Figure 3.7 gives another example of a case where the Hybrid PDM is useful. As can be seen, the standard PDM modelling of a golf swing (constructed from a single continuous sequence of 30 training images) suffers because of the large angular movements involved. The Hybrid PDM again performs much better.

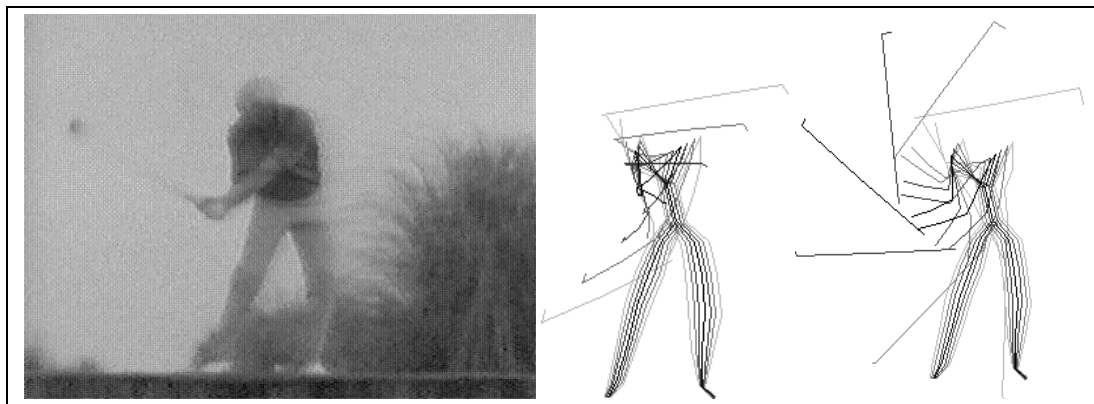


Figure 3.7: Modelling a golf swing; one of the training images (left), the most significant mode of variation for the standard PDM (middle) and the Hybrid PDM (right).

### 3.4.2 Comparing Mapping Generation Algorithms

Results are presented for experiments using training data from human hands. Thirty training examples were captured, all of hands positioned with the palm down and



fingers outstretched, and the positions of 61 landmarks were extracted manually from each one. Both of the automatic mapping generation algorithms described in Section 3.3 were applied to the data. Figures 3.8 and 3.9 illustrate the results.

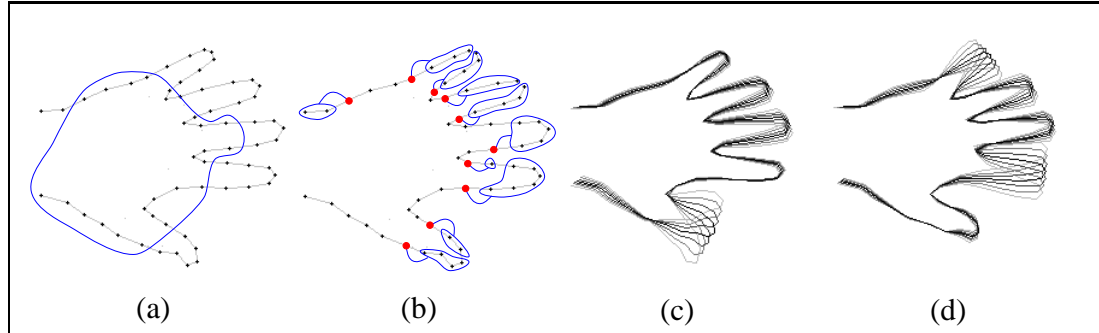


Figure 3.8: Automatic mapping generation for a hand model using the ‘compacter’ algorithm; (a) the base set of landmarks, (b) sets of polar-mapped landmarks and their centres of rotation, (c) & (d) two modes of variation of the trained model.

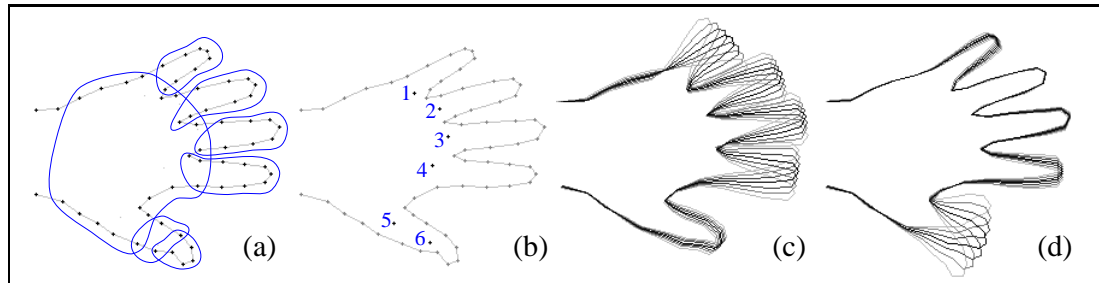


Figure 3.9: Automatic pivot generation for a hand model using the ‘annotator’ algorithm; (a) rigid area sets, (b) the augmented pivots, (c) & (d) two modes of variation.

Figure 3.8 gives results for the ‘compacter’ algorithm, in which existing landmarks are used as pivots. Plot (a) shows which landmarks were included in the base set, and plot (b) shows how other landmarks were mapped. Several landmarks were generally assigned the same pivot – this is indicated by a ring around the landmarks and a line connecting the ring to the chosen pivot. Plots (c) and (d) show two modes of variation after training under this particular mapping.

The base set of landmarks includes most of the palm of the hand, as expected. It also extends part way into the middle finger; this is because following training set alignment, the middle finger is fairly static. Remaining landmarks generally pivot about the closest base set landmark, indicating that this strategy gives the most compact model.

Figure 3.9 gives results for the ‘annotator’ algorithm, in which pivots are located and augmented onto training examples. Plot (a) shows the landmark sets that have been constructed and plot (b) shows the pivots thus generated. Each finger has a single landmark set, and the thumb has been split into two sets, giving two rigid sections. Landmarks near the wrist have not been assigned to the ‘palm’ set because the wrist angle varies significantly over the training set. They have not been assigned their own set because there are too few landmarks for such a set.

All pivots (1 to 6) have been placed roughly where one would expect them; the pivotal structure of the thumb is such that pivot 6 is parented by pivot 5.

Plots (c) and (d) show examples of the variation modes produced after training using the annotated data. Pivotal motion is visible, especially in the thumb.

These results are encouraging: the natural structures present in the human hand have been captured accurately and the pivots have been positioned approximately as expected.

Quantitative data was obtained in a second experiment. A synthetic model was constructed, comprising a fixed body and a pivoting arm (see Figure 3.10). Training sets were generated with each example having the arm pivoted at a different angle, and with additive Gaussian noise applied to the position of each landmark. The idea was to study the accuracy of finding the pivot whilst varying the following input parameters:

- The range of angles of the pivoted arm (a Gaussian distribution with standard deviation of  $\sigma_a$  degrees).
- The amount of noise used to displace landmarks (additive Gaussian, with standard deviation  $\sigma_n$  pixels).
- The number of training examples used,  $E$ .
- The total number of landmarks in the model,  $P$ .

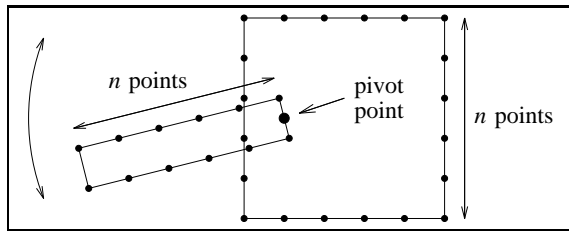


Figure 3.10: A synthetic model of an object with a single pivot.

Running the ‘annotator’ pivot finder on the synthetic data produces the coordinates of a single pivot. The distance of this pivot from ground truth can be calculated as a measure of accuracy. Several hundred trials were performed for each choice of parameters to give the *mean output error* with tight confidence limits.

The angle range  $\sigma_a$  is an inherent property of the object being modelled and the noise  $\sigma_n$  is dependent on the accuracy of the image capture technique. Figure 3.11(a) shows how they both affect accuracy of the pivot position<sup>3</sup>. As expected, the accuracy decreases as the angle decreases, and as the level of noise increases.

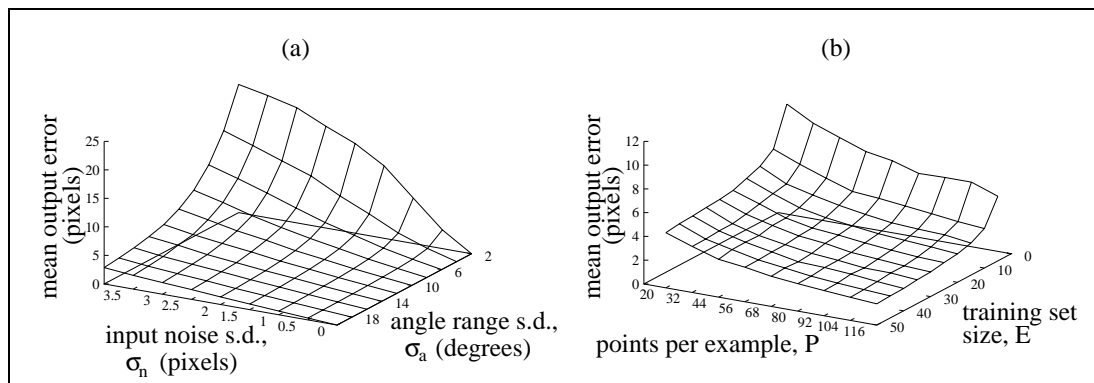


Figure 3.11: Pivot location accuracy under varying conditions of (a) input noise and angle range; (b) training set size and number of points per training example.

The remaining parameters  $E$  and  $P$  can be seen as measures that could be employed to improve accuracy. As can be seen in Figure 3.11(b), accuracy improves with an increase in both the training set size and the number of landmarks<sup>4</sup>. More specifically, increasing the training set size only helps to a certain extent (in this instance, above 25 examples there is little improvement), but increasing the number of landmarks per example gives a more steady decrease in error.

### 3.5 Tracking

Tracking proceeds as for the standard PDM, via Active Shape Models (see [15] and Section 2.2.3). However, in the calculation of the change in shape, some of the suggested landmark movements must be mapped into their polar equivalent in the model frame. This can be achieved by applying a mapping to the vector  $d\mathbf{x}$  of residual landmark movements from Equation 2.11:

<sup>3</sup>The control values used were  $P = 32$  and  $E = 20$ .

<sup>4</sup>The control values used were  $\sigma_n = 3$  pixels and  $\sigma_a = 8^\circ$ .

$$d\mathbf{q} = M(d\mathbf{x}, \mathbf{x}) \quad (3.10)$$

where  $M$  maps individual  $(dx_p, dy_p)$  pairs into  $dq_{2p-1}$  and  $dq_{2p}$  as follows:

$$\begin{array}{lll} (dx_p, dy_p) & \xrightarrow{\text{Cartesian map}} & dq_{2p-1} = dx_p \\ & & dq_{2p} = dy_p \\ (dx_p, dy_p) & \xrightarrow{\substack{\text{polar map} \\ \text{centre landmark} = c \\ \text{axis landmark} = a}} & dq_{2p-1} = dr_p = R_p \frac{xdx_p + ydy_p}{\sqrt{x^2 + y^2}} \\ & & dq_{2p} = d\theta_p \approx \frac{xdy_p - ydx_p}{x^2 + y^2} \end{array}$$

where  $x = x_l - x_c$ ,  $y = y_l - y_c$  and  $R_p$  is the constant scale factor which converts the angle measurement for landmark  $p$  into an arc length (see Section 3.2.1 on polar bias).

In order to test tracking performance, a real time ASM tracker was implemented [32]. A controlled test environment was constructed; a camera was mounted on a tripod, pointing down at a homogeneous dark surface. The various models were exercised rigorously in this environment.

Tracking with the Hybrid PDM proved promising. Experiments were performed for hand tracking, a task at which the standard PDM has already been successful [32]. However, because the hybrid model requires fewer modes of variation, the system runs faster.

### 3.6 Extension to 3D

Extension of the Cartesian-Polar Hybrid PDM into 3D can be achieved using either cylindrical  $(r, \theta, z)$  or spherical  $(r, \theta, \phi)$  polar coordinates. In either case, three reference landmarks are required for each polar-mapped landmark. Figure 3.12 illustrates how they are arranged. 3D hybrid models can thus contain up to three different mappings—Cartesian, cylindrical polar and spherical polar—each being chosen where most appropriate. Figure 3.13 shows a 3D human hand model which has been constructed using a mixture of Cartesian and cylindrical polar coordinates.

Tracking in 3D, using the hand model illustrated in Figure 3.13, was less successful than in 2D (see Section 6.5.2). The polar-mapped regions of the model (i.e. the fingers) sometimes tracked well, but often exhibited unexpected deformations from which it was difficult to recover. Extensive investigation did not reveal the exact

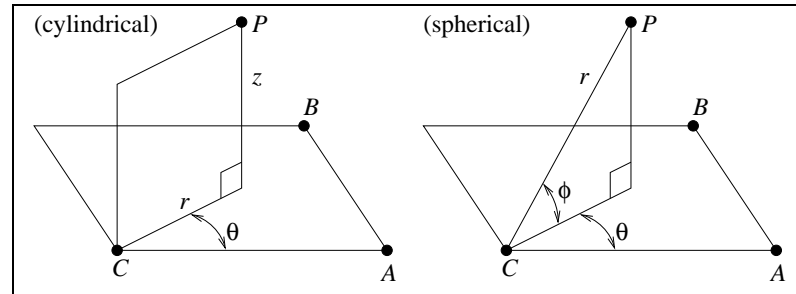


Figure 3.12: Cylindrical and spherical polar coordinates.  $P$  is the polar-mapped landmark,  $C$  is the origin reference landmark,  $A$  is the primary axis reference landmark and  $B$  is the secondary axis reference landmark.

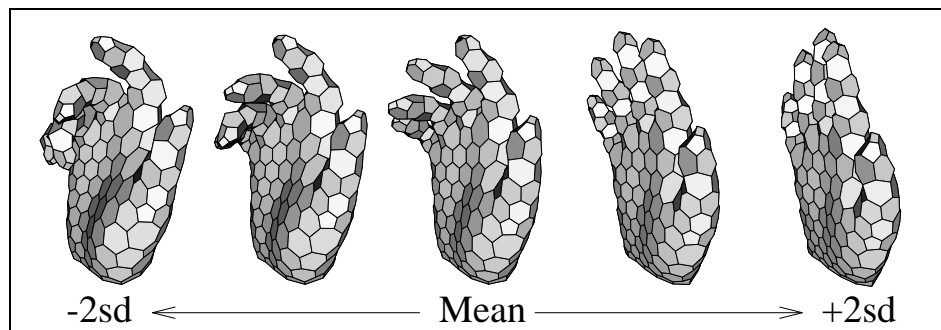


Figure 3.13: First mode of variation for a 3D Hybrid PDM of a human hand.

nature of the instability. The most likely cause is the combination of projective transformations and polar mappings giving rise to unstable pose change calculations. This is an artifact of the data-driven approach employed; use of a fitness-based approach, such as a Genetic Algorithm or the CONDENSATION algorithm (see Section 2.2.4) might alleviate such a problem.

### 3.7 Conclusions

In this chapter it has been shown how the Cartesian-Polar Hybrid PDM can, in some cases, improve the compactness and specificity of a deformable shape model.

It is possible that improvements could be made to the mapping algorithms. The ‘compacter’ algorithm is heuristic, so there is no certainty that it is finding the best mapping. The ‘annotator’ algorithm constructs a pivot ‘tree’ based on a very simple breadth-first search, in which only one set can act as a root. This caters for most cases, but objects with several distinct pivoting structures will cause problems. Both algorithms make use of several manually-fixed thresholds; a method requiring no thresholding would be preferable in order to handle a wider range of tasks.

Another important issue concerns modelling objects which rotate uniformly through a full  $360^\circ$ ; whether or not it is ‘correct’ to have an arbitrary mean position with  $180^\circ$  variation on either side.

Whilst the hybrid model was designed with pivotal or bending deformation in mind, it is possible that it can approximate other types of deformation too. More importantly, the concept of a hybrid model can be extended to incorporate other types of reparameterisation — such as polynomial, elliptical or sinusoidal — simply by providing suitable mapping functions.

It is clear, however, that there are many classes of deformable objects that cannot be effectively modelled using the Hybrid PDM. Also, the Hybrid PDM is in some sense a hand-crafted model; there is an *explicit* representation of angular deformation, and this is counter to the desire to construct models of arbitrary objects *fully automatically*. In the next chapter a technique which is able to model a wider class of deformations is described.

# Chapter 4

## Hierarchical Statistical Shape Models

---

### 4.1 Introduction

Consider the construction of a PDM of a human hand which can perform only three gestures: a flat palm, a fist or a pointing gesture. The training set would consist of the three gestures and the transitions between them, as illustrated in Figure 4.1. Each outline consists of 100 evenly-spaced landmarks, extracted automatically from the images using a simple boundary-finding algorithm.

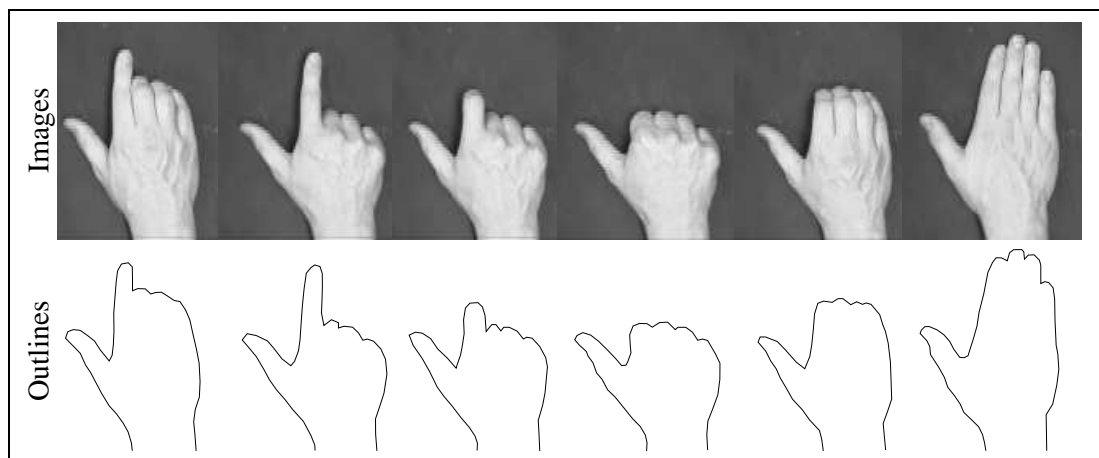


Figure 4.1: Example training data for a three gesture hand model.

A PDM can be constructed from this training data. Figure 4.2 shows projections of the distribution of the training shapes transformed into such a PDM's shape space. As would be expected, the data forms a hollow triangle, with each vertex corresponding to one of the gestures and the edges being the transitions. The triangle is also non-planar, bending through the third dimension of shape space.

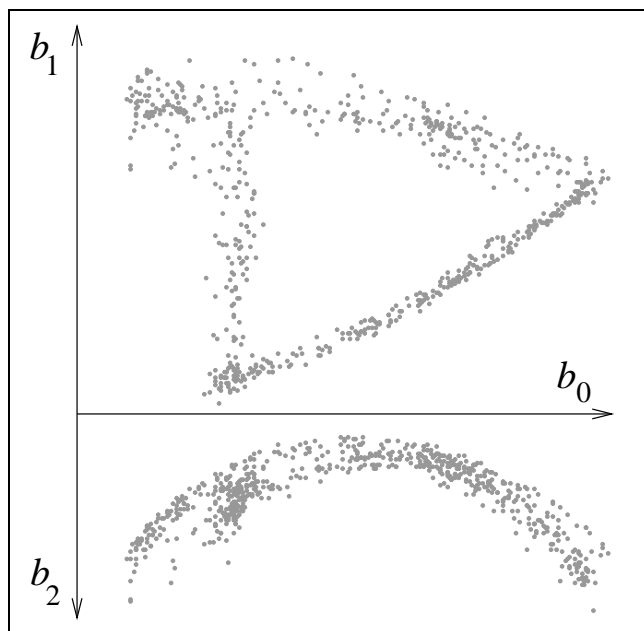


Figure 4.2: Projections of the ‘three gesture’ training data in PDM shape space.  $b_0$ ,  $b_1$ , and  $b_2$  are the three most significant axes of the space.

The region in shape space which corresponds to *valid* shapes (the valid shape region, or VSR, as defined in Section 2.2.1.1) in this case occupies only a very small proportion of the shape space. The model produced is not *specific* and is thus capable of generating invalid shapes as well as valid ones (see Figure 4.3). Even the mean shape is invalid, and although the invalid shapes are not drastically malformed in this case, in a more complex example they might well be.

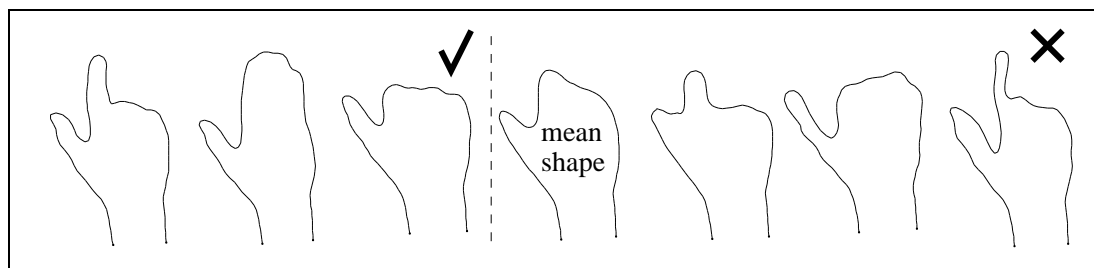


Figure 4.3: Example shapes produced by a standard PDM on the ‘three gesture’ model; good (left) and poor (right).



Whilst it may be possible to linearise the bending seen in the plot of  $b_2$  against  $b_0$  (perhaps using one of the techniques outlined in Section 2.2.1.2), none of the approaches developed so far can cope with a hollow loop topology such as that seen in the plot of  $b_1$  against  $b_0$ . Another situation which causes difficulty is where an object can take on two or more distinct shapes, but *not* any of the shapes in between. This results in a VSR with two or more disjoint parts.

A method is required for modelling VSRs which have an *arbitrary* shape and topology.

## 4.2 Constraints in Shape Space

Bregler and Omohundro describe the use of shape space *constraint surfaces* for modelling complex VSRs [12]. The technique is described more fully in section 2.2.2, but the essence is the use of a piecewise-linear approach. The shape space is split into a number of regions using  $k$ -means cluster analysis on the training data. A principal component analysis (PCA) is performed separately on each cluster to produce a number of linear subspaces with arbitrary orientation and dimensionality. The VSR is represented as a complex, non-linear *surface*, constructed from a combination of these linear patches. To apply shape space constraints to a particular shape instance, a weighted sum of the projections into each subspace is used (see Equation 2.5).

Bregler and Omohundro give examples of the technique applied to synthetic data, and also demonstrate its use in building shape models of human lips [12]. The modelling of complex topology VSRs is possible, including non-linearities, loop-like structures, changing dimensionality and (theoretically) disjoint regions, although the latter is not demonstrated.

The use of a piecewise-linear approach is promising, however constraint surface models suffer from the following problems:

- The linear patches are modelled as *hyperplanes* embedded in the shape space. This has the following disadvantages:
  - The hyperplanes are not limited in extent; they stretch off to infinity. When modelling a continuous smooth surface this is not a problem, but at discontinuities or extremities, the hyperplanes cause the surface to be ‘extended’ to infinity.
  - The model cannot cope with surfaces of finite thickness; each hyperplane dimension is either infinite or zero.

- The dimensionality of each hyperplane is determined via an arbitrary cut-off point; this may need to be fine-tuned for a particular problem.
- The cluster analysis and PCA (and hence all constraint calculations) are performed in a very high dimensional space (2 times the number of model landmarks). This makes for much slower computation than, for example, a PDM.

These problems are illustrated in Figure 4.4(a) below.

### 4.3 A Hierarchical PDM

Proposed here is an approach to modelling complex, non-linear shape spaces, which is related to the constraint surface models described above. The PCA-based piecewise-linear approach is still used, but there are two important differences:

- The linear patches are represented as hyperellipsoid-bounded regions as opposed to hyperplanes. This seems more natural, given that the PCA is effectively finding a best-fit Gaussian for each patch. The axes for each hyperellipsoid are the principal components from the corresponding PCA, and the size is proportional to the significance of each component (i.e. the hyperellipsoid is a surface of fixed Mahalanobis radius).
- A two-level hierarchy of PCAs is used. The first level is a global PCA, identical to that used for a standard PDM. The second level introduces the piecewise-linear element.

The first of these points leads to the most noticeable improvements over Bregler and Omohundro's approach. The use of finite-sized hyperellipsoid-bounded regions removes three problems in one: extremities in the VSR are no longer extended to infinity, finite thicknesses can be modelled and there is no need to choose an arbitrary cut-off point to determine the dimensionality for each patch. Figure 4.4 illustrates these improvements.

The benefits of using a two-level hierarchical approach are more subtle. The main advantage is the decrease in computation time resulting from performing the constraint calculations in a much lower dimensional shape space (for example, in the examples we show later, the dimensionality is reduced from 200 to 16). Another advantage is that the initial global PCA helps to remove noise in the training data

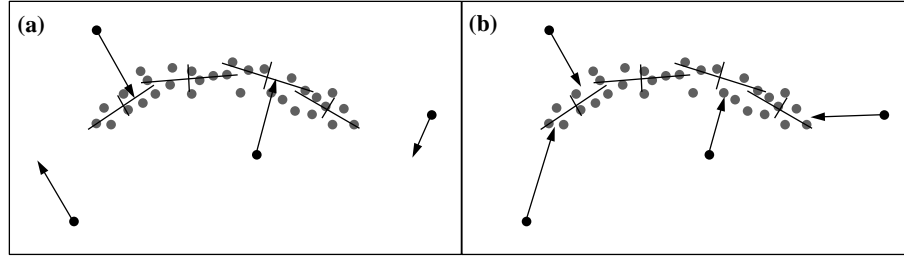


Figure 4.4: Constraining shape using (a) hyperplanes and (b) hyperellipsoids. The grey spots are the training data and the arrows show how four general points are constrained under each model. The hyperplane approach cannot model finite thicknesses and also extends the VSR at extremities.

caused by outliers, which can otherwise greatly affect the calculation of the linear patches.

To summarise, the Hierarchical PDM, or HPDM, produces a VSR which is both more accurate and more efficient in application than a constraint surface model.

### 4.3.1 Implementation Details

Given a set of  $E$  pre-aligned training shape vectors  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_E\}$ , where  $\mathbf{x}_e = (x_{e,1}, y_{e,1}, \dots, x_{e,N}, y_{e,N})$  contains the Cartesian coordinates of each of the  $N$  landmarks for shape  $e$ , an HPDM can be constructed as follows:

1. Perform a PCA on *all* the training data, to find the origin  $\bar{\mathbf{x}}$  and the  $t$  most significant axes  $\mathbf{p}_1 \dots \mathbf{p}_t$  of the global PCA space (these are exactly equivalent to the mean and modes of variation respectively for a linear PDM). In general,  $t \ll 2N$ , giving a significant dimensional reduction.
2. Transform each training vector  $\mathbf{x}_e$  into its global PCA space equivalent  $\mathbf{b}_e$ .

$$\mathbf{b}_e = \mathbf{P}^T(\mathbf{x}_e - \bar{\mathbf{x}}) \quad (4.1)$$

where  $\mathbf{P} = (\mathbf{p}_1 \dots \mathbf{p}_t)$ .

3. Decide on a value for  $k$ , the number of linear patches to use.

The choice of  $k$  is data-dependent. The more complex and non-linear the VSR, the larger the number of linear patches required to model it accurately. However, there is a trade-off between accuracy and speed. If speed is not an issue then  $k$  can be increased in the limit to  $E$ . For noiseless training data

increasing  $k$  produces the smoothest model; however any noise that *is* present is liable to be included in the model.

In these experiments  $k$  has been chosen manually, based on knowledge about the expected shape of the VSR. It seems likely that it would be possible to find a sensible value for  $k$  automatically via some optimisation technique. Experimentation suggests that a good first guess would be  $k = E/10$ .

4. Perform  $k$ -means cluster analysis on the training data in global PCA space, to produce  $k$  exemplars  $\mathbf{e}_1 \dots \mathbf{e}_k$ .
5. For each exemplar  $\mathbf{e}_i$ :

- (a) Decide a number  $n_i$  of nearest neighbours to use in the local PCA.

Our current strategy is to specify a fixed degree of cluster *overlap*,  $O$ , and set  $n_i = Om_i$ , where  $m_i$  is the number of members in the corresponding  $k$ -means cluster. The argument for overlap is that it results in smoother transitions between the linear pieces. Initial experimentation suggests that  $O = 1.5$  produces a good balance between accuracy (allowing all valid shapes) and specificity (disallowing invalid shapes).

- (b) determine this set of nearest neighbours, using the distance metric  $|\mathbf{b}_j - \mathbf{e}_i|$ .

- (c) Perform a local PCA on these  $n_i$  training examples (the global PCA space versions  $\mathbf{b}_j$ , as opposed to the original versions  $\mathbf{x}_j$ ), to produce a linear patch, defined in terms of its origin  $\bar{\mathbf{b}}_i$ , axes  $\mathbf{p}_{i1} \dots \mathbf{p}_{i t_i}$  and associated variances  $\lambda_{i1} \dots \lambda_{i t_i}$  (these are the eigenvalues produced by the PCA). Note that in general,  $\bar{\mathbf{b}}_i \neq \mathbf{e}_i$ .

The values for  $t$  and  $t_1 \dots t_k$  are determined exactly as for the linear PDM [15], by ensuring that a sensible proportion (at least 90%) of the shape variation has been captured.

### 4.3.2 Using the Hierarchical PDM

For our representation of the VSR to be of practical use, it must be possible to apply a ‘nearest point’ query: “Given a general point in shape space, where is the nearest point in the VSR?” This translates as “Given a general shape  $\mathbf{x}$ , what is the nearest *valid* shape  $\mathbf{x}'$ ?” and hence facilitates the application of constraints to ensure that only valid shapes are allowed in the context of tracking.

For the HPDM, applying such constraints is a two-stage process, corresponding to the two levels in the PCA hierarchy. First it is necessary to convert the shape  $\mathbf{x}$  (as described by the  $x$  and  $y$  coordinates of its landmarks) into a corresponding vector  $\mathbf{b}$  in global PCA space exactly as for the training data (see Equation 4.1):

$$\mathbf{b} = \mathbf{P}^T(\mathbf{x} - \bar{\mathbf{x}}) \quad (4.2)$$

Following this, the piecewise-linear model constraints must be applied.

The simplest (but not only) way to apply these stage-two constraints is to find the *nearest* linear patch (using the Euclidian distance metric  $|\mathbf{b} - \bar{\mathbf{b}}_i|$ , where  $\bar{\mathbf{b}}_i$  is the origin for patch  $i$ ), and constrain  $\mathbf{b}$  to lie within the associated hyperellipsoid-bounded region. Finding the closest position within such a region involves a time-consuming gradient descent computation; however a good approximation is to consider the region to be a hypercuboid. The other (fast) alternative of moving the point directly towards the patch origin is grossly inaccurate in the case of eccentric hyperellipsoids. Figure 4.5 illustrates the different possibilities.

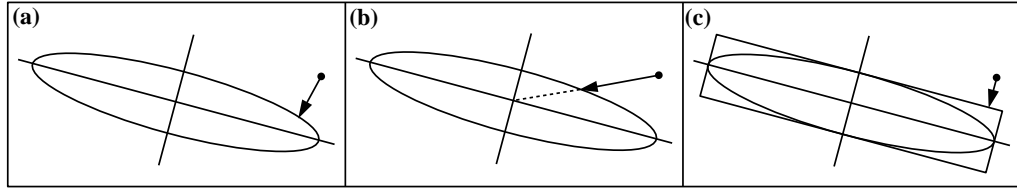


Figure 4.5: Constraining a general point to lie within a hyperellipsoid-bounded region; (a) the correct way, (b) a bad approximate method and (c) a better approximate method.

The function  $C_i(\mathbf{b})$  for constraining into linear patch  $i$  is defined as follows:

$$C_i(\mathbf{b}) = \mathbf{P}_i L_i[\mathbf{P}_i^T(\mathbf{b} - \bar{\mathbf{b}}_i)] + \bar{\mathbf{b}}_i \quad (4.3)$$

where  $\bar{\mathbf{b}}_i$  is the origin for patch  $i$ ,  $\mathbf{P}_i = (\mathbf{p}_{i1}, \dots, \mathbf{p}_{it_i})$  is a matrix of the axes for patch  $i$ , and  $L_i$  is the *limiter* function for patch  $i$ , which, in the case of our hypercuboid approximation, is defined thus:

$$\begin{aligned} L_i[(y_1, y_2, \dots, y_{t_i})^T] &= (y'_1, y'_2, \dots, y'_{t_i})^T \\ \text{where } y'_j &= -M\sqrt{\lambda_{ij}} & \text{if } y_j < -M\sqrt{\lambda_{ij}} \\ y'_j &= y_j & \text{if } -M\sqrt{\lambda_{ij}} < y_j < M\sqrt{\lambda_{ij}} \\ y'_j &= M\sqrt{\lambda_{ij}} & \text{if } y_j > M\sqrt{\lambda_{ij}} \end{aligned} \quad (4.4)$$

The  $\lambda_{ij}$  are the significances for each axis and  $M$  is the Malhalanobis radius of the linear patches. We use  $M = 2.0$  which, statistically, should encompass over 95% of valid shapes. A larger value generally leads to an underconstrained VSR.

A more general stage-two constraint function  $C$  on a shape  $\mathbf{b}$  can be defined as follows:

$$C(\mathbf{b}) = \frac{\sum_i G_i(\mathbf{b})C_i(\mathbf{b})}{\sum_i G_i(\mathbf{b})} \quad (4.5)$$

where  $C_i(\mathbf{b})$  is as defined in (4.3) and  $G_i$  is the *influence* function for patch  $i$ . Setting  $G_i = 1$  if patch  $i$  is closest and  $G_i = 0$  otherwise results in the simple nearest-patch algorithm described above. Alternatively,  $G_i$  can be a Gaussian, centred on patch  $i$ 's origin:

$$G_i(\mathbf{b}) = \exp\left(\frac{\|\mathbf{b} - \bar{\mathbf{b}}_i\|^2}{\sum_j \lambda_{ij}}\right) \quad (4.6)$$

where  $\bar{\mathbf{b}}_i$  is the cluster mean and the denominator is a scale factor related to the overall ‘size’ of cluster  $i$ . It is important to note that in the above equation, the *Euclidian* distance (as opposed to the Malhalanobis distance) is used; otherwise the influence function decays too quickly off-axis for eccentric hyperellipsoids.

For a particular  $\mathbf{b}$ ,  $G_i(\mathbf{b})$  will be very small for the majority of  $i$ . The calculation of  $C(\mathbf{b})$  can be made more efficient by only including terms for which  $G_i(\mathbf{b})$  is significant. A cut-off point of one tenth of the maximum value is suitable.

Using the Gaussian influence functions has the effect of performing an interpolation at positions between neighbouring patches, giving smoother joins, especially in cases where the patches do not actually overlap. (see Figure 4.6). However, a side effect is that the notion of a concrete divide between valid and invalid shapes is lost, insofar that if  $C(\mathbf{b}) = \mathbf{b}'$  then it is not necessarily the case that  $C(\mathbf{b}') = \mathbf{b}'$ . It is thus important to apply the constraint function only *once* each time the shape needs constraining. The other disadvantage is that this method is slower than the nearest-patch method.

Whichever constraint function is used, finding the ‘valid’ shape  $\mathbf{x}'$  is a matter of transforming  $\mathbf{b}'$  out of global PCA space:

$$\mathbf{x}' = \mathbf{P}\mathbf{b}' + \bar{\mathbf{x}} \quad (4.7)$$

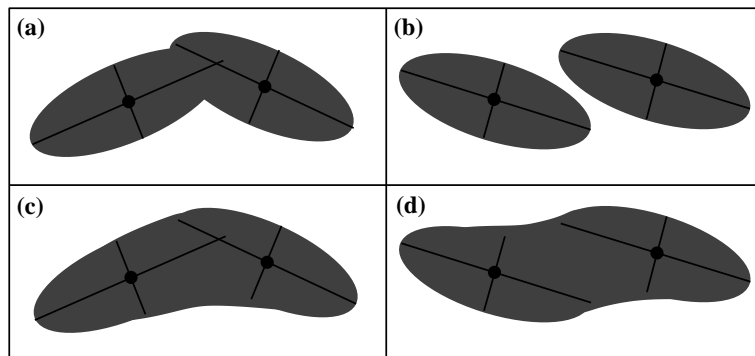


Figure 4.6: The valid shape regions produced under two different constraint algorithms; (a), (b) the ‘nearest cluster’ algorithm and (c), (d) the Gaussian combination algorithm.

## 4.4 Evaluation

We examined four different modelling tasks. A synthetic anglepoise lamp model was used for a comparison with the linear PDM and for various other quantitative tests. Synthetic 2D shape spaces were constructed for a comparison with constraint surface models. Models of human hands were built from both manually and automatically collected training data in order to demonstrate performance on real world problems.

### 4.4.1 Synthetic Anglepoise Lamp

As in Chapter 3, an anglepoise lamp was used as an example of an object for which the linear PDM produces poor models, however in this experiment the training shapes were synthesised. The lamp shape was represented in 2D using 49 landmarks. Training examples were generated by choosing uniformly-distributed random values for the three pivot angles (see Figure 4.7). A global PCA was performed. Figure 4.8 shows scattergrams of the position in global PCA space of 500 training examples and Figure 4.9 shows the three most significant modes of variation for the corresponding PDM. As can be seen, even along the principal axes several invalid shapes have been generated.

A Hierarchical PDM was then constructed from the same training data. Figure 4.10 shows the training set with the linear constraint patches superimposed, giving some idea of the VSR that has been learned. The concept of a mode of variation does not exist within the context of a HPDM; the nearest equivalent is to ‘drag’ the model through shape space, whilst applying shape constraints, to produce shape space *traversals*. This is achieved by fixing one of the global PCA parameters

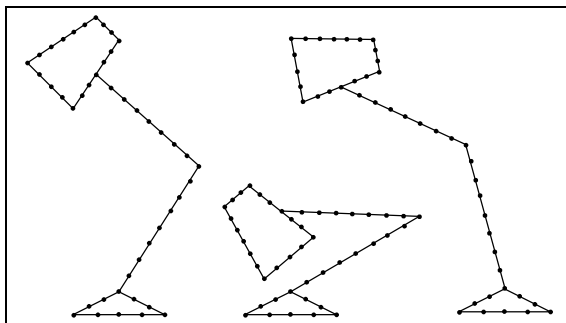


Figure 4.7: Three examples from the synthetic anglepoise lamp training set.

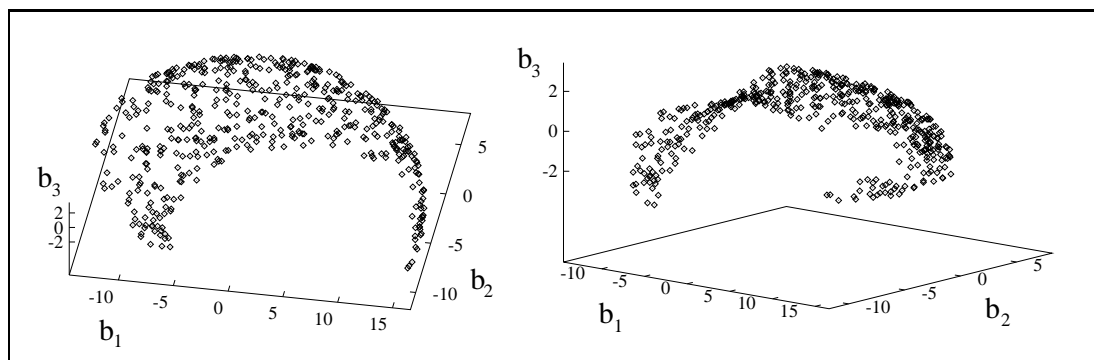


Figure 4.8: Two views of the lamp model training data transformed into global PCA space.

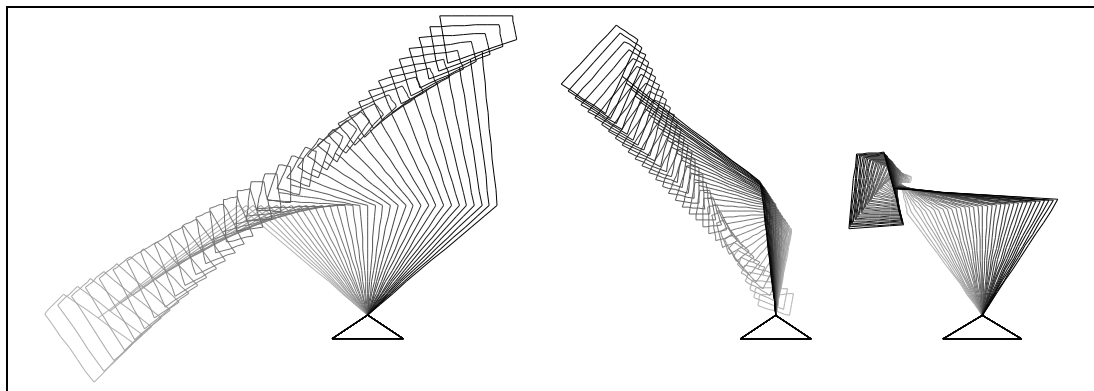


Figure 4.9: The three most significant modes of variation of the *linear* lamp PDM. Many invalid shapes can be seen.

at uniform increments and applying the constraint function to obtain suitable values for all the other parameters. Figure 4.11 illustrates three such traversals. The results are much improved over the linear PDM; points are seen to move along arcs, not straight lines, and for the most part the lamp head remains a constant size.

A quantitative analysis was undertaken in order to compare the performance of the HPDM with previous approaches, to determine the optimum choice of various



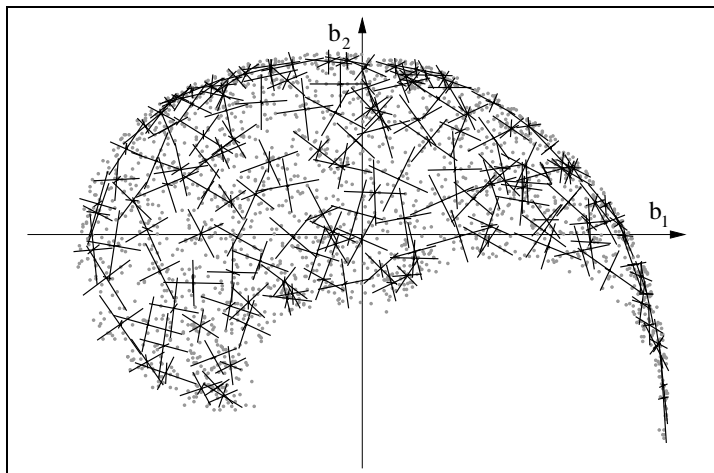


Figure 4.10: The lamp model global PCA space (2D projection), showing training data and principal component axes for the constraint patches.

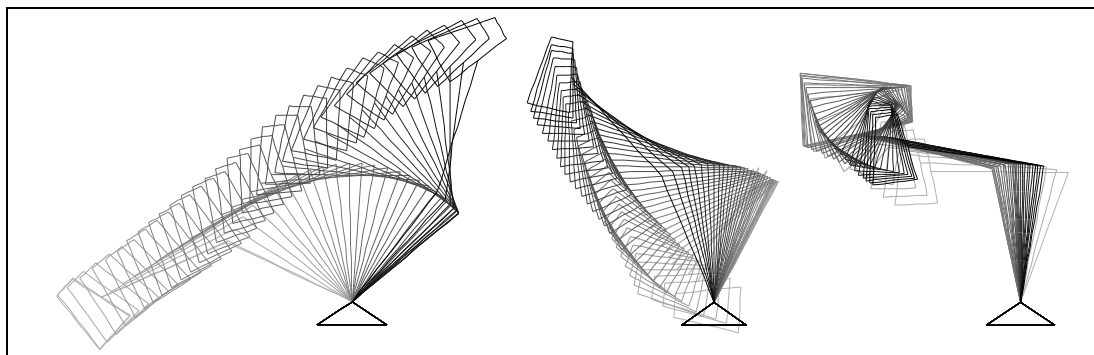


Figure 4.11: Three traversals through the VSR for the lamp HPDM.

parameters and to see how performance varies with the number of training examples provided. To this end, quantitative measurements have been devised for model *accuracy* and model *specificity*.

Model *accuracy* measures a model's ability to allow *valid* object shapes to be modelled without distortion. This is also a test of a model's ability to generalise from the training data as it involves testing on *unseen* shapes. To measure the degree of accuracy:

1. Generate or collect a large number  $L$  of valid shapes  $\mathbf{x}_1 \dots \mathbf{x}_L$ , where  $\mathbf{x}_e = (x_{e,1}, y_{e,1}, \dots, x_{e,N}, y_{e,N})$ , and ensure they are correctly aligned.
2. Apply shape constraints to each one to produce  $\mathbf{x}'_1 \dots \mathbf{x}'_L$ .
3. Find the average landmark displacement  $D$  over all the shapes:

$$D = \frac{1}{L} \sum_{e=1}^L \frac{1}{N} \sum_{i=1}^N \sqrt{(x'_{e,i} - x_{e,i})^2 + (y'_{e,i} - y_{e,i})^2} \quad (4.8)$$

4. Express  $D$  as a percentage of the average model size:

$$\hat{D} = \frac{D \times 100}{\frac{1}{L} \sum_{e=1}^L (\max_i |y_{e,i} - \min_i |y_{e,i}|)} \quad (4.9)$$

Model *specificity* measures a model's ability to exclude invalid shapes from the VSR. This directly affects robustness of tracking. The notion is to generate random shapes, apply constraints and then see how far away from ground truth the constrained shapes are. Ground truth is approximated via a large number of valid shapes, since calculating the analytical ground truth is not always possible. The algorithm proceeds as follows:

1. Generate a large number  $L$  of *entirely random* positions in global PCA space  $\mathbf{b}_1 \dots \mathbf{b}_L$ .
2. Apply shape constraints to each one to produce  $\mathbf{b}'_1 \dots \mathbf{b}'_L$ .
3. Generate a large number  $K$  of *valid* shapes  $\mathbf{y}_1 \dots \mathbf{y}_L$  for use as ground truth, and ensure they are correctly aligned.
4. Transform the ground truth shapes into global PCA space, using (4.2), to produce  $\mathbf{c}_1 \dots \mathbf{c}_L$ .
5. For each test shape  $\mathbf{b}'_e$ , find the distance  $t_e$  to the nearest ground truth shape:

$$t_e = \min_j |\mathbf{b}'_e - \mathbf{c}_j| \quad (4.10)$$

6. The specificity error is defined as the 90th percentile of the  $t_e$  values. This gives a measure of the maximum possible distance from ground truth, whilst avoiding statistical outliers.

Figure 4.12 shows accuracy and specificity error graphs for HPDMs of the angle-poise lamp with varying numbers of linear patches and varying degrees of overlap. The degree of overlap is fixed for each curve shown, so as the number of patches increases, the size of each patch decreases accordingly.

In all cases, accuracy error decreases up to about 30 patches and then increases monotonically. The monotonic increase is due to the successive decrease in patch size

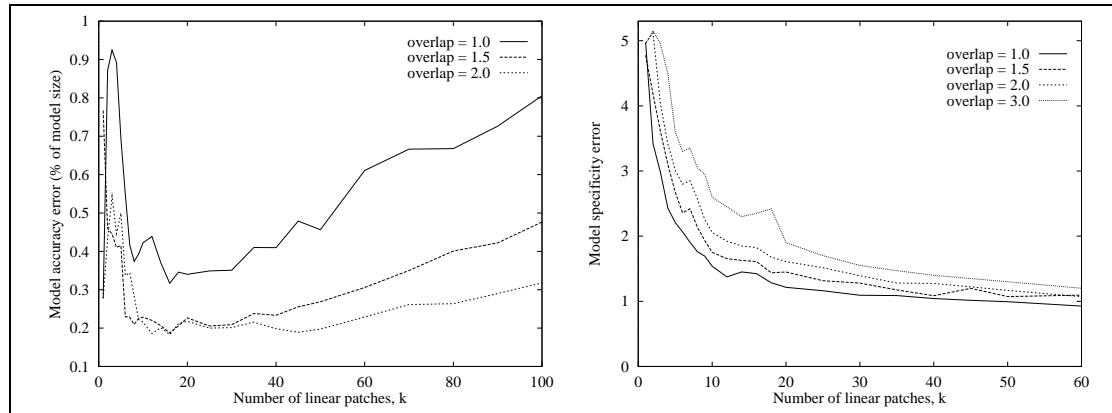


Figure 4.12: Accuracy (left) and specificity (right) error graphs for anglepoise lamp HPDMs with varying numbers of linear patches and degrees of overlap (see text).

resulting in decreased accuracy in the local PCAs. As would be expected, specificity error decreases as the number of patches increases: the model becomes more specific as the non-linear VSR is better approximated with more linear pieces. In this case, there is little to be gained by having more than 30 patches, especially as this incurs both degraded accuracy and a speed penalty.

An increase in overlap effectively increases the size of each linear patch. As expected, this results in an increase in accuracy but a decrease in specificity. In this case it seems that an overlap of 1.5 is optimal as the improvement in accuracy above this value is small, but lowering it would hamper specificity.

Figure 4.13 shows how accuracy and specificity compares under the various different modelling techniques.

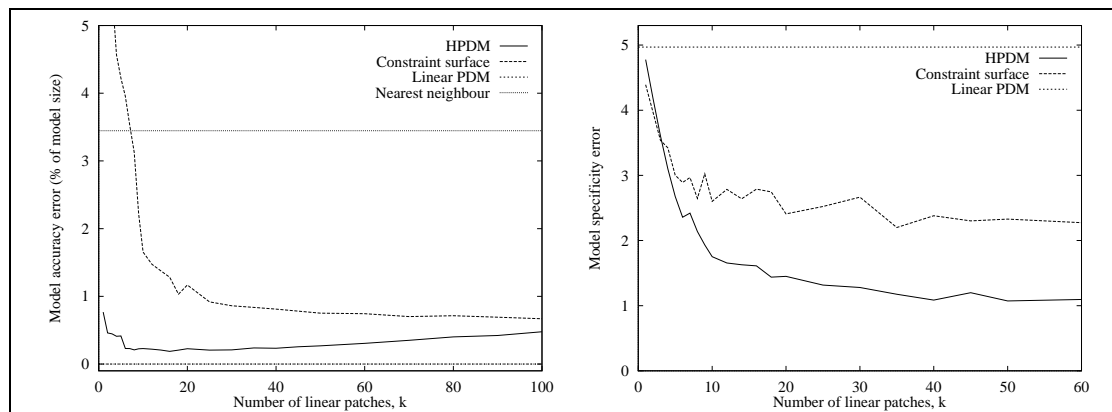


Figure 4.13: Accuracy (left) and specificity (right) error graphs for anglepoise lamp models built under the various modelling techniques (see text).

The most accurate model is the linear PDM, with zero accuracy error; this is

because the linear PDM shape space covers the whole of the VSR (which in the case of the lamp model is only 6 dimensional). The HPDM and constraint surface models introduce extra constraints within this space and thus can potentially create accuracy error by excluding parts of the VSR. Both the HPDM and the constraint surface model have an accuracy error of less than 1% when over 30 linear patches are used; this is sufficient for tracking purposes. A ‘nearest neighbour’ accuracy plot is also shown; this involves constraining a shape by moving it to the nearest training example shape. Both the constraint surface and the HPDM out-perform the nearest neighbour algorithm, suggesting that they have learned to *generalise* the training set, and should thus perform well on unseen data.

The specificity graph clearly shows that the HPDM out-performs both the linear PDM and the constraint surface model. At  $k = 20$  its specificity error is approximately half that of the constraint surface and a quarter that of the linear PDM.

Figure 4.14 shows how the size of the training set affects the accuracy of the lamp HPDM. Values of  $k = 20$  and  $O = 1.5$  were used.

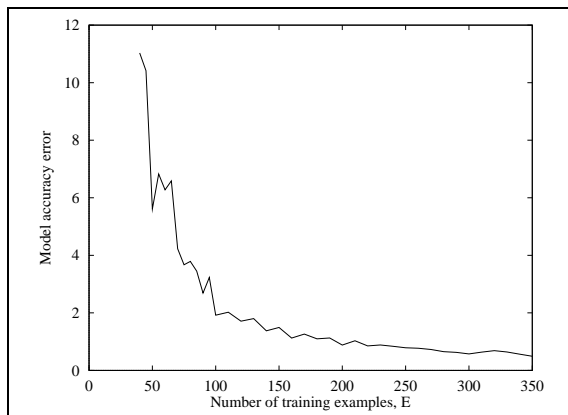


Figure 4.14: Effect of training set size on the accuracy of the lamp HPDM.

As might be expected, accuracy improves with an increase in training set size. The graph begins to level out at around  $E = 100$ , and by  $E = 200$  the error is below 1%.

Intuitively, a HPDM will require more training data than a linear PDM. Since a HPDM with  $k$  linear patches is effectively  $k$  separate PDMs, then arguably each patch should have enough training data to build a satisfactory PDM. If the overlap is  $O$  then a HPDM should notionally require  $k/O$  times as many training examples as a linear PDM. Of course, this assumes that all patches are equally well represented, which is rarely the case in practice, so the true figure may be much higher. The inescapable truth is that the HPDM performs poorly given sparse training

data; models produced under these conditions are *too* specific, often excluding valid shapes.

#### 4.4.2 Other Synthetic Examples

In this section we show some synthetic examples which illustrate how the HPDM provides an improvement over constraint surface models.

In Figure 4.15, the first column shows two different sets of training data; a circle and a ‘U’ shape, both of finite thickness. The second column shows two corresponding sets of test data; this data is to be constrained under each model in order to illustrate the model’s performance. Three HPDMs and three corresponding constraint surface models were built: the circle data was modelled with both 10 and 15 linear patches, and the ‘U’ shape was modelled with 15 patches. Columns 3 and 4 show the results of constraining the test data under the various models.

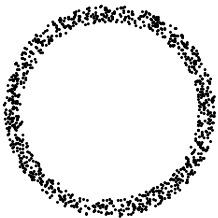
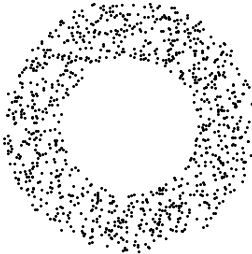
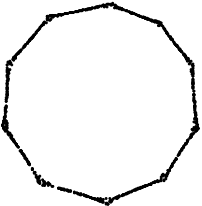
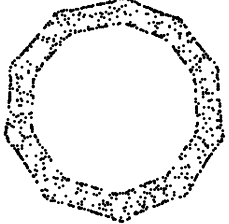
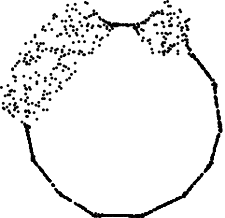
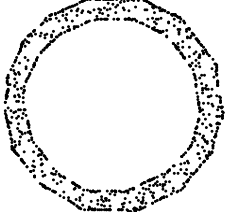

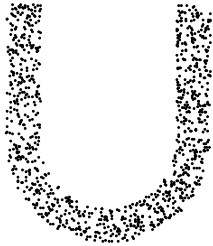
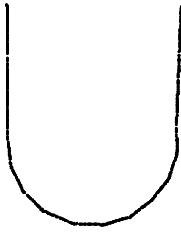

Training data	Test data		Constraint surfaces	Hierarchical PDM
		10 patches		
		15 patches		
		15 patches		

Figure 4.15: Comparison of the HPDM with constraint surface models (see text).

In the 10-patch circle model, the constraint surface model has approximated the circle as a one-dimensional surface; consequently the VSR produced is too thin. The

HPDM has correctly captured its finite thickness. In both models there is a degree of polygonisation of the circle. Increasing the number of patches to 15 improves the HPDM, but the smaller patches in this case have caused the constraint surface to interpret parts of the circle as two-dimensional, and the VSR is under-constrained. The ‘U’ shape model illustrates how the Constraint Surface tends to extend the VSR at extremities (as described in Section 4.2), whereas the HPDM correctly limits its extent.

### 4.4.3 Manually Collected Real Data

The first real data model built using the HPDM was a 2D multi-gesture hand model. 105 examples of hands in five different poses (open, fist, point (thumb out), point (thumb in) and crossed fingers) were each annotated manually with 89 landmarks around the boundary<sup>1</sup>. Figure 4.16 shows some examples.

A HPDM was built using 20 clusters and an overlap factor of 1.5. Figure 4.17 shows a scatter graph of the training data projected into the first two dimensions of global PCA space (left) and the calculated linear patches (right). Figure 4.18 compares deformations for the linear PDM (top row) and HPDM (bottom row).

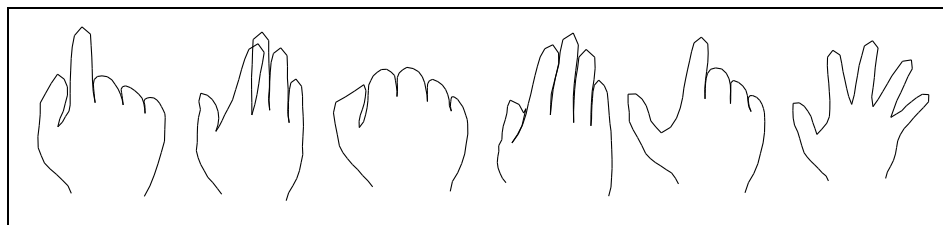


Figure 4.16: Manually annotated training examples for a hand model.

Figure 4.17 shows that the VSR for the hand model features a loop. The HPDM has captured this topology with a series of overlapping patches. It is less obvious from Figure 4.18 than for the lamp example that the HPDM has more accurately captured the VSR, however, Figure 4.19 shows some maximally non-valid shapes<sup>2</sup> both from the linear PDM (top row) and the HPDM (bottom row). The linear PDM is capable of producing a wide variety of implausible shapes, but the HPDM’s *worst* shapes are still all sensible.

<sup>1</sup>Thanks are due to Andreas Lanatis of the Wolfson Image Analysis Unit, University of Manchester, for providing the data.

<sup>2</sup>Maximally non-valid shapes are shapes with large *specificity* errors.

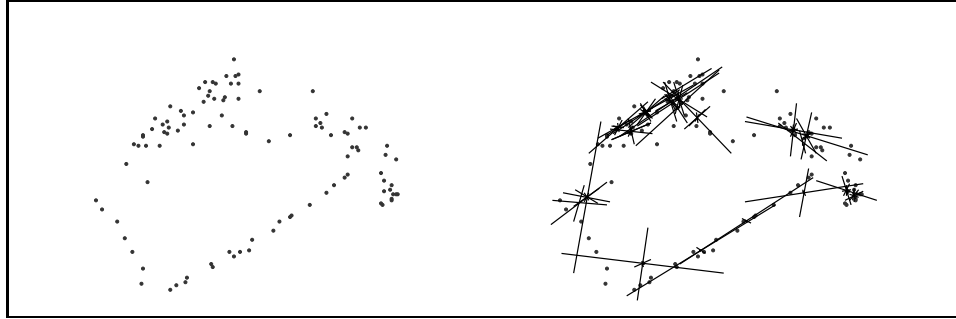


Figure 4.17: The manually annotated hand training data projected into the first two dimensions of global PCA space (left) and the HPDM linear patches (right).

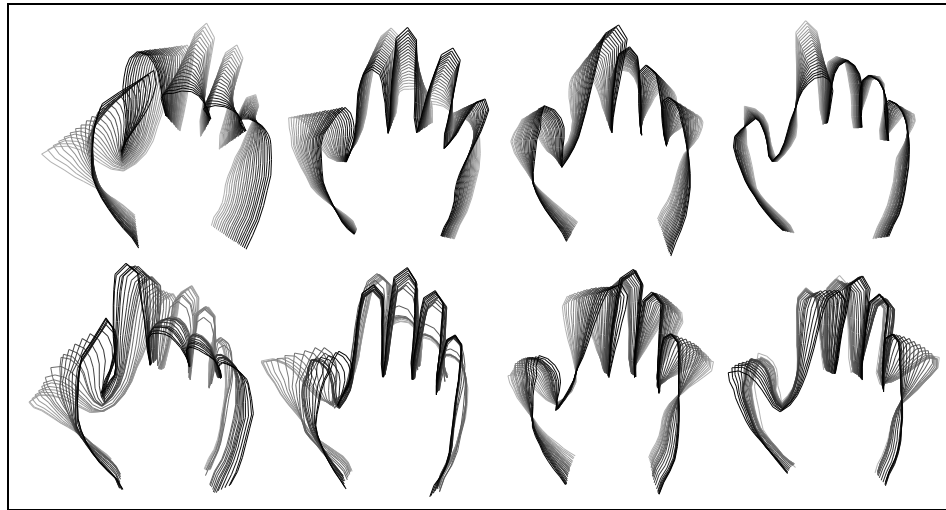


Figure 4.18: Modes of variation for the manually annotated hand PDM (top row) and equivalent HPDM shape space traversals (bottom row).

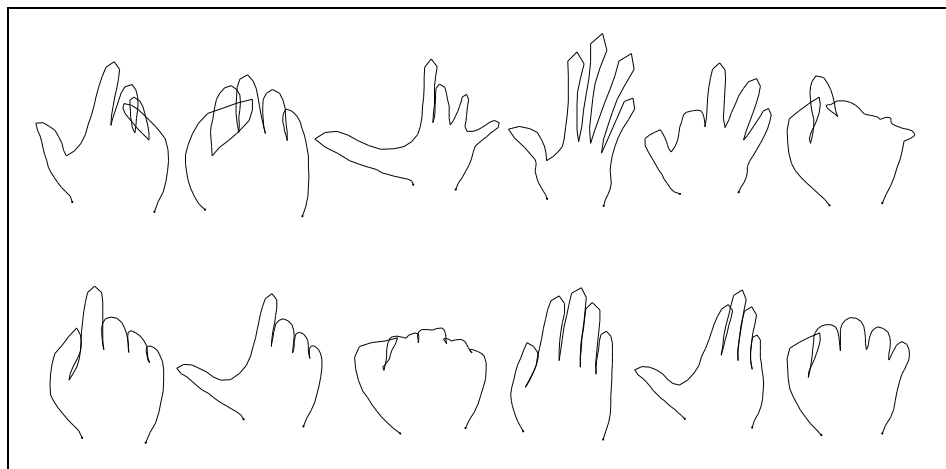


Figure 4.19: Maximally non-valid shapes for the manually-annotated hand model; linear PDM (top row) and HPDM (bottom row).

#### 4.4.4 Automatically Collected Real Data

The main motivation for this work was the desire to build models from automatically collected training data. In this experiment, hand shapes were sampled directly from a live video stream. Various gestures were performed against a black background; the image was thresholded and the hand outline extracted using a simple boundary-finding algorithm. 100 landmarks were positioned at equal intervals around the boundary. This method of data collection suffers greatly from the problem that landmarks rarely mark the same physical object feature across training examples. For example, when the fingers are outstretched the boundary is much longer than for a pointing gesture; the landmarks spread out more and tend to ‘slide’ around the boundary. There were just over a thousand training shapes in all; Figure 4.20 shows some examples.

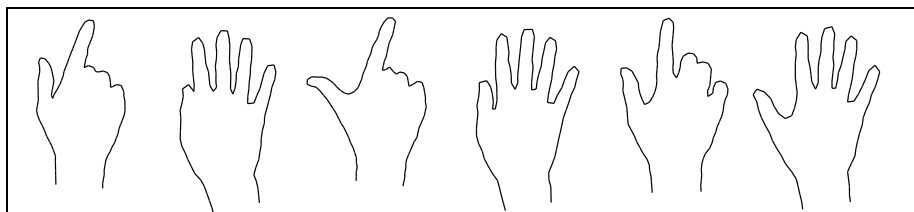


Figure 4.20: Automatically collected training examples for a hand model.

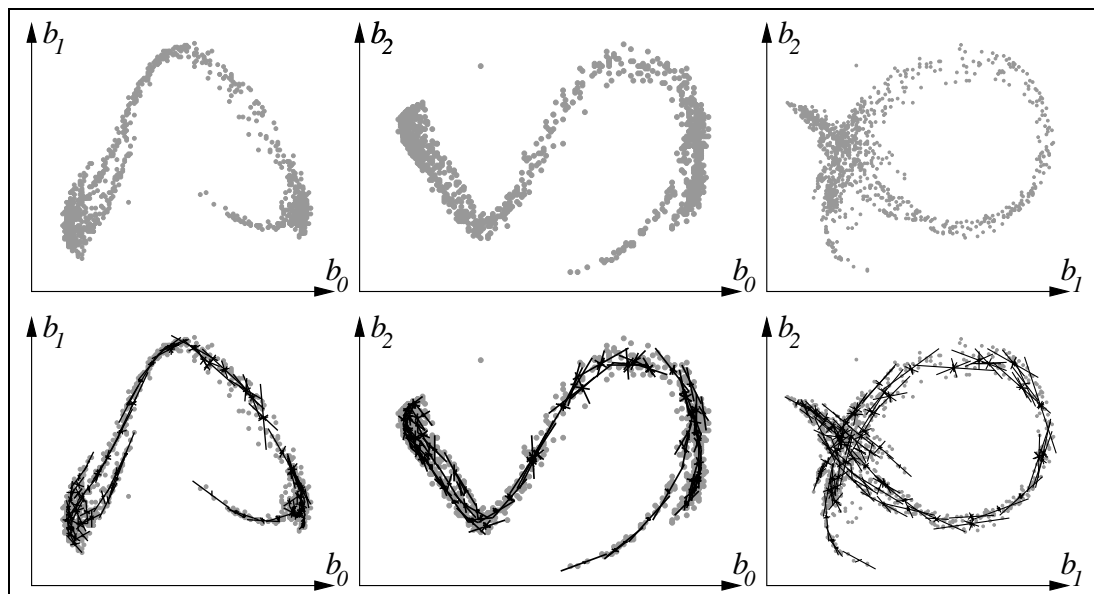


Figure 4.21: Several projections of the automatically collected hand training data in global PCA space (top row) and HPDM linear patches (bottom row).



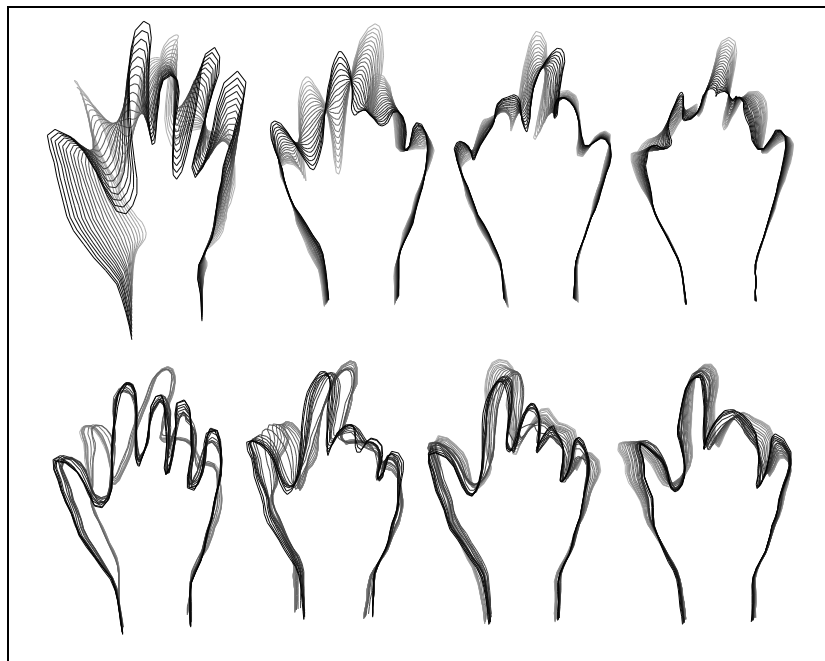


Figure 4.22: Modes of variation for the automatically trained hand PDM (top row) and equivalent HPDM shape space traversals (bottom row).

A HPDM was constructed from the training data, using 80 linear patches. Figure 4.21 shows several projections of the training data in the global PCA space, along with the patches calculated, and Figure 4.22 shows the four major modes of variation for a linear PDM (top row) and the four equivalent shape space traversals for the HPDM (bottom row).

Figure 4.21 clearly illustrates that the training data is virtually one-dimensional in nature, representing transitions between the various gestures. However, the paths through the global PCA space are highly non-linear, spiraling through at least 3 dimensions.

Figure 4.22 demonstrates how, in this case, the linear PDM fails to produce a model which would be specific enough for object tracking or location. The HPDM traversals include only valid object shapes. There appear to be discontinuities in some of the traversals; this is expected because the VSR is not necessarily continuous parallel to any one axis in the global PCA space, and is a side-effect of the ‘dragging’ technique used to generate the traversals.

Figure 4.23 shows some maximally non-valid shapes. As can be seen, the linear PDM is capable of producing shapes which are barely even recognisable as hands, whereas the HPDM’s *worst* shapes are only slightly distorted versions of plausible hand shapes.

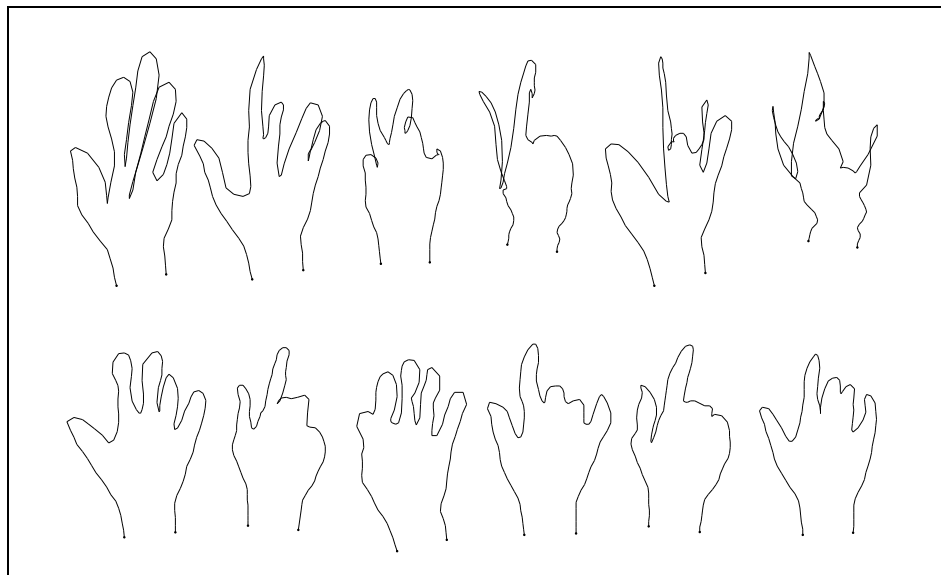


Figure 4.23: Maximally non-valid shapes for the manually-annotated hand model; linear PDM (top row) and HPDM (bottom row).

In terms of quantitative analysis; Figure 4.24 shows how the automatically-trained hand HPDM's specificity compares to the linear PDM, and how it varies with the number of clusters used<sup>3</sup>.

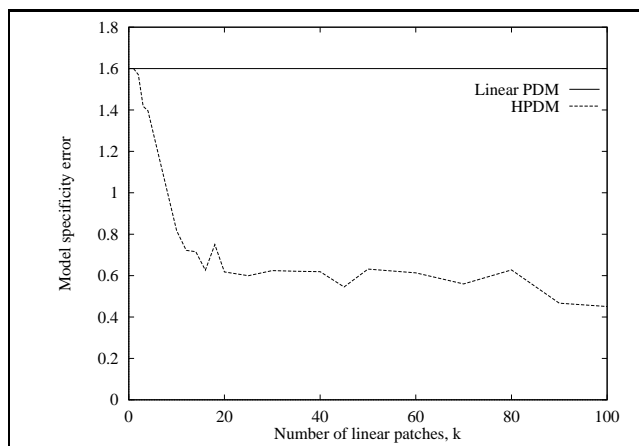


Figure 4.24: Graph showing how the specificity of the hand HPDM compares with a linear PDM, and how it varies with the number of clusters used.

The trend is very similar to that of the synthetic examples. The graph also shows that in this case there is not much improvement in specificity gained by using 80 clusters (as we did) over 20 clusters.

<sup>3</sup>An overlap factor of 1.5 was used for this experiment.

## 4.5 Tracking using HPDMs

The chief test of the success of a HPDM is whether it can be used to good effect for object location or tracking. Tracking with a HPDM is straightforward, being the same as for the linear PDM (i.e. using Active Shape Models, or ASMs [20]), but with shape constraints being applied after each iterative deformation.

Qualitative tracking experiments were performed using both the manually- and automatically-trained hand models described above. In both cases, the HPDM was compared with a linear PDM built from the same training data. A real-time tracker (as described in Section 3.5) was exercised under each of the models; various gestures were performed in a controlled environment and the models' performance was observed. The following observations were made:

Tracking using the manually trained HPDM proved to be very promising. In comparison to the linear PDM, the HPDM experienced fewer distractions and tracking was generally more robust (see Figure 4.25).

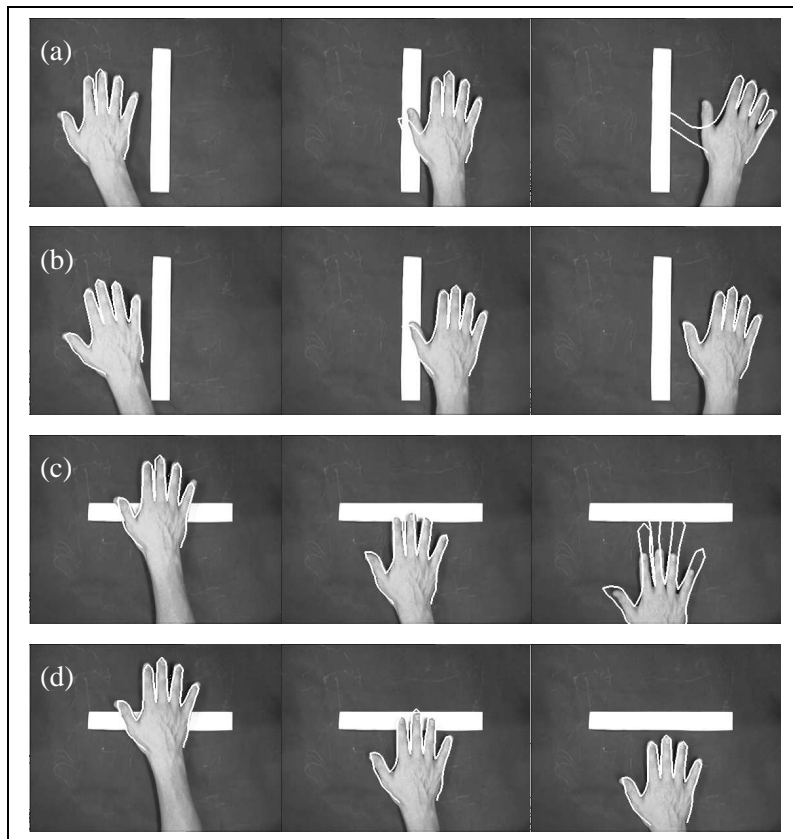


Figure 4.25: Tracking examples using manually-trained hand models; (a), (c) the linear PDM and (b), (d) the HPDM.

Tracking using the automatically-trained hand HPDM was less successful. Some

deformations were tracked well, such as movement of the thumb and ‘wagging’ of the fingers (Figure 4.26(a)). Other deformations were tracked less well. For example, the transition from fingers outstretched to a pointing gesture (Figure 4.26(b)) requires the movement of landmarks *around* the boundary, because the overall length of the boundary decreases, causing the evenly spaced landmarks to draw together. The ASM tracker has difficulty coping with such shape changes; there are no image cues to encourage movement of landmarks along a boundary (this is a manifestation of the aperture problem [36]). Another transition which was poorly tracked is that from fingers outstretched to a flat palm (Figure 4.26(c)). In this case there is a discontinuous change in the boundary shape as the fingers close together. The ASM tracker, which uses a local optimisation paradigm, cannot cope with such changes.

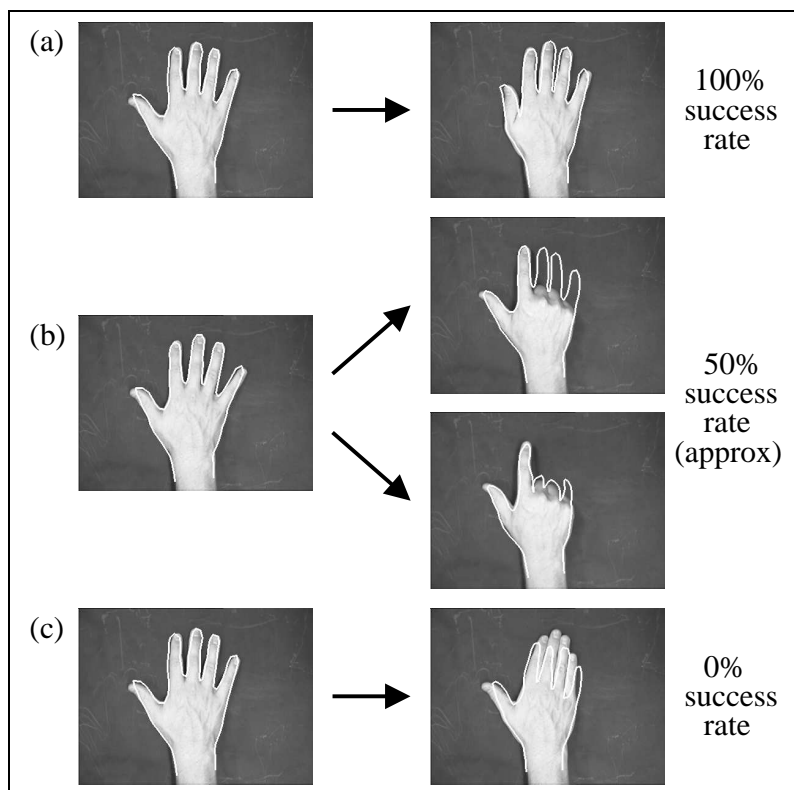


Figure 4.26: Tracking examples using the automatically-trained HPDM of the hand.

These problems might be alleviated by improved training data collection. The movement of landmarks around the boundary is due to the fact that the landmarks are *evenly spaced*, be the boundary long or short, as opposed to fixed on particular features. A slightly smarter tool for collecting training data might be able to spot features, such as areas of high curvature, and produce training data with a better correspondence between landmarks and object features. Hill *et al* [35, 37] have

suggested two possible algorithms for this. However this approach appears to be fairly object-specific; it is uncertain whether it would work in the general case. The sudden boundary shape changes are harder to cope with. In this example, they occurred when the fingers closed together. The simple boundary-finding algorithm used for training has no knowledge of expected hand shape, and consequently locates only the tops of the fingers of a closed palm, as opposed to dipping into the finger webs; hence a very different boundary shape is produced than for a hand with fingers outstretched. A possible solution is to use the bootstrapping algorithm described by Cootes and Taylor [16] to collect training data; new training examples are located in images using a model built from previous examples, but with extra variation included via a physical (FEM) model. In this way some shape knowledge is incorporated into the training. However, this process almost certainly requires manual guidance in all but the simplest of cases, and will also not work for objects which can take on a number of distinct shapes (this often occurs due to a change in viewpoint).

To put this performance in context, tracking using a linear PDM built from the same training data is very poor indeed. The model is underconstrained to such an extent that tracking is rarely successful (see Figure 4.27).

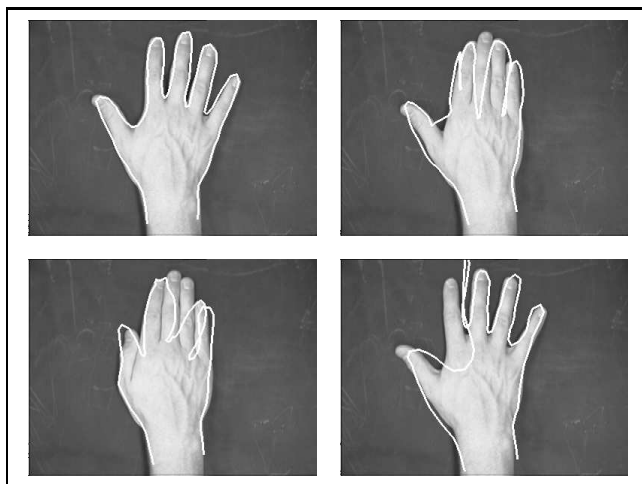


Figure 4.27: Tracking examples using the automatically-trained linear PDM of the hand.

## 4.6 Conclusions

The construction of Hierarchical PDMs has been described; use was made of a piecewise-linear PCA strategy. Qualitative and quantitative analyses on synthetic data have been undertaken, and performance on automatically-collected real data

has been examined with promising results. The HPDM is a great improvement over both the linear PDM and constraint surface models, and is a viable solution to the problem of constructing deformable models fully automatically.

The piecewise-linear approach requires a large amount of training data to build good models. However, this problem is negated by the fact that training data can be collected automatically. In the example given of the hand, it took less than 5 minutes to collect all the training data and build the model. More intelligent training data collection (e.g. Hill and Taylor's approach [37]) might give rise to a less complex global PCA space which could then be modelled with fewer linear pieces.

Another issue is that of speed. When applying the shape constraints it is necessary to calculate distances to every linear patch. This process is order  $n$  in the number of patches. Bregler and Omohundro suggest the use of 'Bumptrees' [55] (a tree-like data structure for representing functions and constraints) to decrease the number of calculations. A related approach would be to extend the hierarchical model to more than two levels, inserting intermediate-sized PCA spaces between the coarsest (global) and finest levels to give a multi-level tree structure. Search for the nearest patch(es) would descend through the tree giving, at worst, order  $n \log n$  performance and maybe better in the case of only a partial tree descent.

There are very strong parallels with this work in the statistics literature. The VSR can alternatively be thought of as a probability density function, and approximated as a Gaussian mixture. Instead of using  $k$ -means to determine the Gaussians, an *Expectation-Maximisation* algorithm [26] can be used with equal, if not better, success. This approach has very recently been taken by Cootes and Taylor [18]; their results are similar to those presented here. However, because their end application is static object location, little consideration is given to issues of speed, and the importance of a hierarchical approach is not emphasised.

To summarise, HPDMs are a definite improvement over linear PDMs. For manually collected training data, the improved models produced can be applied directly to ASM tracking with good effect. For automatically collected training data, the models produced are a vast improvement, but ASM tracking performance is less than satisfactory. In the next chapter an alternative approach to tracking is described which, when combined with HPDMs, provides an ideal solution to the problem of building deformable shape models automatically for tracking.

# Chapter 5

## Learning Models of Shape Dynamics

---

### 5.1 Introduction

Existing tracking algorithms which use deformable shape models generally rely on the assumption that objects move and deform smoothly over time. Object features and edges are detected in an image via a local search from the object's previous position. There are cases where this continuous behaviour is not adhered to, the main example being when tracking the *silhouette* of a 3D object.

Object silhouettes change shape smoothly for most of the time, but in certain situations there can be a discontinuous shape change. For example, in the case of human hands, this occurs when the fingers close together and the gaps between them disappear or when the hand turns sideways (see Figure 5.1). Similar effects are seen on the arms and legs of a walking person. Sometimes these sudden changes are a side effect of temporal sampling (as a result of using, for example, a 25Hz camera) but there are also cases where there is an intrinsic temporal discontinuity (e.g. the touching together of the thumb and forefinger).

Local optimisation-based approaches generally cannot track objects through such discontinuities; they usually become trapped in a local maximum and sometimes fail more catastrophically.

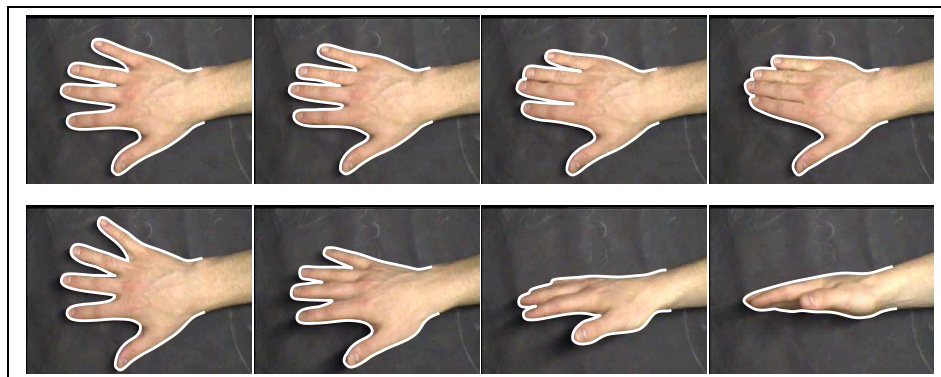


Figure 5.1: Discontinuous changes in object boundary shape due to deformation (top row) and rotation (bottom row).

In the case of models built from automatically-collected training data (by way of the Hierarchical PDM), as described in Chapter 4, the non-intelligent placement of landmarks means that shape discontinuities are fairly common. In addition, even during smooth shape changes, landmarks can in some cases ‘slide’ around the object boundary; existing trackers do not perform well in such situations (as discussed in Section 4.5).

A recent development, the CONDENSATION algorithm (Section 2.2.4), improves matters by providing a stochastic framework for tracking. Performance on the types of deformations described above is superior to previous trackers, mainly due to a departure from the deterministic hill-climbing approach; CONDENSATION is effectively able to traverse hills and also jump over (small) valleys. However, there is still an underlying assumption of smooth dynamics, and abrupt changes are not tracked well.

In this chapter a novel method for modelling shape dynamics is proposed, which encompasses both continuous and discontinuous shape changes in a non-deterministic framework. Continuous shape deformations correspond to continuous movements through the model shape space, whereas discontinuous shape changes require ‘jumps’ through shape space. Use has been made of the Hierarchical PDM, wherein allowable shapes are represented as a union of (learned) bounded regions within such a space. Shape changes are described in terms of movement within and between these regions; discontinuous shape changes involve transitions between non-adjacent regions. Transition probabilities are learned from training sequences and stored as a Markov model. In this way ‘wormholes’ in shape space are created.

Tracking with such models is via an adaptation of the CONDENSATION algorithm. CONDENSATION tracks by propagating a probability distribution in model state



space over time. The propagation algorithm is substituted with an adapted Markov process, driven by the new model of shape transition.

The remainder of the chapter is set out as follows. Firstly the method for modelling shape transitions is described. Secondly it is shown how to adapt the CONDENSATION algorithm, based on the new model, in order to cope with the tracking of discontinuous shape changes. Following this the new technique is evaluated in comparison to previous approaches. Finally an extension to the CONDENSATION algorithm is described which gives an improvement in performance, and some conclusions are drawn.

## 5.2 Modelling Discontinuous Changes in Shape

The discontinuous shape changes described above are entirely predictable: they occur repeatedly between certain pairs of shapes. As such, they can be learned from training sequences containing examples of characteristic object movement.

The Hierarchical PDM (HPDM), as described in Chapter 4, has been used. The HPDM produces a model of shape which is represented by a set of local patches that cover all the valid regions in shape space. The key to our approach is the observation that these patches are generally small enough to cover only minor variations in object shape, and can thus be looked on as being discrete states for the object shape. This situation allows for the use of a Markovian representation of object shape dynamics, with each state representing a different shape and the state transition probabilities reflecting typical shape changes.

To this end, a Markov state transition matrix  $\mathbf{T}$  can be constructed as follows:  $E$  pairs of consecutive shapes,  $\mathbf{x}_e$  and  $\mathbf{y}_e$ , are collected from one or more continuous training sequences of characteristic object movement<sup>1</sup>. A preliminary matrix  $\mathbf{T}'$  is first calculated using:

$$\mathbf{T}' = \sum_{e=1}^E \mathbf{p}(\mathbf{x}_e)\mathbf{p}(\mathbf{y}_e)^T \quad (5.1)$$

where  $\mathbf{p}(\mathbf{x})$  is a vector of probabilities  $(p(\mathbf{x}, 1) \dots p(\mathbf{x}, k))^T$ , with  $p(\mathbf{x}, c)$  being the probability that shape  $\mathbf{x}$  is a member of patch  $c$ . The simple rule that  $p(\mathbf{x}, c) = 1$  if  $c$  is the nearest patch and  $p(\mathbf{x}, c) = 0$  otherwise has been used, but a more complex function could be derived based on relative distances to patches.

---

<sup>1</sup>From a training sequence of  $N$  frames of object movement it is possible to generate  $N - 1$  such pairs: 1 and 2, 2 and 3, ...,  $N - 2$  and  $N - 1$ ,  $N - 1$  and  $N$ .

The preliminary matrix  $\mathbf{T}'$  is then normalised with respect to each row so that  $[\mathbf{T}]_{a,b}$  gives the probability of a transition from patch  $a$  to patch  $b$ :

$$[\mathbf{T}]_{a,b} = [\mathbf{T}']_{a,b} / \sum_i [\mathbf{T}']_{a,i} \quad (5.2)$$

The matrix  $\mathbf{T}$  provides a probabilistic model of object shape transitions.  $\mathbf{T}$  would be expected to have large values along its diagonal, indicating that for the majority of the time a shape remains in the same patch. Large off-diagonal values represent learned shape transitions, many of which will be to adjacent patches. However, if a specific discontinuous change appears repeatedly in the training data, this too will give rise to a high transition probability. Figure 5.2 illustrates a typical transition matrix.

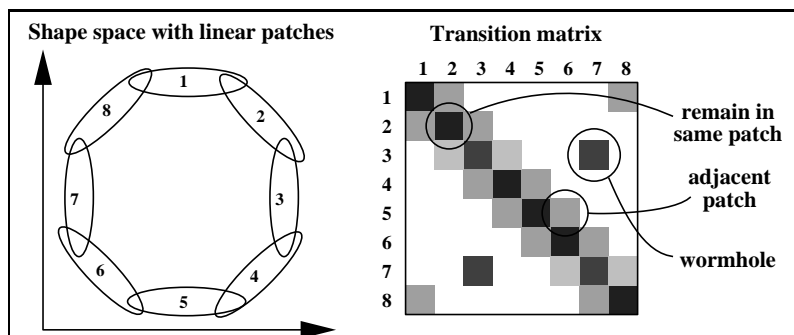


Figure 5.2: A typical transition matrix. The transition probabilities are represented by shaded blocks; a darker shade corresponds to a higher probability.

For practical use, it is convenient to construct a *cumulative* version of the transition matrix; this makes for more efficient probability sampling:

$$\mathbf{C}_{a,b} = \sum_{i=1}^b [\mathbf{T}]_{a,i} \quad (5.3)$$

This model of shape transition can be used for generation or prediction of object deformation over time; the latter case is now illustrated by applying it to model-based object tracking.

### 5.3 Tracking

The model of shape transition described above is based on probabilities and, as such, lends itself to a non-deterministic approach to tracking. For this reason, use was made of Isard and Blake's CONDENSATION algorithm.

A more detailed description of CONDENSATION (Stochastic **Conditional Density Propagation**) is given in Section 2.2.4, but the essence of the approach is the representation of an object's location not by a single point in the model parameter space, but by a probability density function (pdf) over the space. A model of conditional probability (learned from training sequences) is used to propagate the pdf over time. In other words, given the pdf at time  $t$  there is a mechanism to predict the pdf at time  $t + 1$ , based on a simple model of object motion. The new pdf is consolidated and refined by referring to the current image. Specifically, a *fitness* function is used to determine the goodness-of-fit of model to image at any position in the parameter space.

In practice, the pdf is represented by a population of samples drawn from the parameter space. Each one has its fitness calculated and *factored sampling* [30] is used to choose seeds for propagation.

The benefits of CONDENSATION are as follows:

- It can support multiple hypotheses; this is represented by a pdf with multiple peaks.
- It provides an improvement over deterministic hill-climbing trackers in that it can effectively traverse hills (in this sense it is a non-greedy algorithm).
- It recovers well from failure; the stochastic nature of the algorithm allows it to escape from local maxima.
- It incorporates a level of prediction (learned from training data) which improves the speed of convergence and the quality of results over, for example, a Genetic Algorithm.

The prediction aspect is embedded in the propagation equations. Isard and Blake use a 'Fokker-Planck' style stochastic differential equation (see Section 2.2.4) which consists of two elements: a deterministic term which allows for simple drifting of the pdf, and a stochastic term which encourages spreading of the pdf. Although the tracker *can* escape from local maxima (due to the stochastic term), the underlying dynamical model is still based on an assumption of smooth, continuous object movement.

The tracking of sudden shape changes is possible via a modification to the *propagation* step of CONDENSATION. The specification for propagation is quite general; the abstract notion is a conditional pdf  $p(\text{shape at time } t+1 | \text{shape at time } t)$  which

propagates the shape pdf over time. The algorithmic requirement is a function which, given a vector  $\mathbf{s}$ , being the shape at time  $t$ , returns a new vector  $\mathbf{s}'$ , being a *possible* shape at time  $t + 1$ , sampled from the conditional pdf.

Our learned model of object shape transitions can be used to provide the control parameters for a Markov process-based propagation algorithm, using the cumulative transition matrix defined in (5.3). This proceeds as follows, noting that  $\mathbf{s}$  and  $\mathbf{s}'$  are vectors representing shapes within the HPDM global PCA space:

1. Determine the HPDM patch membership of the source shape  $\mathbf{s}$ . Currently it is assumed that  $\mathbf{s}$  is member of the nearest patch (but, as for the transition matrix learning algorithm, a more complex function could be derived based on relative distances to patches). Label this patch  $a$ .
2. Use row  $a$  of the cumulative transition matrix  $\mathbf{C}$  to select probabilistically the destination patch  $b$ . To do this, generate a random number  $z$  from a uniform distribution over the range  $[0, 1]$  and choose the smallest  $b$  such that  $\mathbf{C}_{a,b} > z$ .
3. Set  $\mathbf{s}'$  (the destination shape) at a position within patch  $b$ . If  $b = a$  then  $\mathbf{s}'$  is set to  $\mathbf{s}$  plus a random perturbation. If  $b \neq a$  then  $\mathbf{s}'$  is set to the centre of cluster  $b$  plus a random perturbation. In either case, the random perturbation is normally distributed, and scaled and orientated with respect to the principal axes of linear patch  $b$ . Put more formally:

$$\mathbf{s}' = \begin{cases} \mathbf{s} + \mathbf{P}_b \mathbf{\Lambda}_b \mathbf{\Omega} & : a = b \\ \bar{\mathbf{b}}_b + \mathbf{P}_b \mathbf{\Lambda}_b \mathbf{\Omega} & : a \neq b \end{cases} \quad (5.4)$$

where  $\bar{\mathbf{b}}_b$ ,  $\mathbf{P}_b$  and  $\mathbf{\Lambda}_b = \text{diag}(\sqrt{\lambda_{b1}} \dots \sqrt{\lambda_{bt_b}})$  are the patch centroid, orientation axes and variances respectively, as defined in Section 4.3.1, and  $\mathbf{\Omega}$  is a vector of  $t_b$  independent standard normal random variables.

This final stage makes the propagation model more than just a discrete-state process; by including a ‘spread’ function the model becomes continuous (in fact it becomes very similar to a *Hidden Markov Model*).

The result of using the above algorithm is that a small number of samples from the shape population ‘jump’ through wormholes in shape space at every iteration. In most cases these jumps will result in low fitness candidates which are unlikely to survive into the following iteration. However, if a sudden shape change *has* occurred then a high fitness candidate will be produced and other population members will

quickly migrate to the new fitness peak. This has the desired effect of tracking such discontinuous shape changes.

## 5.4 Evaluation

In order to demonstrate the new tracker, a comparative evaluation was performed with two other trackers: a simple CONDENSATION tracker and the snake-like Active Shape Model (ASM) tracker used previously.

The underlying shape model used for all three trackers was the same: a HPDM of the human hand, similar to that illustrated in Figure 4.22, but with a gesture set chosen specifically to produce discontinuities in the shape space. Training data was collected by recording a sequence of gestures performed against a homogeneous background, and applying a simple boundary-finding algorithm to each frame in the sequence. Each training example consisted of 100 evenly spaced landmarks around the hand silhouette boundary.

The HPDM was then constructed as described in Chapter 4. Figure 5.3 shows the training data (as projected into the two principal dimensions of global PCA space) and the 20 linear patches which were constructed from it.

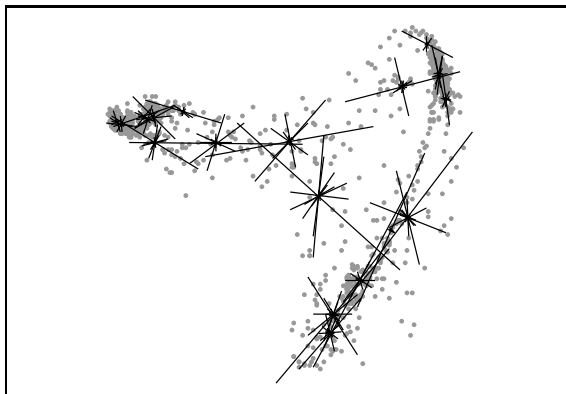


Figure 5.3: The automatically captured training examples (projected into the two principal dimensions of global PCA space) and the HPDM linear patch principal axes.

The same training data could also be used to build the Markov transition matrix which models the conditional pdf; this was possible because the training examples were collected as a continuous sequence. Figure 5.4 shows some example conditional pdfs for both Fokker-Planck (top row) and Markov model (bottom row) propagation algorithms, generated by choosing a single ‘seed’ position in global PCA space, calculating a large number of destination positions and compiling a 2D histogram.

The Fokker-Planck algorithm used was in fact ‘untrained’: a zero-velocity assumption was made and a simple Gaussian spread used. A trained tracker would in theory perform better, but would still be incapable of modelling non-continuous dynamics.

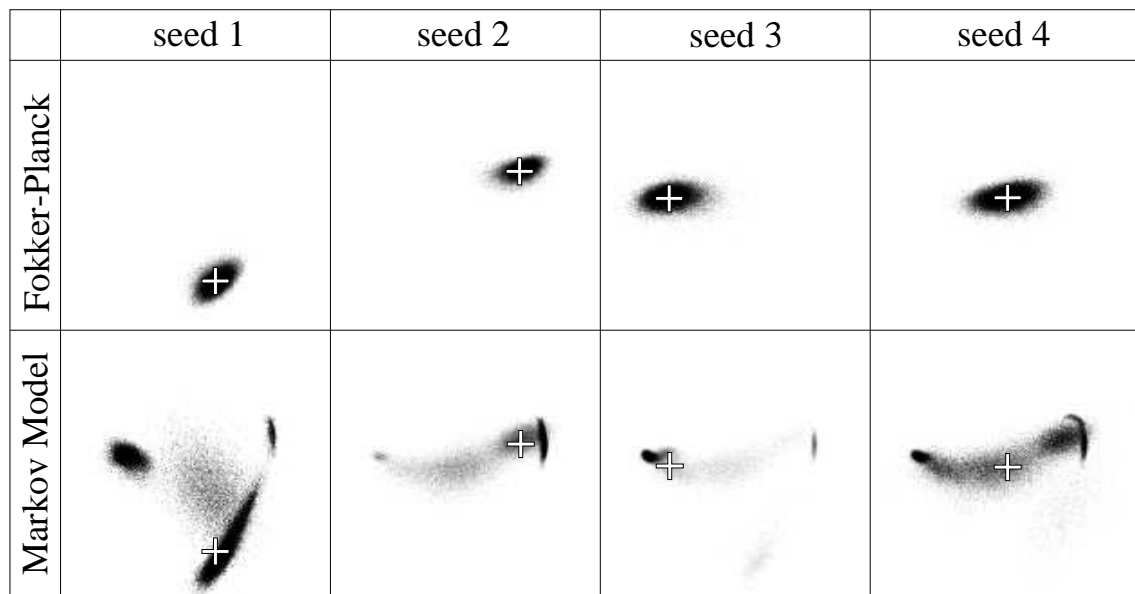


Figure 5.4: Some example conditional pdfs generated from single seeds via both the Fokker-Planck and Markov model algorithms. The crosses indicate the seed positions.

The Fokker-Planck algorithm essentially produces a Gaussian distribution around the seed, truncated by the HPDM constraints. The Markov model algorithm uses the transition matrix to generate more general conditional pdfs. This model has captured valid jumps through shape space, indicated by the multiple dark areas. Note that the Fokker-Planck algorithm in this example is untrained; training would alter the shape of the pdf, but only in terms of orientation and eccentricity of the Gaussian. More complex pdfs are not possible under this model.

To compare the trackers’ performance, an unseen sequence of hand movement was filmed. The sequence featured many gestures, including the types of sudden shape changes which have caused problems for our previous trackers. Figure 5.5 shows a graph of the fitness score<sup>2,3</sup> for each tracker over the video sequence. Note that a hybrid version of the CONDENSATION algorithm was used in this experiment;

<sup>2</sup>The fitness function used examines image pixels along a line normal to the contour at each control point and returns a score proportional to the number, proximity and strength of edges found. More details can be found in [32].

<sup>3</sup>For the CONDENSATION-based algorithms the *modal* fitness value was used.

this is detailed in Section 5.5. The failure threshold gives the approximate fitness value (manually determined) above which a good fit is deemed to have occurred.

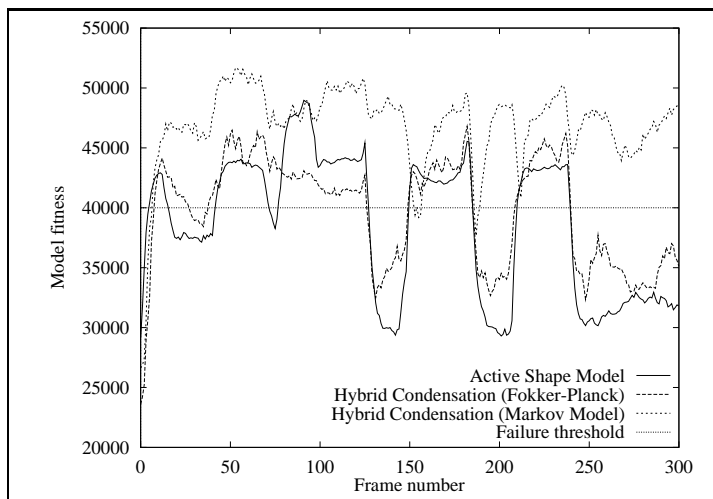


Figure 5.5: Graph showing the fitness scores for the different tracking algorithms on an image sequence containing sudden shape changes.

The positions of the sudden shape changes in the sequence (approximately every 30 frames) can easily be identified as the fitness for all three trackers changes suddenly. The ASM tracker is unable to respond at all to the shape changes; there are no image edges to follow in order to deform the model shape in the right way. This is indicated by the large fluctuations in the fitness score. The Fokker-Planck tracker has slightly better performance: the same large fitness fluctuations are seen, but in some cases the tracker starts to recover (e.g. frames 125 to 150) until another shape change occurs. The Markov model tracker, however, clearly out-performs the other two in this example. Its fitness level does initially drop after a shape change, but it quickly manages to return to full strength, indicating that the underlying population has managed to ‘migrate’ to the correct shape.

Figure 5.6 provides a more detailed analysis of how the two different stochastic propagation algorithms handle a sudden shape change. A section of the above experiment was isolated and snapshots were taken every five frames showing the input image and the positions in shape space (projected into 2D) of the populations for both the Fokker-Planck and Markov model propagation algorithms.

Initially, the populations for both trackers are clustered around the correct position in shape space (towards the right hand side). Note that the Markov model population is more focussed, but with a small number of members in distant areas; these areas are potential destinations of a sudden shape space jump. As soon as the

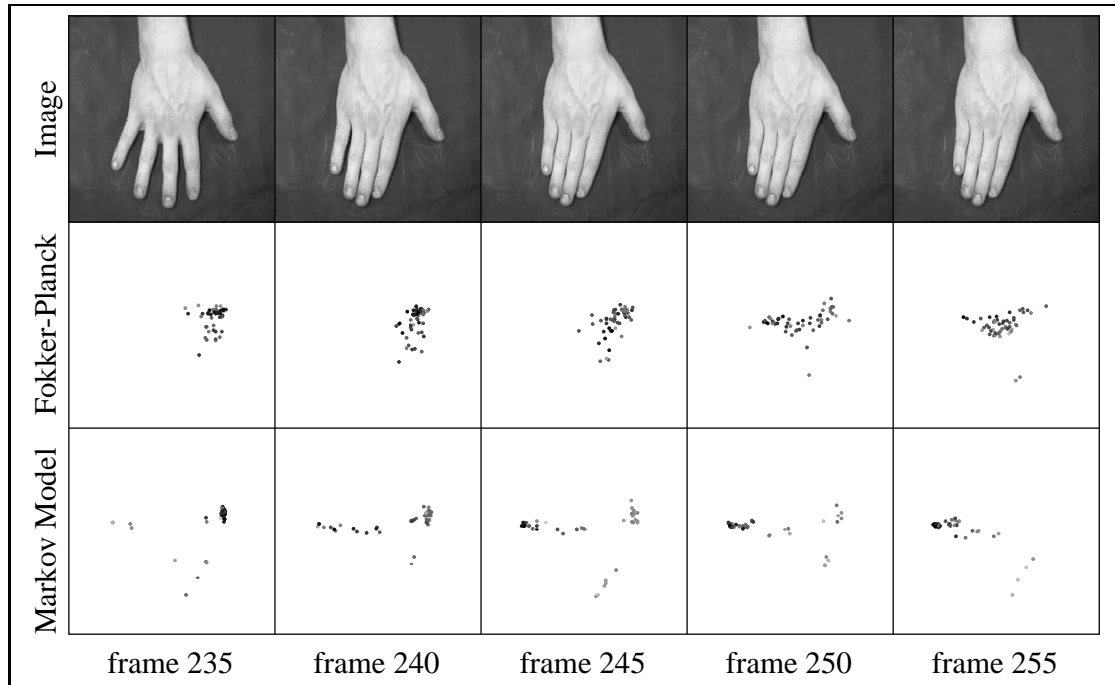


Figure 5.6: Coping with a sudden change: the image sequence, and the population distributions for the (untrained) Fokker-Planck and Markov model tracking algorithms, as projected into the first two dimensions of global PCA space.

shape change occurs, the Markov model population begins to migrate to the left, and by frame 245 a new cluster has formed on the left hand side. The Fokker-Planck population is much slower to move, and even after 20 frames it is only approximately half way there.

## 5.5 Improving CONDENSATION

A drawback of the CONDENSATION algorithm over a deterministic local optimisation (i.e. snake-like) tracker is that of speed. This is because in every frame there is a population of up to several hundred candidate shapes to process, whereas deterministic trackers process only one shape.

The construction of a hybrid tracker which makes use of both stochastic and deterministic tracking has been investigated. This produces a system which is robust but also faster and more accurate than a purely stochastic tracker.

The hybrid tracker is based on a CONDENSATION tracker as described above, but after each new candidate shape is produced via the propagation function, one or more iterations of local optimisation are performed (using an ASM) in order to refine the shape and hence improve its fitness. The number of iterations applied to



each shape is proportional to its fitness so that more CPU cycles are allocated to promising shapes and fewer to poor shapes. A fixed number of iterations are shared out in order to keep the tracker running at a steady rate. The result is that the peaks in the pdf are better represented, and consequently a much smaller population is required for accurate tracking. A similar approach has been used to good effect by Hill *et al* in combining Genetic Algorithms with ASMs [33].

Figure 5.7 illustrates the propagation of a population of shapes under the hybrid algorithm. The local optimisation step results in better clustering of samples around the fitness peaks and higher overall fitness levels.

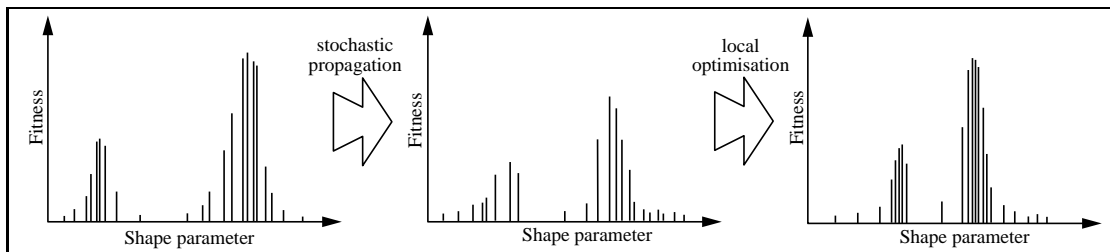


Figure 5.7: Propagation of a 1D shape population using the hybrid tracker.

Figure 5.8 shows a comparison of three different tracking algorithms—purely stochastic (CONDENSATION), purely deterministic (ASM) and hybrid—on a typical sequence of hand movement. The graph on the left shows comparative performance on moderate speed hand movement and the graph on the right is for the same sequence at high speed (only every eighth frame is used). The failure threshold shown is a (manually determined) level below which a tracking failure is deemed to have occurred. The CONDENSATION tracker had a population of 150 shapes, this number being chosen to give a reasonable level of performance. The hybrid tracker was allowed only 50 shapes and 50 cycles of local optimisation were shared out per iteration. In total it was approximately 1.5 times as fast as the CONDENSATION tracker. The ASM tracker was appended with a (very slow) failure recovery mechanism which was activated if the fitness dropped below the failure threshold. This was for the purpose of illustration only.

The hybrid algorithm clearly out-performs the purely stochastic tracker, despite having a smaller population. At moderate speeds the ASM and hybrid trackers' performance is comparable. At high speeds the ASM tracker is not robust, requiring re-initialisation every few frames, whereas the hybrid tracker experiences only one perceived failure (in frame 78), and it recovers from this without any help. Note that the ASM only performs better than pure CONDENSATION because the hand

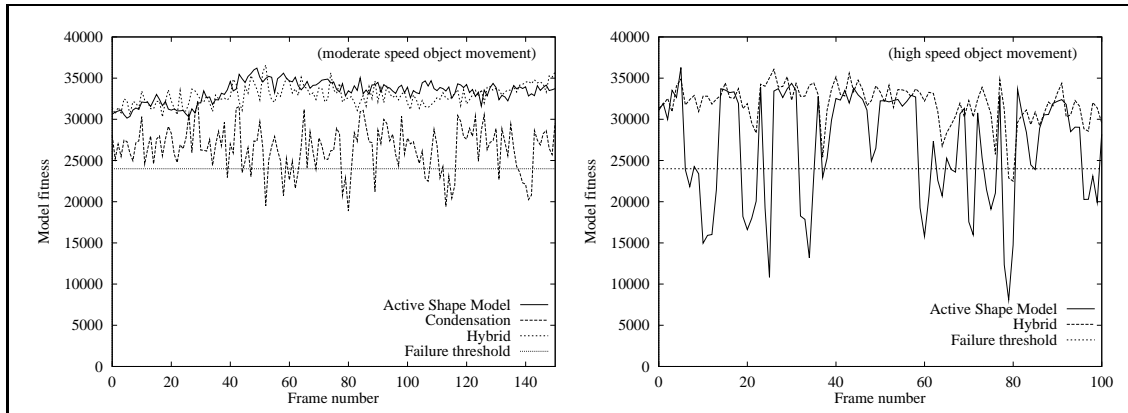


Figure 5.8: Graphs comparing the hybrid tracker to a CONDENSATION tracker and an Active Shape Model tracker, on moderate (left) and high (right) speed hand movements. The failure threshold was chosen manually.

movement is relatively slow and the scene is uncluttered.

In summary, the hybrid tracker is faster and more accurate than the purely stochastic tracker, and more robust but slower than the deterministic ASM tracker. It is worth noting that the hybrid approach may not be suitable for all applications because in some cases it might counteract the stochastic process which makes the tracker robust to tracking ambiguities such as clutter.

There is some commonality between the ASM and CONDENSATION algorithms—they both require the same image measurements to be made—so a substantial speed increase can be gained by coding the combined algorithm optimally.

## 5.6 Conclusions

This chapter has described the modelling of shape dynamics using a union-of-regions representation of shape along with a learned Markov model containing transition probabilities between the regions. Transitions between non-adjacent regions correspond to ‘wormholes’ in the shape space. Tracking of temporal discontinuities has been enabled using this model in combination with an adaptation of the CONDENSATION algorithm. It has been shown how the new method can give improved performance over previous approaches.

These results are interesting for two reasons:

- The tracking of 3D objects using 2D shape models on their silhouettes is made possible.

- Tracking using models which have been trained *fully automatically* is more successful. Previously, manual annotation of training examples was highly desirable in order to ensure that the models produced were sufficiently *specific* and *continuous* for tracking purposes.

Although the described experiments were performed in a controlled, uncluttered environment, Isard and Blake [43] have demonstrated that CONDENSATION is effective in a cluttered environment and so it is anticipated that the adapted version will also cope with such situations.

A substantial amount of training data is required to build the underlying HPDM shape model and the transition matrix. However, training data capture is automatic and thus a large number of training shapes can be collected in a short time. In the case of the hand models, the training phase involved performing several gestures repeatedly under a rostrum camera; this took around five minutes.

There are some useful extensions that could be made to the dynamic model as described:

- In the construction of the transition matrix the patch membership of each training example is currently decided in a nearest-neighbour fashion. A better alternative might be to provide a probabilistic membership function; this would be especially useful for shapes which lie halfway between two patches.
- The stochastic propagation model could be further refined by combining the Markov model with learned Fokker-Planck ‘drift and diffuse’ dynamics. It would be possible to learn a different dynamic behaviour for each linear patch, or even for each transition pair or patches.

Finally, it is worth noting the similarity of our model to a Hidden Markov Model (HMM) [61]. A HMM differs from a standard Markov Model in that each state is associated not with a single output value, but with a distribution over either a discrete alphabet or a continuous space. In our case the distribution for each state is a Gaussian within the (continuous) global PCA space, and the states themselves are in effect ‘hidden’; only the PCA space is visible. With this observation in mind, it is likely that techniques used in the HMM literature will also be applicable to our model; this is certainly worthy of investigation.

# Chapter 6

## Tracking with 3D Deformable Shape Models

---

### 6.1 Introduction

So far the emphasis of this work has been on a 2D approach to shape modelling; the models built have essentially been *appearance* models, based on the projection of objects into a 2D plane. The rationale for this has been their suitability to the particular sensor technology being used (i.e. standard video cameras producing 2D images), as well as the associated benefits of efficiency and simplicity.

One of the disadvantages of having a 2D abstraction is that it is sometimes difficult to infer the 3D pose of an object given only 2D information such as its silhouette boundary. In particular, the modes of variation of a silhouette model only indirectly represent the physical deformations of a 3D object. An obvious alternative is to attempt to build 3D models of objects and use them directly for tracking. Pose inference then becomes trivial, and the need for complex models of discontinuous shape change is alleviated. The questions that arise instead are how can such models be constructed, and how is it possible to track with them from only 2D images.

In this chapter both these issues are addressed. It is shown how it is possible to build 3D models as surface meshes, with deformation being achieved via a 3D Point

Distribution Model. It is then described how the Active Shape Model approach to tracking can be modified to enable the fitting of 3D models to 2D image sequences. Finally some qualitative results are presented and discussed.

## 6.2 3D Model Construction

In previous work, deformable 3D models used in tracking systems have exclusively been hand-crafted, making use of primitives such as cylinders connected by hard pivotal constraints. Building such models is generally laborious, and attaining accuracy is also very difficult.

Experiences with 2D modelling suggest that statistical shape models may provide a viable alternative. The use of real-life training examples makes for accurate models and deformation characteristics are learned automatically; no hard-coding is required.

Point Distribution Models (PDMs) extend naturally into 3D. A PDM is represented in terms of the Cartesian coordinates of a number of landmarks on an object. The extension to 3D involves the inclusion of the  $z$  coordinate as well as the  $x$  and  $y$  coordinates in the model.

When working in 2D, the shape of an object has been modelled in terms of its boundary because this feature is easy to locate in training images, is good for visualisation, and is well-suited to tracking by way of simple edge detection methods. An analogous approach is taken in 3D: an object is modelled in terms of a mesh over the whole of its surface for exactly the same reasons.

The main difficulty in the construction of such models is the collection of training data. There are two aspects to this problem: the acquisition of 3D image data for a number of example object shapes and the extraction of a suitable surface mesh from each of these images.

### 6.2.1 Training Image Acquisition

The acquisition of good quality 3D image data is certainly not trivial. A variety of sensor technologies exist, but each has its drawbacks.

It is possible to generate 3D data by way of 2D images using stereo or multiple views. Shape can be deduced quite accurately from multiple views of an object's silhouette [67]; careful calibration and/or constrained circumstances are necessary for this. Uncalibrated systems have also been demonstrated [31, 7]; these generally use

feature correspondence and so require an abundance of distinctive object features, and in any case often produce less than accurate results.

One level up in technology are laser range finders, which produce a *depth* image of the target object. Careful calibration is not required and results are more accurate than from multiple 2D images. However, scans can take up to several minutes and also, a full 3D image is not obtained – only one side of an object is captured. For a full 3D image it is necessary to scan an object from two or more directions and attempt to ‘stitch’ the two scans together [25]. Another disadvantage is that at present laser technology is not widely affordable. A cheaper alternative is to use *structured lighting*. The theory is similar, but the hardware is much less expensive; however, the resolution of images produced is lower.

The best results are generated from medical 3D imaging apparatus. Magnetic resonance (MR), computer tomography (CT) and 3D ultrasound can all produce accurate gray-scale voxel (3D *volume pixel*) maps of various types of object. Such machines are very expensive and are generally only found in large hospitals, and so availability is scarce.

Strong links with St. James’ Hospital in Leeds have meant that it *has* been possible as part of this work to obtain a small number (8) of MR scans of human hands in various poses. Figure 6.1 shows some slices (parallel to the  $z$  axis) from one of these images.



Figure 6.1: Slices from a Magnetic Resonance scan of a human hand.

Each volume image consists of  $256 \times 256 \times 20$  voxels, with 1mm resolution in the  $x$  and  $y$  axes and 2.4mm resolution in the  $z$  axis. The images are 256 level grey-scale; the sensor responds to water in objects and hence skin, tissues and bone marrow appear quite light, whereas bones are much darker.

The images were post-processed to make them more suitable for training mesh capture. Firstly, they were re-sampled along the  $z$  axis (with each new voxel being interpolated from the original image) to give a  $256 \times 256 \times 48$  image with 1mm

cubic voxels. Following this they were passed through a 3D median filter to reduce the level of image noise, and finally thresholded to give a binary image in which the hand was completely white and the background was completely black. Removing the internal hand features in this way avoided distractions in the mesh-fitting process.

### 6.2.2 Training Mesh Capture

The collection of training data for a PDM essentially involves finding the coordinates of perhaps several hundred *landmarks* for each of the training images of the object being modelled. For 2D models, this process is often performed by hand, with the aid of some visualisation tool. It is time-consuming and laborious, and inevitably leads to inaccuracy and error. Gathering landmark data manually for 3D models is near-impossible due both to the problem of image visualisation and the sheer quantity of data involved.

There are, of course, many established methods for capturing the positions of image features automatically. Lorensen and Cline [51] describe a “Marching Cubes” algorithm which triangulates a surface from 3D voxel data. More recently, work on physically-based deformable meshes [25, 14, 11] has provided more robust methods of capturing surface information.

However, for the purposes of building a PDM there should ideally be a direct correspondence between similar landmarks across the whole training set (i.e. a particular landmark should mark the *same* feature on each training example), so applying any of the above techniques independently to each training example is unlikely to be of much use.

Attempts have been made to address this problem. Hill and Taylor [35] and Baumberg and Hogg [3] both describe methods for 2D models which work in the case where it is possible to obtain a single clean pixelated boundary from each training image. Hill and Taylor apply a pairwise corresponder in a hierarchical fashion to find approximate matches between training boundaries, whereas Baumberg and Hogg constrain the problem by assuming constant object orientation. Both employ some form of iterative optimisation to improve the models produced. It may be possible to extend these ideas into 3D: in other work, Hill and Taylor [38] show how to capture 3D data by way of 2D slice contours.

In this work, a semi-automated process, based on 3D physically-based modelling techniques, is used. There are two stages to the process. Firstly a surface mesh model of the object is constructed; this can be done manually, or automatically

with the aid of one of the training examples. Following this, the mesh is deformed to fit each training example in turn: a few key features are located manually and various forces are applied to drive the mesh into position. Internal forces keep the mesh smooth and even, and image forces help to give an accurate fit.

This idea is not entirely new: Cootes and Taylor [16] describe how to combine physical and statistical shape models in such a way as to rely initially on physical modelling but to place emphasis more on statistical modelling as the number of training examples increases. They demonstrate how such a system can be used to ‘bootstrap’ a PDM. Syn and Prager [71] develop this idea into a more robust and practical tool by allowing *guided* model fitting, whereby key features are located by hand. The algorithm described here also makes use of guided fitting, but a slightly different approach to modelling is adopted by drawing a separation between the physical and statistical domains. This allows the use of *any* physical modelling technique, not just the Finite Element Method used in the aforementioned works.

### 6.2.3 Physically-Based Models

Physically-based models come in a variety of different forms. They all have in common the ability to deform under the action of various forces. When applied to feature location/tracking, a physical model is usually considered as a system of  $N$  point masses whose motion over time is governed by standard Newtonian dynamics. Two types of force are generally applied:

- *External Forces*: the point masses are ‘attracted’ towards particular image features in order to fit the model to the image data. These forces might be applied manually (in a *guided* system) or via some sort of feature detection (e.g. edge detection).
- *Internal Forces*: the point masses interact with one another to hold the model in shape. These are usually elastic forces tending to drive the model towards a stable rest configuration.

By allowing these forces to act over time it is hoped that the model will deform to fit the image data. We can describe the dynamics of the system with the following Newtonian law of motion:

$$m_i \frac{d^2 P_i}{dt^2} = F_{int} + F_{ext} - \gamma \frac{dP_i}{dt} \quad (6.1)$$



where  $P_i$  is the instantaneous position of point  $i$ ,  $F_{int}$  and  $F_{ext}$  are the instantaneous internal and external forces on point  $i$ ,  $m_i$  is its mass and  $\gamma$  is a damping factor. If time is discretised in even steps and unit mass is assumed, integrating (6.1) with respect to time gives the following:

$$P_i^{t+1} = P_i^t + F_{int} + F_{ext} + (1 - \gamma) \cdot (P_i^t - P_i^{t-1}) \quad (6.2)$$

where  $P_i^t$  is the position of point  $i$  at discrete time interval  $t$ . This equation can be used to calculate the new position of the model, given its previous two positions. Deformation of such a physical model thus progresses iteratively.

### 6.2.4 Simplex Meshes

The physical model used here is a basic version of the Simplex Mesh as described by Delingette [25]. Simplex meshes are surface meshes which exist in a 3D space, consisting of a number of vertices (of unit ‘mass’), each connected to exactly three neighbouring vertices. It is possible to model any conceivable topology in this way. Delingette describes many properties of Simplex Meshes; the most useful is the concept of the simplex angle - this is measured as shown in Figure 6.2.

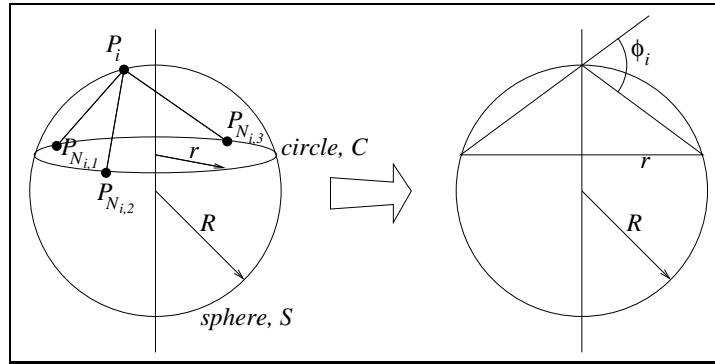


Figure 6.2: The Simplex Angle  $\phi_i$  at vertex  $P_i$  with neighbours  $P_{N_{i,1}}$ ,  $P_{N_{i,2}}$  and  $P_{N_{i,3}}$ .  $\phi$  is a function of only  $r$  and  $R$ .

The simplex angle  $\phi_i$  for a vertex  $P_i$  is a measure of the surface *curvature* in the locality of the vertex. It has the good properties that it is invariant to scale, to the positions of its neighbours  $P_{N_{i,1}}$ ,  $P_{N_{i,2}}$  and  $P_{N_{i,3}}$  on the circle  $C$ , and to the position of  $P_i$  on the sphere  $S$ . The simplex angle can be put to good use when generating the model’s internal forces. For vertex  $P_i$ , an elastic force is constructed which drives the point towards a position such that:

- The simplex angle subtended is some specific angle  $\Phi_i$ , and

- $P_i$  is equidistant from  $P_{N_{i,1}}$ ,  $P_{N_{i,2}}$  and  $P_{N_{i,3}}$ .

Full details are given in [25]. The choice of the  $\Phi_i$  alters the ‘stable’ shape of the mesh. For example, choosing  $\forall i. \Phi_i = 0$  encourages smoothness over the surface of the mesh. Using  $\forall i. \Phi_i = \phi_i^t$  (where  $\phi_i^t$  is the value of  $\phi_i$  at time  $t$ ) sets the stable shape to the shape at time  $t$ . Use is made of both of these settings at various stages of the model fitting process.

### 6.2.5 Initial Mesh Construction

As mentioned in Section 1.1.1, the first stage in building a model of an object is to decide which features of the object are to be modelled. It has already been decided to model the *surface* of objects, via a mesh, but the structure of the mesh must be determined: the number of vertices required, which part of the object each vertex represents, and how they connect together.

For models of a few hundred vertices or less, it is possible to define the structure by hand. However, this is very time-consuming, involving a great deal of pencil-and-paper work, and is also error-prone (although errors are usually easily identified).

Alternatively, any of the mesh-fitting algorithms described in [51], [25], [14] or [11] can be used to generate an initial structure for the model automatically by applying them to one of the training images. However, it is not guaranteed that particular object features will be landmarked and for more complex objects some manual intervention may be required (particularly for Simplex Meshes).

The automatic mesh-fitting algorithms also provide an initial *shape* for the model, whereas manual construction does not. However, this is not a large handicap because the physical deformation can take place (admittedly more slowly) with the vertices initialised at random coordinates, initially using manually-placed guiding forces (as explained below). This is only necessary for the first training example; the second and subsequent examples can be deformed from the first.

### 6.2.6 Mesh Deformation

Once a physical model has been constructed, it is deformed under the action of various forces in order to fit each training example, using (6.2). Internal forces are as described for the Simplex Mesh above, using various simplex angle constraints as detailed below. External forces come from two sources:

- *Guiding forces.* These are set up manually and are used to help the model find its approximate destination in the early stages of deformation. The coordinates of prominent object features are located in the training image by hand, and virtual ‘springs’ are attached between these positions and the corresponding model vertices. The stiffness of the springs can be altered to strengthen or weaken the forces. The force  $S_i$  on vertex  $i$  is given by:

$$S_i = k_i(D_i - P_i) \quad (6.3)$$

where  $P_i$  is the vertex’s current position,  $D_i$  is the ideal position and  $k_i$  is the spring stiffness coefficient. A value of  $k_i = 0.7$  provides relatively rapid convergence whilst avoiding oscillation.

- *Image forces.* These are forces exerted on vertices due to the 3D image data itself. The aim is to drive each vertex towards a ‘good’ position locally with respect to the image data. In the simple case we look for edges or surfaces in the image data close to the vertex. The current implementation looks at pixels along a normal to the model surface at each vertex (defined as the normal to the plane containing the vertex’s three neighbours), finds the strongest edge (intensity change) within a fixed distance and forces the vertex towards that edge. This is an extension to 3D of a well-used technique first suggested by Curwen and Blake [24].

For best results, strategic use of these forces is required. A three-stage deformation has been adopted as follows.

1. *Gross location using guiding forces*

The initial model is deformed under the action of guiding forces alone to move it into approximately the right position. This can be a trial-and-error process; if the fit obtained is not close enough (as decided by the human eye) then more guiding forces may be needed. This is especially the case when the vertices are initially randomly positioned as described in Section 6.2.5 above. The model’s ‘stable’ position is chosen to suit the particular situation: if deforming from an initially randomised position, the maximum continuity constraint ( $\forall i. \Phi_i = 0$ ) is used, however, if deforming from a previously discovered training example shape, the *initial* shape can be used as the stable shape ( $\forall i. \Phi_i = \phi_i^0$ ).

2. *Refinement using image forces*

Image forces are introduced to drive every vertex into its ideal position. Vertices which find no sufficient image data are pulled into position by their neighbours. The guiding forces are kept in place during this stage to maintain stability, and the maximum continuity constraint is used to ensure maximum smoothness.

### 3. *Fine tuning*

The guiding forces are removed so that previously guided vertices can adjust to their ideal location. This increases the tolerance for slightly misplaced guiding forces.

## 6.2.7 Example: 3D Hand Model

The technique was applied to the 3D volume data of human hands, obtained via MRI, as described in Section 6.2.1. A Simplex Mesh with 498 vertices was constructed by drawing a mesh on a surgical rubber glove stuffed with tissue paper, labelling each vertex, then entering the connectivity data by manual inspection. This was a laborious one-off task, but was deemed quicker and easier than implementing a mesh-building algorithm.

The processes detailed above were applied to each of 8 training images. For the first image, the mesh was deformed from an initially random position, as described in Section 6.2.5 (see Figure 6.3). Manually-located guiding forces were required for 80 vertices in order to ‘untangle’ the mesh. After 200 iterations, image forces were introduced at *all* vertices to draw the mesh towards edge data, and after 400 iterations the original guiding forces were removed to allow guided points to equilibrate (this is most noticeable around the wrist).

The resulting model was used as a starting position and stable shape for fitting the mesh to subsequent training images. Consequently, fewer guiding forces were needed (roughly 25 per example), and convergence was quicker (125 iterations as opposed to 460). Figure 6.4 shows fitting to an image where the thumb has moved. After 50 iterations (3rd frame) image forces were applied and after 100 iterations (4th frame) the guiding forces were removed.

The eight meshes thus produced were used to construct a PDM; Figure 6.5 shows the two most significant modes of variation for the model produced. The deformations are realistic, despite the small number of training examples.

It is worth noting that the small number of training examples used means that in this case the examples are effectively being used as *key frames* [10], but with strictly

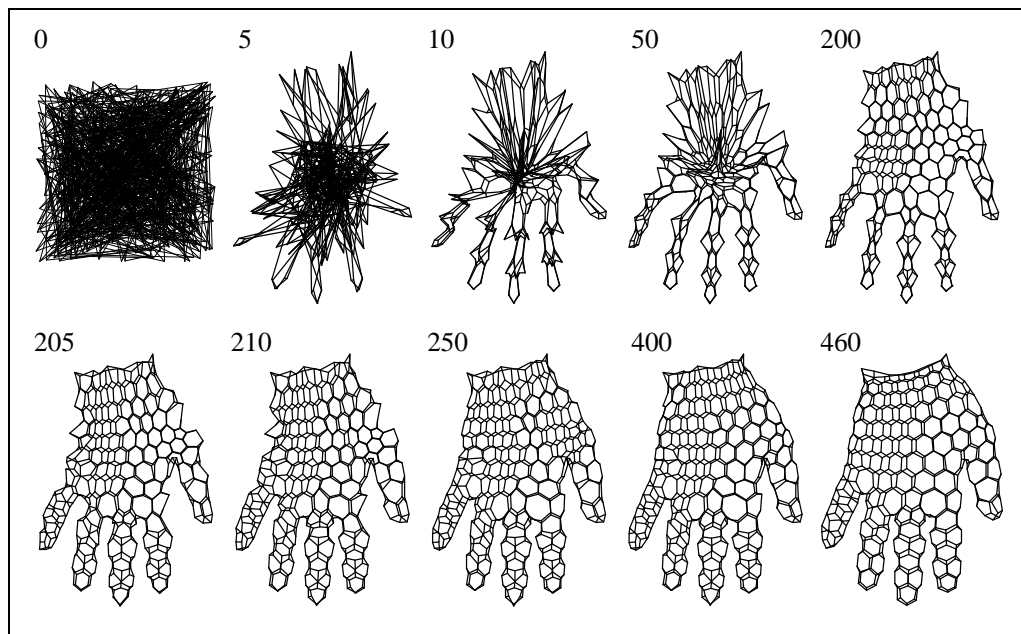


Figure 6.3: Deforming a Simplex Mesh from an initially random position to fit MRI data of a human hand (numbers show iterations).

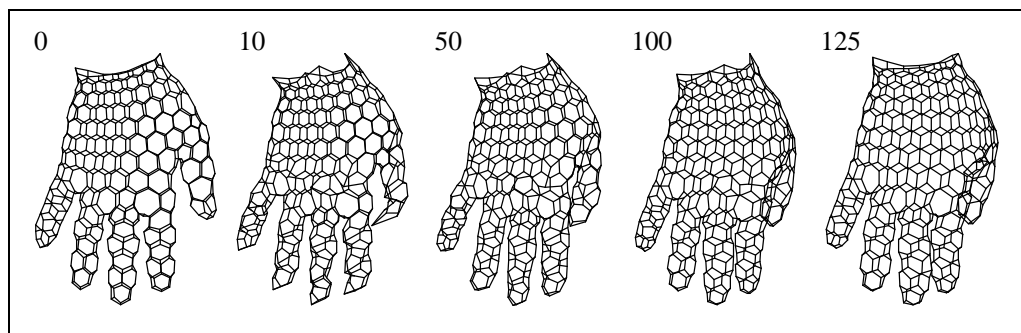


Figure 6.4: Deforming the first model to fit a second training image.

orthogonal degrees of freedom. However, of the 7 modes of variation produced, only 5 represented significant deformations; consequently it was decided to remove the last two from the model.

### 6.3 Tracking

There has been much work on using PDMs for object location and tracking in both 2D and 3D. In most of this previous work, the dimensionality of the model has matched that of the input image (i.e. a 2D model for 2D images [48, 4, 32] or a 3D PDM for 3D images [38]). Work on matching a 3D model to a 2D image has so far assumed a ground plane constraint and only one degree of rotational freedom [67, 74].

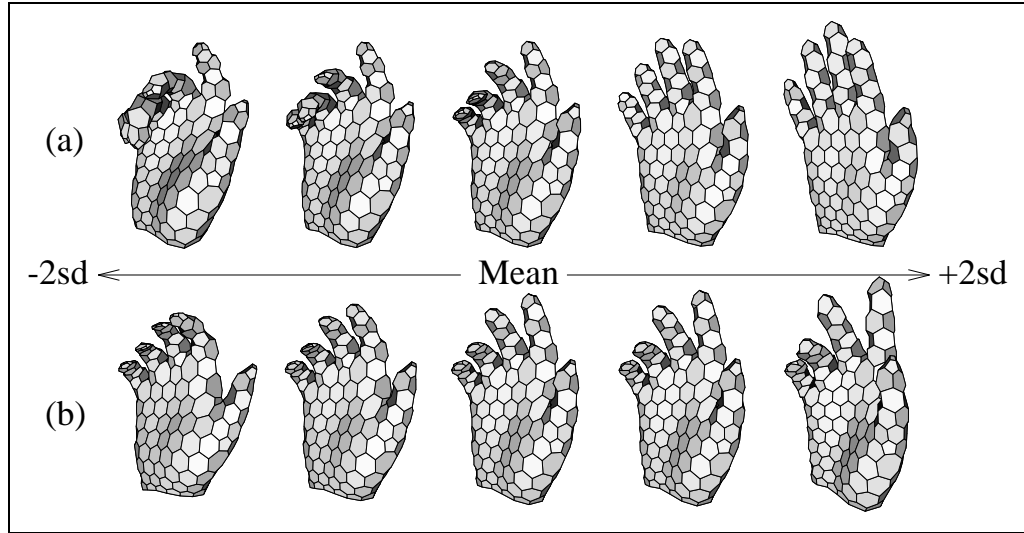


Figure 6.5: The first (a) and second (b) modes of variation of the Point Distribution Model produced.

In this work an attempt is being made to match a 3D PDM to a 2D image under full 6 degrees of freedom (DoF) plus deformations.

An adaptation of the ASM (Section 2.2.3) is used for this task. The key to this model-based approach is to find the set of model parameter values that cause the model to fit best the image data. In this case the parameters are a translation vector  $\mathbf{u} = (u, v, w)$ , a  $3 \times 3$  orthonormal rotation matrix  $\mathbf{R}$ , a scale factor  $s$  and the five significant deformation parameters  $b_j$  (giving a total of 12 DoF). Iterative pose refinement is used: given a fair initial estimate at an object's location, local image information (e.g. edge data) is extracted and used to calculate a small change in the model parameters which will improve the fit.

To compare model and image, it is necessary to project the model onto the image. The model is first deformed from the mean shape  $\bar{\mathbf{x}}$  using the standard PDM equation:

$$\mathbf{x} = \bar{\mathbf{x}} + \sum_{j=1}^t b_j \mathbf{p}_j \quad (6.4)$$

The deformed model  $\mathbf{x}$  is then rotated, scaled and translated into the *posed* model  $\mathbf{X}$ , such that the position  $\mathbf{X}_i$  of the  $i^{th}$  landmark is given by:

$$\mathbf{X}_i = s\mathbf{R}\mathbf{x}_i + \mathbf{u} \quad (6.5)$$

$\mathbf{X}$  is projected into the 2D image using an orthographic projection (simply by

discarding the  $z$ -coordinates). This allows projections and inverse projections to be calculated quickly and, with a sufficiently distant camera, produces negligible distortion. Of course,  $z$ -position information is lost but, assuming a fixed-size object and known intrinsic camera properties,  $z$ -position can be inferred from scaling (this is effectively a *scaled* orthographic projection).

As mentioned above, the intention is to find values for  $\mathbf{u}$ ,  $s$ ,  $\mathbf{R}$  and the  $b_j$  which give the best match between model and image. These parameters are updated iteratively using image evidence, specifically by finding the best local movement for individual model landmarks. The result is a collection of suggested landmark movements (in the form of  $(dx, dy)$  pairs) which undergo statistical voting to change the overall model pose.

Because the process is iterative, it extends naturally to tracking an object over a time sequence of images: the model's final position in one image is used as the starting position for the next image.

### 6.3.1 Gathering image evidence

It is required to find suggested movements for individual landmarks by examining image data. The evidence that can be gathered from a 2D image with respect to a 3D model is limited. Firstly, if a hand is to be tracked unmarked, the only reliable position evidence that can easily be extracted is from edge data. Also, only a subset of the model landmarks will lie on the model's boundary in any particular 2D projection; no movement evidence can be collected for any other landmarks as there will be no corresponding edge in the 2D image (Shen observed this in his work on vehicle model building [67]). The ASM tracker, as originally described by Cootes and Taylor, does not cater for this situation; a landmark *weighting* scheme has consequently been introduced in the algorithm described below. The *aperture problem* [41, 36] is also experienced, in two separate guises. Firstly, the desired position *along* any discovered edge (in the  $x$ - $y$  plane) is uncertain. Secondly, because a single 2D image is being used for input, no depth information is available i.e. the  $z$ -coordinate of an edge is uncertain.

The data required are a suggested movement  $d\mathbf{X}_i$  for each model landmark  $i$ , along with an associated weighting  $W_i$  indicating how strong the evidence is for this movement. Evidence is only collected for landmarks which lie on the projected model boundary. For each landmark  $i$ , the unit normal  $\mathbf{n}_i$  to the model surface is found, defined as the normal to the plane containing landmark  $i$ 's three mesh

neighbours. If  $\mathbf{n}_i$  subtends an angle of less than  $30^\circ$  to the  $x$ - $y$  plane, landmark  $i$  is deemed to lie on or very near to the projected model boundary (this is imprecise, but fast to calculate), and a corresponding image edge is likely. A line of pixels is extracted from the image either side of the landmark and in the direction of the projection of  $\mathbf{n}_i$  into the  $x$ - $y$  plane. The greatest intensity change (i.e. strongest edge) along this line is found and  $d\mathbf{X}_i$  is set accordingly (its  $z$  component is set to zero).  $W_i$  is set to the magnitude of the intensity change. If  $\mathbf{n}_i$  subtends an angle of greater than  $30^\circ$  to the  $x$ - $y$  plane, no image evidence is collected;  $d\mathbf{X}_i = \mathbf{0}$  and  $W_i = 0$  are accordingly set.

Figure 6.6 shows an enlargement of the feature extraction on part of the hand. The model is shown in white and the suggested movements, where discovered, are shown as black lines. To increase speed, not every pixel is sampled along the normal; this explains why some of the black lines do not quite meet the image edges.

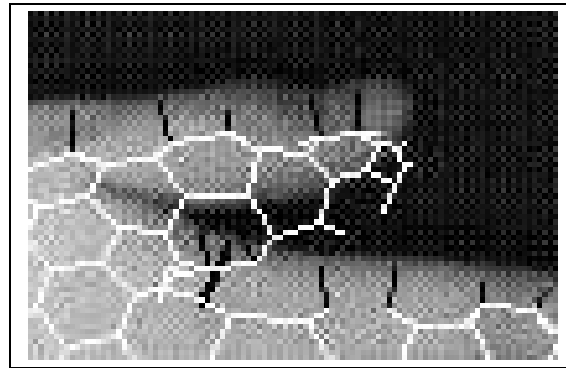


Figure 6.6: Suggested landmark movements.

### 6.3.2 Updating the model position

Given a suggested movement  $d\mathbf{X}_i = (dx_i, dy_i, 0)$  for each landmark  $i$ , and an associated weighting  $W_i$ , the task is to update the model parameters  $\mathbf{u}$ ,  $s$ ,  $\mathbf{R}$  and the shape parameters  $b_j$ . A weighted least-squares solution is used, which involves finding values for  $\mathbf{u}' = \mathbf{u} + d\mathbf{u}$ ,  $s' = s + ds$ ,  $\mathbf{R}' = d\mathbf{R}\mathbf{R}$  and  $b'_j = b_j + db_j$  that minimise  $\varepsilon$  in:

$$\varepsilon = \sum_{i=0}^N W_i \|\mathbf{X}_i + d\mathbf{X}_i - \mathbf{X}'_i\|^2 \quad (6.6)$$

where  $\mathbf{X}'_i$  is defined in terms of  $\mathbf{u}'$ ,  $s'$ ,  $\mathbf{R}'$  and the  $b'_j$  using equations (6.4) and (6.5). Substituting in gives:



$$\varepsilon = \sum_{i=0}^N W_i \|\mathbf{X}_i + d\mathbf{X}_i - (s'\mathbf{R}'(\bar{\mathbf{x}} + \sum_{j=1}^t b'_j \mathbf{p}_j)_i + \mathbf{u}')\|^2 \quad (6.7)$$

Hill *et al* [36] describe an iterative solution to this problem, whereby the *rigid* parameters (translation, rotation and scale) are calculated separately from the deformation parameters. Hill *et al* use a quaternion approach to solve for the rotation matrix, as described by Horn [42]. Instead, an alternative approach is taken here, using a singular valued decomposition (SVD), as described by Arun *et al* [2]. They give an *unweighted* solution; this is very easily adapted into a weighted version because each component of the solution is some form of summation over the control points, which can be replaced with a weighted summation<sup>1</sup>. The solutions for  $d\mathbf{u}$ ,  $ds$ ,  $d\mathbf{R}$  and the  $db_j$  are as follows:

$$d\mathbf{u} = \frac{\sum_{i=1}^N \mathbf{W}_i d\mathbf{X}_i}{\sum_{i=1}^N \mathbf{W}_i} \quad (6.8)$$

where  $\mathbf{W}_i = (W_i, W_i, 0)$ , and the products and divisions are performed element-wise.  $d\mathbf{u}$  is then used in the calculation of  $ds$  and  $d\mathbf{R}$ :

$$ds = \sqrt{\frac{\sum_{i=1}^N W_i \|\mathbf{X}_i + d\mathbf{X}_i - (\mathbf{u} + d\mathbf{u})\|^2}{\sum_{i=1}^N W_i \|\mathbf{X}_i - \mathbf{u}\|^2}} \quad (6.9)$$

To calculate  $d\mathbf{R}$  a weighted version of Arun's SVD method [2] is used (the derivation of this method is beyond the scope of this work). The  $3 \times 3$  matrix  $\mathbf{H}$  is first found.

$$\mathbf{H} = \sum_{i=1}^N W_i (\mathbf{X}_i - \mathbf{u})(\mathbf{X}_i + d\mathbf{X}_i - (\mathbf{u} + d\mathbf{u}))^T \quad (6.10)$$

and then the SVD of  $\mathbf{H}$  is calculated:

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \quad (6.11)$$

where, in this case,  $\mathbf{U}$  and  $\mathbf{V}$  are  $3 \times 3$  orthonormal matrices and  $\mathbf{\Lambda}$  is a  $3 \times 3$  diagonal matrix (see [59] for more details).  $d\mathbf{R}$  is then given by:

$$d\mathbf{R} = \mathbf{V}\mathbf{U}^T \quad (6.12)$$

---

<sup>1</sup>The argument being that having an *integer* weight  $W$  is equivalent to having  $W$  control points of unit weight at the same location; this can be extended to non-integer weights by considering that the use of arbitrarily large integer weights gives exactly the same solution.

Before calculating the  $db_j$ , the effects of  $d\mathbf{u}$ ,  $ds$  and  $d\mathbf{R}$  are removed from each  $d\mathbf{X}_i$ :

$$d\mathbf{X}_i' = s'\mathbf{R}'\mathbf{x}_i + \mathbf{u}' - \mathbf{X}_i + d\mathbf{X}_i \quad (6.13)$$

The  $db_j$  are then calculated thus:

$$db_j = \frac{\mathbf{p}_j^T \mathbf{W} d\mathbf{X}_i'}{\mathbf{p}_j^T \mathbf{W} \mathbf{p}_j} \quad (6.14)$$

where  $\mathbf{W} = \text{diag}(W_1, W_1, W_1, W_2, W_2, W_2, \dots, W_N)$ .

As mentioned above, this solution is iterative. In fact only a single iteration is performed for reasons of speed, given that the tracker is iterative over frames anyway.

Although the weighted least-squares approach does find a suitable solution, it has been noted that convergence can be hampered by the *aperture problem*: if an edge is found along a model normal, the landmark is encouraged towards that point. However the landmark's true resting position might be further along the edge. Also, when tracking from a 2D image,  $dz = 0$  must be assumed, because there is no evidence to the contrary. The true resting position of the landmark may require  $dz \neq 0$ .

Hill *et al* propose a method to overcome these problems using *directional weights* [36], whereby landmarks are made free to 'slide' along target edges or across target planes. Hill *et al*'s solution involves the inversion of large weight matrices; it would be useful to avoid this computationally expensive operation. It is possible to improve on the 'simply' weighted least-squares approach (as described above) without incurring too much computational cost. Directional information from the suggested landmark movements can be used to determine how much the evidence from a particular landmark should contribute towards updating a particular parameter. For example, if the normal to landmark  $i$  is parallel to the  $x$  axis, its image evidence should make no contribution in calculating the  $y$  component of  $d\mathbf{u}$ . This tactic is put into practice as follows: the least-squares equations are as for the 'simply' weighted approach; however, in calculating the change  $dq$  to a general model parameter  $q$ , the weighting  $W_i$  is replaced with  $W_{q,i}$ , which is calculated from  $W_i$  and  $d\mathbf{X}_i$  specifically with respect to parameter  $q$ .

The following calculations are used:

$$\mathbf{W}_{\mathbf{u},i} = W_i(|dx_i|, |dy_i|, 0) \quad (6.15)$$

$$W_{s,i} = \frac{W_i |d\mathbf{X}_i \cdot (\mathbf{X}_i - \mathbf{u})|}{|\mathbf{X}_i - \mathbf{u}|} \quad (6.16)$$

$$W_{\mathbf{R},i} = \sqrt{W_i^2 - W_{s,i}^2} \quad (6.17)$$

$$\mathbf{W}_b = \text{diag}(W_1|dx_1|, W_1|dy_1|, 0, \dots, W_N|dx_N|, W_N|dy_N|, 0) \quad (6.18)$$

where  $\mathbf{W}_b$  is used in place of  $\mathbf{W}$  in the calculation of the  $db_j$ .

It is important to appreciate that the above weighting scheme does not fully address the aperture problem: the weightings are calculated independently for each model parameter; no allowance is made for the interdependency of the parameters. However, it provides an improvement over the simply-weighted scheme at virtually no extra computational cost.

### 6.3.3 Handling Self-occlusion

Previous work (due to Rehg [63]) has made use of *layered templates* to model self-occlusion. A simpler method is adopted here whereby the visibility of each vertex is determined individually by considering whether any model facets lie in front of it (with respect to the image plane).

A facet bounded by a set of  $k$  vertices,  $F = \{\mathbf{x}_{f_1}, \mathbf{x}_{f_2}, \dots, \mathbf{x}_{f_k}\}$ , is said to be occluding a vertex  $\mathbf{x}_v$  if:

$$\begin{aligned} x_v > x_{\min} \quad \text{and} \quad x_v < x_{\max} \quad \text{and} \\ y_v > y_{\min} \quad \text{and} \quad y_v < y_{\max} \quad \text{and} \\ z_v > z_{\min} \quad \text{and} \quad \forall i.v \neq f_i \end{aligned} \quad (6.19)$$

where  $\mathbf{x}_i = (x_i, y_i, z_i)$  for the  $i^{\text{th}}$  vertex, and

$$\begin{aligned} x_{\min} &= \min |_i x_{f_i}; & x_{\max} &= \max |_i x_{f_i}; \\ y_{\min} &= \min |_i y_{f_i}; & y_{\max} &= \max |_i y_{f_i}; \\ z_{\min} &= \min |_i z_{f_i}; \end{aligned} \quad (6.20)$$

Figure 6.7 illustrates this situation. This criterion is of course not precise; it uses the bounding box for the facet, and thus over-estimates the extent of occlusion. A non-occluded vertex which is deemed occluded does not cause a catastrophic problem; it just means that no image evidence will be collected for that vertex, and tracking might be slightly degraded. This is preferred over a false negative, which

forces an attempt to track from an occluded vertex.

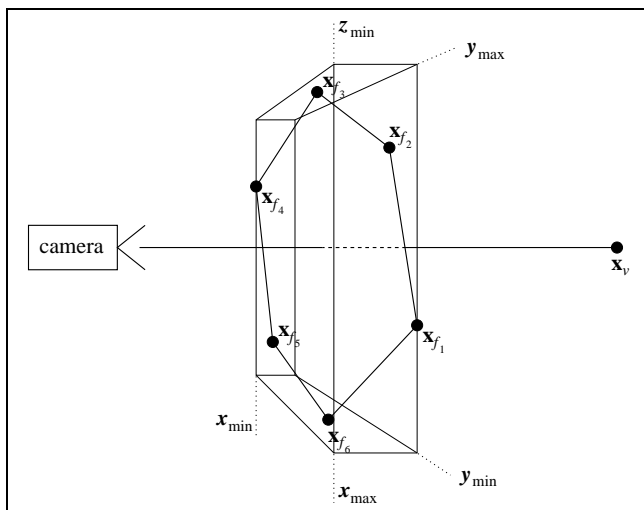


Figure 6.7: Self occlusion of a vertex  $\mathbf{x}_v$  by a facet  $F$ .

The drawback of this approach is that it is very slow. A ‘brute-force’ implementation has been used here, whereby every potential boundary vertex is checked against every facet. With approximately 200 such vertices in any one pose and 1000 facets this represents approximately 1 million comparisons per iteration. It is almost certain that heuristics can be used to speed this up, either via a pre-learned look-up table of all geometrically possible occlusions, or perhaps using some form of hierarchical grouping of facets which allows whole groups to be eliminated from the search all at once. Such techniques are commonly found in the field of computer graphics [28].

## 6.4 Evaluation

An experimental mock-up of the tracker was constructed using a colour camera pointing downwards at a homogeneous dark surface and connected to a Silicon Graphics Indy workstation running at 134MHz. Images were captured from the camera and the tracking algorithm was applied in real-time. Images were echoed to the workstation screen, with the hand model superimposed. The tracking rate was approximately 18 frames per second (without occlusion detection), not including image echoing and graphical model rendering. To avoid the global search problem (a hand must be found before it can be tracked), the model was initialised centrally in the image and only began tracking when a hand was moved into position ‘under’ it. This event was detected by the presence of strong edges at over 80% of the

model boundary landmarks. The user could see the model tracking his or her hand, providing useful feedback.

The tracker was exercised rigorously, with a diversity of movements and deformations being performed many times over. Figure 6.8 shows some snapshots from these experiments.

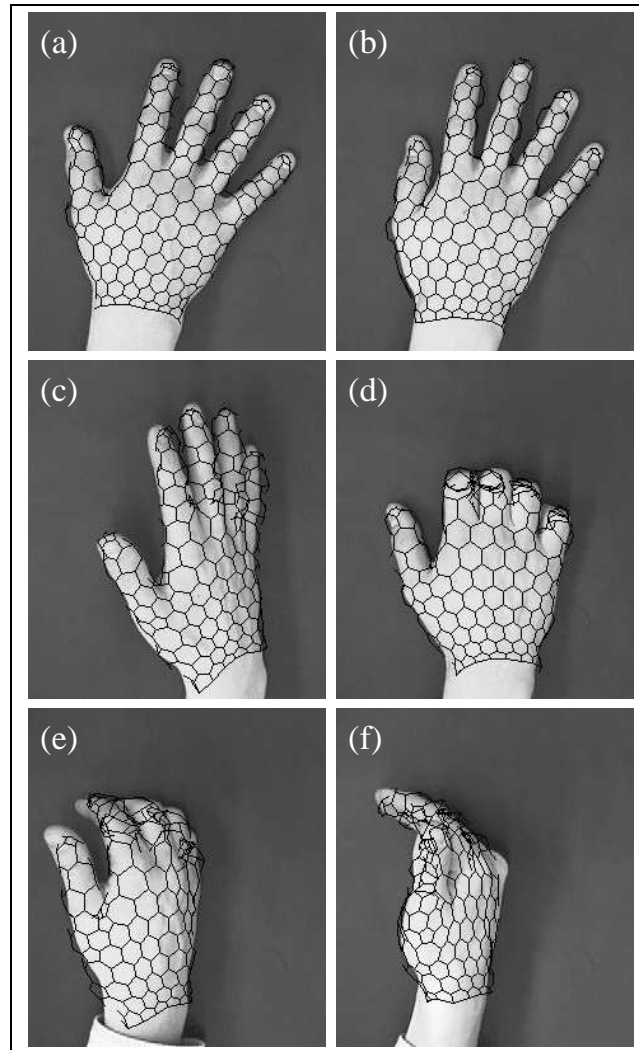


Figure 6.8: Snapshots from hand tracking experiments using the 3D PDM.

A qualitative evaluation of the tracker is as follows:

- Changes in  $x$  and  $y$  translation, scale and rotation in the  $x$ - $y$  plane were tracked with no difficulty, irrespective of the hand pose.
- Rotations out of the  $x$ - $y$  plane initially caused problems. In particular, the transition from (a) to (c) in Figure 6.8 produced a decrease in scale instead

of the expected rotation. This is because much of the evidence collected from the 2D image (i.e. the sides of the hand moving inwards) is consistent with such a change, and the only evidence to the contrary comes from the static position of the fingertips (the wrist is unmarked and provides no evidence). To circumvent this problem, the model size was fixed at a constant value. The rotations were then tracked correctly, at the cost of not being able to track gross movements towards or away from the camera.

- Rotations out of the  $x$ - $y$  plane were tracked better with the size constraint. However, success depended very much on the starting pose. Most problems were caused by ambiguity: because the hand is roughly planar, positive and negative rotations of the hand viewed from either a direct or sideways-on viewpoint appear very similar in an orthogonal projection (e.g. the transition from (a) to (c)). Consequently the model sometimes rotated the wrong way.
- Clearly visible deformations were tracked well; for example, the transition from (a) to (b). Self-occluded deformations were tracked less well, since there is little image evidence to support them. An example is the transition from (a) to (d), which was always tracked accurately, but more slowly than visible deformations.
- In the case where occlusion detection was not used (in order to increase speed), there were problems: occluded vertices tended to be ‘attracted’ to the nearest *visible* edge. This occurred in poses such as (e) and (f). When occlusion detection was used, these problems were no longer experienced.

The system was also tested against a cluttered background. Figure 6.9 shows an example. It is interesting to note here that the performance was *almost* as good as for the homogeneous background, suggesting that the overall approach is generally fairly robust to image clutter. However, this can be attributed at least partially to the use of a colour filtering algorithm to lessen the effects of clutter [32].

To summarise, tracking was not a failure, but also not as robust and not as fast as the 2D trackers described in previous chapters. This is perhaps to be expected as a much more complex (perhaps *too* complex) model is being used.



Figure 6.9: Tracking against a cluttered background.

## 6.5 Extensions

### 6.5.1 Stereo

The use of two or more camera views can help to improve robustness of the tracker. The silhouettes of an object as seen from different views generally correspond to different sets of vertices in the model mesh, so the amount of image evidence that can be collected is increased. More importantly, the combination of evidence from two or more non-parallel image planes provides important *depth* information, and also helps to resolve many of the ambiguities described in Section 6.4.

The nature of the system being used means that the extension to two or more cameras is relatively simple. The object model is projected separately into each camera view<sup>2</sup> and image evidence (in terms of suggested landmark movements) is collected from each view. These movement vectors are transformed into world coordinates and then combined into a single set of movements; the new model pose is then calculated exactly as for the single camera system. In most cases, a particular vertex will only have provided evidence in, at most, one view. In a few cases where a vertex has measurements from two or more views, the strongest measurement is used<sup>3</sup>.

An experiment was conducted using such a system. Two views were achieved using a single camera by way of a carefully-placed mirror (see Figure 6.10), effectively to produce a stereo tracker.

---

<sup>2</sup>Each camera must be calibrated in order to find the transformations from world coordinates to camera coordinates and *vice versa*.

<sup>3</sup>A more elaborate scheme might seek to combine evidence from different views by finding the intersection of the uncertainty plane from each view, thus better constraining the vertex movement.

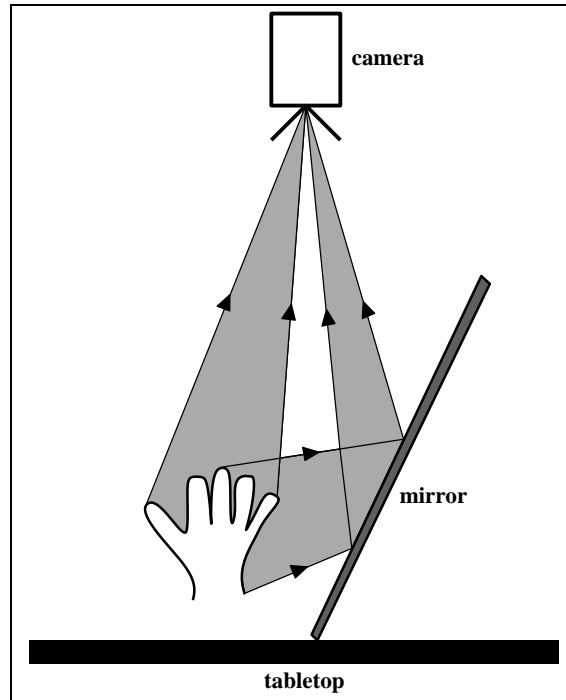


Figure 6.10: The stereo tracking environment; two views are achieved from a single camera using a mirror.

An object of known dimension was used to calibrate the system (see Figure 6.11); four specific reference points were located manually in each view and the relevant orthogonal transformation parameters calculated. Because a mirror was being used, one of the transformations also included a reflection.

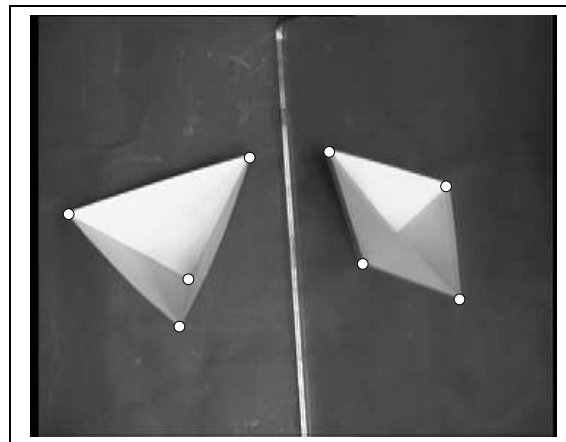


Figure 6.11: Calibration image for the stereo environment, with manually-located calibration points shown.

A sequence of hand movement was pre-recorded and the stereo tracker was applied off-line, using full occlusion detection. The output from the tracker was used



to drive a simple 3D drawing tool in which the index fingertip was the stylus and the thumb position determined whether or not to commence drawing. Figure 6.12 shows some snapshots from the sequence, and Figure 6.13 shows several views of the line drawing produced, demonstrating that it is indeed a 3D drawing.

### 6.5.2 Polar Coordinates

As described in Section 3.6, it is possible to apply the Cartesian-Polar Hybrid PDM to 3D objects. Figure 3.13 shows the results of using the 3D hand training data in this context. This approach is used in order to improve the *specificity* of the model produced, and thus hopefully to improve tracking robustness. In practice however, using such a model for tracking resulted in poor performance: instabilities occurred such that the model fingers would flail uncontrollably with little regard to image evidence.

There are no good explanations for this behaviour; however, several possibilities (none of which are entirely satisfactory) are given below:

1. Polar-mapped vertices close to their centre of rotation are causing disproportionate suggested angular movement (although a weighting factor has been included to counteract this behaviour).
2. Certain combinations of projective transformations and polar mappings may give rise to unstable pose change calculations.
3. The particular model being used is unsuitable, perhaps due to the small number of training examples giving rise to poor polar modelling. However, to the eye it appears to be an ideal model.
4. There is a bug in the code that the author has not been able to find.

Considerable effort has been expended on this problem, but to no avail. In the case of the first two suggestions, the problem is associated with the use of a data-driven tracking algorithm; use of a fitness-based approach, such as a Genetic Algorithm or the CONDENSATION algorithm (see Section 2.2.4) might alleviate such a problem. This line of pursuit has not been investigated because of the apparent limitations to the 3D modelling approach in general.

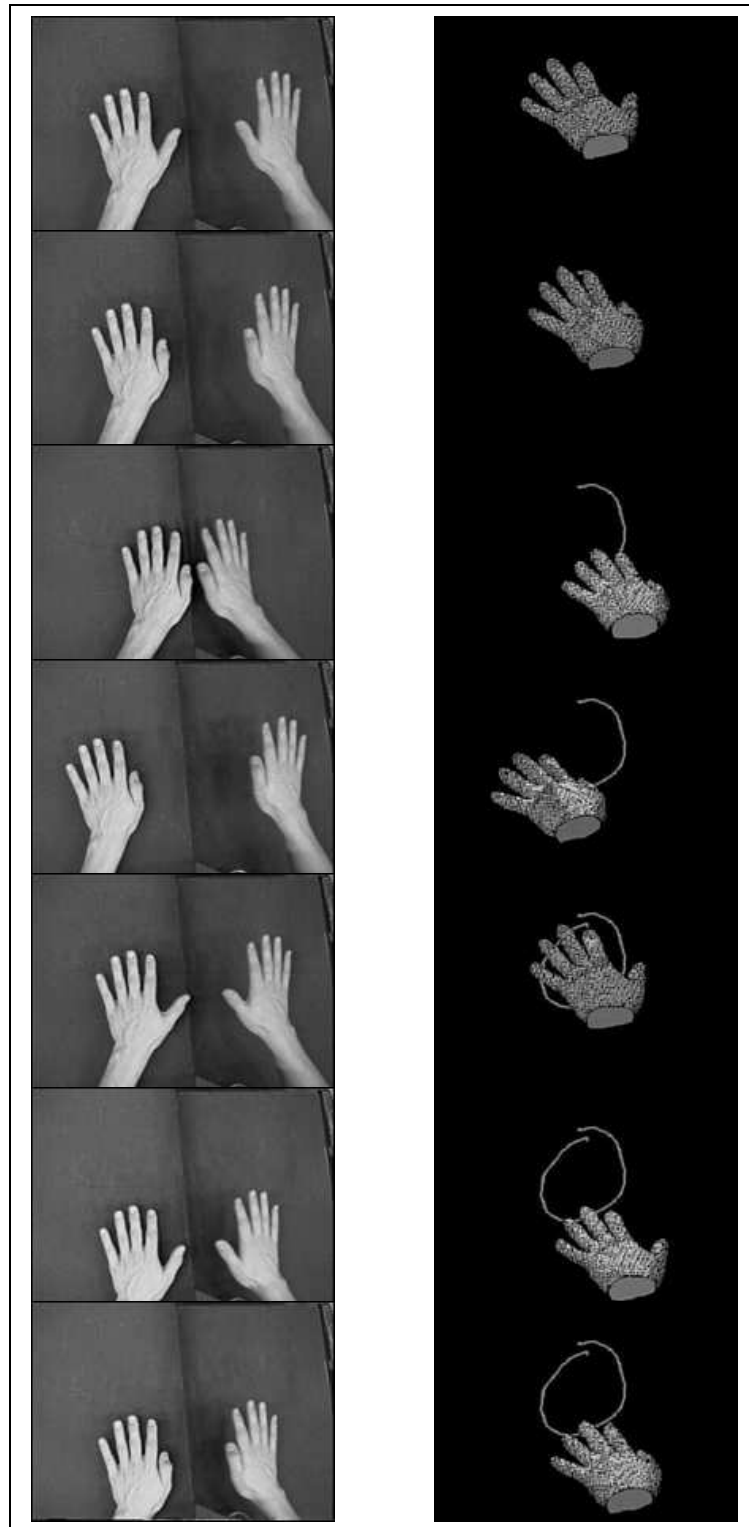


Figure 6.12: Snapshots from a 3D stereo tracking sequence; the video input, which uses a mirror to give two views of the hand (left), and the tracking result, shown from a different angle (right).

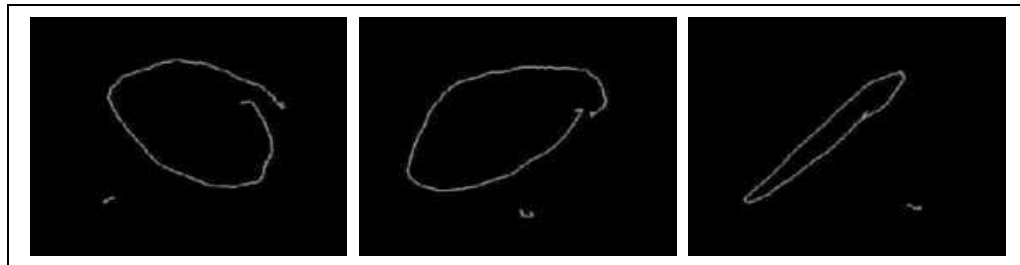


Figure 6.13: Three views of the line drawing output from the 3D stereo tracker sequence.

## 6.6 Discussion and Conclusions

It has been shown how a 3D PDM can be constructed from 3D volumetric training images via the use of physically-based models. The technique is not fully automatic but *guided*; with a well-designed interface user effort would be minimal.

A specific procedure using a Simplex Mesh in 3D has been described. Some obvious generalisations apply:

- The physical model used does not have to be a Simplex Mesh. Any physical modelling technique could be used, for example the less constrained meshes used in [14], or the Finite Element Method models used in [58].
- The technique can be applied in a 2D situation using a 2D physical model such as a Snake.
- Where MRI data is not available it may be possible to use a sparser input source, such as range data, since internal model forces keep non-visible vertices in position (and in any case, statistical models are fairly robust to small errors in point position). However, there is still the question of how to obtain an initial shape.

A description has also been given of a system for tracking 3D objects in real time from a single camera, using the aforementioned models.

It has been shown how information can be extracted from a 2D image to move and deform a 3D model; the instances where this is most and least successful have been highlighted and discussed.

The use of a 3D model means that pose inference becomes a trivial task. Also, because the model is a PDM, object shape is described in terms of a small number of scalars, providing a good starting point for shape inference. However, the cost of these benefits is reduced speed and robustness as compared to a purely 2D tracker.

An obvious extension is to use the 3D hand model in a CONDENSATION tracking environment. This would perhaps improve tracker performance in the event of ambiguity, as well as improving general robustness. There would of course be a considerable speed penalty.

# Chapter 7

## Conclusions

---

### 7.1 Summary of Work

The focus of this work has been the generation of deformable shape models for computer vision-based tracking, as well as suitable tracking algorithms to complement such models.

Various forms of statistical models have been described; object shape being represented by a set of landmarks (generally joined together to form a contour) and principal component analysis (PCA) being used as a basis for learning shape and deformation.

The shortcomings of existing statistical shape models have been noted: lack of specificity, lack of compactness, and the effort involved in collecting training data. Extensions to existing methods have been developed to overcome these problems.

In Chapter 3 a method for better modelling objects which experience bending or pivotal motion was described. This is made possible in a statistical framework by explicitly remapping selected model landmarks into a polar domain, thus effectively ‘linearising’ polar deformations with respect to the PCA. It was noted that the polar model is a useful extension to statistical modelling, but only in certain specific cases.

In Chapter 4 a more general approach was described, detailing how arbitrary non-linearity in object deformation can be modelled. More *specific* models of shape were achieved via a *hierarchical* PCA; the first level provides a dimensional reduction

of the shape space, and the second level captures non-linear detail using a piecewise-linear approximation, formulated as a union of hyperellipsoid-bounded regions. It was observed that it is possible to build good Hierarchical PDMs from automatically-captured training data, but that in this case, tracking performance using an Active Shape Model is less than satisfactory.

In Chapter 5 an alternative tracking algorithm was described which enables tracking from fully-automatically generated models, as well as being able to track discontinuous shape changes. The Hierarchical PDM (HPDM) is used as the underlying model of shape, and a model of shape dynamics is learned from training sequences of characteristic object movement. The dynamic model is represented as a Markov process, the Markov states being the linear patches from the HPDM. The CONDENSATION algorithm provides a non-deterministic framework in which to apply these dynamics. It was shown how this new system can be used for tracking using automatically-captured training data.

In Chapter 6 it was noted that pose inference from 2D shape models is a non-trivial task. Consequently, the construction of 3D statistical shape models was described, as was tracking with such models, with a view to easing this task. It was discovered that, although pose inference is eased, the speed and robustness of tracking with such models is much poorer overall than with the 2D models.

## 7.2 Discussion

An underlying theme in this thesis had been the learning of various object features from *real-life* training examples. This appears, in general, to be a very promising approach to computer vision: the models produced are necessarily true-to-life and the model building process lends itself to automation. The fact that this is how the human visual system functions perhaps reinforces the argument. However, it is important to think carefully about the models being built.

Taking too general an approach is certainly a mistake. Take, for example, the apocryphal story of the neural net-based military tank detection system. Full colour, high resolution images were fed straight into the neural net with two classifications to learn: ‘tank’ or ‘no tank’. The 100% success rate was surprisingly high, until it became apparent that all the ‘tank’ training examples were slightly darker and the neural net was simply acting as an averaging thresholder.

Conversely, models which are too specific are only applicable in a limited domain (e.g. using articulated skeleton models for hands or whole bodies), which is fine for

a particular application, but not when attempting to develop general techniques.

The use of boundary features in models lies in the middle ground. The majority of (interesting) objects have a coherent boundary, which is useful for both training *and* tracking; boundaries are also very distinctive (good for recognition) and invariant to lighting conditions.

Statistics have also featured prominently throughout this work. The representation of real life is inherently imprecise (some might say chaotic); the field of statistics provides the most rigorously formalised tools for dealing with uncertainty, and also lends itself well to computation. It could be argued that neural networks might be more appropriate for a task which is attempting to emulate a human process, but present-day technology is several orders of magnitude away from true life biological neuron counts, and at least with pure statistics there is some sort of order within the chaos.

Finally, it is worth mentioning that the work in this thesis is presented in a slightly different order to that in which it was conducted. The work on polar models came first, followed by a strong belief that 2D models were not powerful enough for tracking 3D objects; they were ‘too limited’, not by the use of boundary models, but by the underlying statistical model. The 3D work followed on from that, together with the discovery that tracking with 3D models was not necessarily the way forward. Finally, the 2D emphasis returned and the more general HPDM models were derived, along with a suitable tracking process. It is perhaps ironic that the best solution (with respect to tracking) turned out to be the most simple and elegant one.

### 7.3 Future Work

There are many avenues that, given more time, it would be interesting to explore. The models that have been built are quite general and, as such, might be useful as ‘enabling technology’ for many different applications. Simply exercising the HPDM on different modelling tasks would be desirable, either to validate its generality or to discover any weaknesses and refine the technique, and also to see how well it scales up to larger problems.

A key issue has been the automation of the training process. So far, a rather crude method of training data acquisition has been used. The fact that this is possible gives credit to the power of the HPDM and it would be interesting to see how far this power can stretch. For example, in 3D a fair amount of effort was expended on developing algorithms to give a good point correspondence between

training examples; however, if using an HPDM this might not be so crucial.

Also on the subject of automation is the issue of fitness function learning. The fitness-based tracking algorithm described in Chapter 5 makes heavy use of a fitness function which has been entirely hand-crafted, with no evaluation of how good a discriminant it is between good and bad model/object matches. It would be favourable to be able to *learn* a suitable fitness function in order to improve tracking performance, or at least to prove that the existing *ad hoc* fitness function is in fact the right one.

The most interesting possibility is a link-up between the 2D and 3D models used in this work. The 2D models excel in tracking and the 3D models are useful for object pose inference and graphical rendering. It would seem sensible to use each model for its strengths: track with the 2D model, and infer pose and/or render using the 3D model. For this it would be necessary to learn the mapping from 2D model shape to 3D model shape and orientation. This might be possible by constructing artificial training sequences of silhouettes with the 3D model in known poses, and then using these to learn a suitable mapping.



# Bibliography

- [1] T. Ahmad, C.J. Taylor, A. Lanitis, and T.F. Cootes. Tracking and recognising hand gestures using statistical shape models. In *Proc. BMVC*, pages 403–412, Birmingham, UK, 1995. BMVA Press.
- [2] K.S. Arun, T.S. Huang, and S.D. Blostein. Least-squares fitting of two 3D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.
- [3] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *Proc. 3rd ECCV*, pages 299–308, Stockholm, Sweden, 1993. Springer-Verlag.
- [4] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In IEEE Computer Society Press, editor, *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 194–199, November 1994. Also available as [ftp://agora.leeds.ac.uk/scs/doc/reports/1994/94\\_11.ps.Z](ftp://agora.leeds.ac.uk/scs/doc/reports/1994/94_11.ps.Z).
- [5] A. Baumberg and D.C. Hogg. An adaptive eigenshape model. In *Proc. BMVC*, pages 87–96, Birmingham, UK, 1995. BMVA Press.
- [6] A. Baumberg and D.C. Hogg. Generating spatiotemporal models from examples. In *Proc. BMVC*, pages 413–422, Birmingham, UK, 1995. BMVA Press.
- [7] P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In B. Buxton and R. Cipolla, editors, *Computer Vision - ECCV96*, volume 2 of *Lecture Notes in Computer Science (1065)*, pages 683–695, Cambridge, UK, 1996. Springer-Verlag.
- [8] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145, 1993.

- 
- [9] A. Blake and M.A. Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *Proc. ACM Siggraph*, pages 185–192, 1994.
- [10] A. Blake, M.A. Isard, and D. Renyard. Learning to track the visual motion of contours. *Artificial Intelligence Journal*, 78:179–212, 1995.
- [11] R. Bowden, T.A. Mitchell, and M. Sahardi. Real-time dynamic deformable meshes for volumetric segmentation and visualisation. In *Proc. BMVC*, pages 310–319, Colchester, UK, 1997. BMVA Press.
- [12] C. Bregler and S. Omohundro. Surface learning with applications to lipreading. In J D Cowan, G Tesauro, and J Alspector, editors, *Advances in neural information processing systems 6*, 1994.
- [13] R.A. Brooks. Symbolic reasoning among 3-D models and 2-D images. In M. Brady, editor, *Computer Vision*, pages 285–348. North-Holland Publishing Company, 1981.
- [14] A.J. Bulpitt and N.D. Efford. An efficient 3D deformable model with a self-optimising topology. In *Proc. BMVC*, Birmingham, UK, September 1995.
- [15] T.F. Cootes and C.J. Taylor. Active shape models - ‘Smart Snakes’. In *Proc. BMVC*, pages 266–275, Leeds, UK, 1992. Springer-Verlag.
- [16] T.F. Cootes and C.J. Taylor. Combining point distribution models with shape models based on finite element analysis. In *Proc. BMVC*, pages 419–428, York, UK, 1994. BMVA Press.
- [17] T.F. Cootes and C.J. Taylor. Data driven refinement of active shape model search. In *Proc. BMVC*, pages 383–392, Edinburgh, UK, 1996. BMVA Press.
- [18] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variation. In *Proc. BMVC*, Colchester, UK, 1997. BMVA Press.
- [19] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Training models of shape from sets of examples. In *Proc. BMVC*, pages 9–18, Leeds, UK, 1992. Springer-Verlag.
- [20] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active Shape Models - their training and applications. *Computer Vision and Image Understanding*, 61(2), January 1995.

- 
- [21] T.F. Cootes, C.J. Taylor, and A. Lanitis. Active shape models: Evaluation of a multi-resolution method for improving image search. In *Proc. BMVC*, pages 327–336, York, UK, 1994. BMVA Press.
- [22] T.F. Cootes, C.J. Taylor, A. Lanitis, D.H. Cooper, and J. Graham. Building and using flexible models incorporating grey-level information. In *Proc. ICCV*, Berlin, Germany, 1993.
- [23] J.L. Crowley and J. Martin. Appearance-based techniques for recognition of hand gestures. Technical report, Institut National Polytechnique de Grenoble, 1996.
- [24] R. Curwen and A. Blake. Dynamic contours: Real-time active splines. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 2, pages 39–57. MIT Press, 1991.
- [25] H. Delingette. Simplex Meshes: a general representation for 3D shape reconstruction. Technical Report 2214, INRIA, 1994.
- [26] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B39:1–38, 1977.
- [27] B. Dorner. Hand shape identification and tracking for sign language interpretation. Looking at People Workshop, Chambéry, France, 1993.
- [28] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics*. Addison Wesley, 1990.
- [29] W.T. Freeman, K. Tanaka, J. Ohta, and K. Kyumu. Computer vision for computer games. In *Proc. 2nd International Face and Gesture Recognition Conference*, Killington, Vermont, 1996.
- [30] U. Grenander, Y. Chow, and D. M. Keenan. *Hands. A Pattern Theoretic Study of Biological Shapes*. Springer-Verlag, 1991.
- [31] R.I. Hartley. Euclidean reconstruction from uncalibrated views. In Joseph L. Mundy, Andrew Zisserman, and David Forsyth, editors, *Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes in Computer Science*, pages 239–256. Springer-Verlag, Second Joint European-US Workshop, Ponta Delgada, Azores, Portugal, October 1993.

- [32] A.J. Heap. Real-time hand tracking and gesture recognition using Smart Snakes. In *Proc. Interface to Human and Virtual Worlds*, Montpellier, France, June 1995. Also available as <ftp://ftp.cam-orl.co.uk/pub/docs/ORL/1995/tr.95.1.ps.Z>.
- [33] A. Hill, T.F. Cootes, and C.J. Taylor. A generic system for image interpretation using flexible templates. In *Proc. BMVC*, pages 276–285, Leeds, UK, 1992. Springer-Verlag.
- [34] A. Hill and C.J. Taylor. Model-based image interpretation using genetic algorithms. In *Proc. BMVC*, pages 266–274. Springer-Verlag, 1991.
- [35] A. Hill and C.J. Taylor. Automatic landmark generation for point distribution models. In *Proc. BMVC*, pages 429–438, York, UK, 1994. BMVA Press.
- [36] A. Hill and C.J. Taylor. Active shape models and the shape approximation problem. In *Proc. BMVC*, pages 157–166, Birmingham, UK, 1995. BMVA Press.
- [37] A. Hill and C.J. Taylor. A method for non-rigid correspondence for automatic landmark identification. In *Proc. BMVC*, pages 323–332, Edinburgh, UK, 1996. BMVA Press.
- [38] A. Hill, A. Thornham, and C.J. Taylor. Model-based interpretation of 3D medical images. In *Proc. BMVC*, pages 339–348, Guildford, UK, 1993. BMVA Press.
- [39] D. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.
- [40] D.C. Hogg. *Machine Vision*, volume 14 of *Handbook of Perception and Cognition*, chapter 7, pages 183–227. Academic Press, 1996.
- [41] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.
- [42] B.K.P. Horn. Closed-form solution of absolute orientation using quaternions. *Journal of the Optical Society of America*, 4(4):629–642, April 1987.
- [43] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In B Buxton and R Cipolla, editors, *Proc. ECCV '96*, volume I, pages 343–356, 1996.

- 
- [44] I.T. Joliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
- [45] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. In *Proc. ICCV*, pages 259–268, London, England, 1987.
- [46] C. Kervrann and H. Heitz. Robust tracking of stochastic deformable models in long image sequences. In *Proceedings of the IEEE International Conference on Image Processing*, volume III, pages 88–92, Austin, Texas, US, 1994. IEEE Computer Society Press.
- [47] R. Kjeldsen and J. Kender. Knowledge based hand gesture recognition. In *IEEE Workshop on Context-Based Vision, ICCV*, Cambridge, MA, 1995.
- [48] A. Lanitis, C.J. Taylor, and T.F. Cootes. A generic system for classifying variable objects using flexible template matching. In *Proc. BMVC*, pages 329–338, Guildford, UK, 1993. BMVA Press.
- [49] A. Lanitis, C.J. Taylor, and T.F. Cootes. An automatic face identification system using flexible appearance models. In *Proc. BMVC*, pages 65–74, York, UK, 1994. BMVA Press.
- [50] J. Lee and T.L. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, pages 77–86, September 1995.
- [51] W.E. Lorensen and H.E. Cline. Marching cubes: a high resolutions 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [52] D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(13):441–450, 1991.
- [53] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, July 1997.
- [54] P. Nesi and A. Del Bimbo. A vision-based 3-D mouse. *Int. J. Human-Computer Studies*, 1996.
- [55] S. Omohundro. Bumptrees for efficient function, constraint, and classification learning. In Touretzky Lippmanm, Moody, editor, *Advances in neural information processing systems 3*, pages 693–699, San Mateo, CA., 1991.

- 
- [56] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, July 1997.
- [57] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):730–742, 1991.
- [58] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modelling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.
- [59] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, and B. Face. *Numerical Recipes in C*. Cambridge University Press, 1992.
- [60] F.K.H. Quek and M. Zhao. Inductive learning in hand pose recognition. In *Proc. 2nd International Face and Gesture Recognition Conference*, Killington, Vermont, 1996.
- [61] L.R. Rabiner and B.H. Juang. An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 1986.
- [62] J.M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Proc. 3rd ECCV*, volume II, pages 35–45, Stockholm, Sweden, 1994. Springer-Verlag.
- [63] J.M. Rehg and T. Kanade. Visual tracking of self-occluding articulated objects. In *Proc. ICCV*, Boston, MA., 1995.
- [64] L.G. Roberts. Machine perception of three-dimensional solids. In J. Tippet, editor, *Optical and Electro-optical Information Processing*, pages 159–197. MIT Press, 1965.
- [65] Z. Schuss. *Theory and Applications of Stochastic Differential Equations*. John Wiley & Sons, 1980.
- [66] A. Sclaroff and A. Pentland. Physically-based combinations of views: Representing rigid and nonrigid motion. In IEEE Computer Society Press, editor, *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 158–164, November 1994. Also available as [ftp://agora.leeds.ac.uk/scs/doc/reports/1994/94\\_11.ps.Z](ftp://agora.leeds.ac.uk/scs/doc/reports/1994/94_11.ps.Z).

- 
- [67] X. Shen and D.C. Hogg. Generic 3D shape model: Acquisitions and applications. In *Proc. CAIP*, Prague, Czech Republic, September 1995.
- [68] P.D. Sozou, T.F. Cootes, C.J. Taylor, and E.C. Di-Mauro. A non-linear generalisation of PDMs using polynomial regression. In *Proc. BMVC*, volume II, pages 397–406, York, UK, 1994. BMVA Press.
- [69] P.D. Sozou, T.F. Cootes, C.J. Taylor, and E.C. Di-Mauro. Non-linear point distribution modelling using a multi-layer perceptron. In *Proc. BMVC*, volume I, pages 107–116, Birmingham, UK, 1995. BMVA Press.
- [70] N. Sumpter, R.D. Boyle, and R.D. Tillet. Modelling collective animal behaviour using extended Point Distribution Models. In *Proc. BMVC*, Colchester, UK, 1997. BMVA Press.
- [71] M. Syn and R. Prager. A model based approach to 3D freehand ultrasound imaging. In Y. Bizais, C. Barillot, and R. Di Paola, editors, *Information Processing in Medical Imaging*, Computational Imaging and Vision, pages 361–362. Kluwer Academic, 1995.
- [72] D. Terzopoulos and R. Szeliski. Tracking with Kalman Snakes. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 1, pages 3–20. MIT Press, 1991.
- [73] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.
- [74] A.D. Worrall, J.D. Ferryman, G.D. Sullivan, and K.D. Baker. Pose and structure recovery using active models. In *Proc. BMVC*, Birmingham, UK, September 1995.
- [75] A. Yuille and P. Hallinan. Deformable templates. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 2, pages 21–38. MIT Press, 1991.