Probabilistic Approaches to Matching and Modeling Shapes

Graham McNeill

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2008

Abstract

Comparing contours is an important problem in content-based image retrieval and object recognition. While humans can easily assess the similarity of two contours, a generic algorithm for approximating this 'perceptual distance' has proved elusive. We argue that a multiscale approach is required for this task and introduce a conceptually simple yet highly effective segment-based algorithm (Hierarchical Procrustes Matching (HPM)) which outperforms existing techniques on benchmark retrieval tests. Whereas HPM is designed to match complete contours, many real-world applications of shape matching involve partial occlusion and clutter. Probabilistic approaches to shape matching are particularly well-equipped to handle these problems, yet previous work in this area has focused on matching *unordered point sets* rather than contours. After reviewing the basic ideas behind probabilistic matching, we show how it can be extended to handle both contours and *part-based shapes*. In the final chapters of the thesis, we move from pairwise matching to the more complex problem of modeling shape classes. We present *linear and nonlinear generative probabilistic models* for both contours and unordered point sets and apply these to a variety of data sets. A particularly interesting finding is that the nonlinear contour model performs very well on a benchmark correspondence test relative to state-of-the-art alternatives.

Acknowledgements

I would like to sincerely thank my supervisor Sethu Vijayakumar for the invaluable guidance and support he provided throughout the project. Thanks are also due to Chris Williams for feedback on various aspects of my work, to John Collomosse for allowing me time to write up whilst working in Bath and to Narayanan Edakunni, Tim Hospedales, Giogos Petkos and Marc Toussaint for the many interesting and enjoyable discussions. Finally, I wish to thank Emma for always believing in me.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Graham McNeill*)

Table of Contents

Chapter 1

Introduction

Why Study Shape?

Quantitative shape analysis is fundamental to many important applications spanning a large variety of domains. Some illustrative examples include:

1. Image retrieval, *e.g.* retrieve images from a database which are *most similar* to a given query image.
2. Computer Vision, *e.g.* locate the tennis racquet in a given photograph.
3. Biomedical Image Registration and Analysis, *e.g.* compare normal corpus collosi to those with a specific pathology.
4. Chemoinformatics, *e.g.* understand the relationship between shape and observed chemical behavior for a group of related molecules.
5. Biology, *e.g.* investigate the relationship between skull morphology and gender in gorillas.

Let us define a *shape* as a set of 2D or 3D points (the terms *shape* and *point set* are used synonymously throughout this thesis). Even under this simplified definition, the above examples involve different types of shape analysis which we now consider.

Labeled or Unlabeled?

In example 5, a trained biologist might manually label corresponding points of each skull (*e.g.* the tip of each nose, the bottom of the chin,...). However, such complete,

precise manual labeling is either impossible or impractical in examples 1 to 4: image databases can be huge, the pointwise correspondence between the smooth contours of corpus collosi can only be approximated by visual inspection and molecules often have complex shapes and can be composed of thousands of atoms. Thus, examples 1-4 differ from example 5 in that they take *unlabeled point sets* as input and the *correspondence problem* must be solved as part of the analysis. In this thesis, we only consider problems of this type and note that analysis of *labeled point sets* is already well-understood – see Dryden and Mardia (1998) for a comprehensive introduction.

Ordered or Unordered?

If the shapes under consideration represent object boundaries, then the ordering of the points is meaningful and we refer to such shapes as *ordered point sets*. In contrast, there is no intuitive ordering for the atoms of a molecule and we refer to such shapes as *unordered points sets*. Both ordered and unordered point sets are considered in this thesis.

Pairwise Comparison or Group Analysis?

Pairwise comparison of shapes is key to many applications. In retrieval and matching problems (examples 1 and 2), we are typically only interested in the 'distance' between two shapes; in chemoinformatics, we might wish to compare just two molecules. In contrast, examples 3 and 4 require an analysis of within-group and between-group variation – a more demanding task. Both pairwise comparison and group analysis are addressed in this thesis.

1.1 Thesis Outline

Pairwise matching problems are considered in Chapters 3-5; Chapters 6 and 7 focus on learning class models from multiple point sets. Discussion of related work and results of the proposed algorithms are presented in the relevant chapter to keep each chapter as self-contained as possible. The following is a brief outline of the thesis together with details of any publications containing the material covered in the relevant chapter:

Chapter 2 reviews the basics of Procrustes matching which is used extensively throughout the thesis.

Chapter 3 introduces a fast, simple method for contour matching and retrieval.

- *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005 [McNeill & Vijayakumar, 2005].

Chapter 4 extends the approach in Chapter 3 to a hierarchical technique which leads to improved retrieval performance.

- *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006 [McNeill & Vijayakumar, 2006a].

Chapter 5 describes probabilistic approaches to pairwise matching for both ordered and unordered shapes.

- *International Conference on Pattern Recognition (ICPR)*, 2006 [McNeill & Vijayakumar, 2006b].
- *International Conference on Image Processing (ICIP)*, 2006 [McNeill & Vijayakumar, 2006d].
- *Advances in Neural Information Processing Systems (NIPS)*, 2006 [McNeill & Vijayakumar, 2006c].

Chapter 6 shows how to learn a generative probabilistic class model for unordered point sets.

- submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

Chapter 7 addresses the same problem as Chapter 6 but for ordered point sets.

- *International Conference on Machine Learning (ICML)*, 2007 [McNeill & Vijayakumar, 2007].

Chapter 8 discusses the main contributions of the thesis and plans for future work.

Chapter 2

Procrustes Matching

Procrustes analysis forms the basis of modern statistical shape analysis of *labeled* point sets.[1] Here, we review some basic results from Procrustes analysis that will be used to analyze *unlabeled* point sets. With the exception of Section 2.4, the material in this chapter is adapted from Dryden and Mardia (1998) where derivations for the stated results can be found. The results in Section 2.4 are straightforward extensions of those in Section 2.2.

2.1 Introduction

Consider the shapes in Figs. 2.1a and 2.1b.[2] It is obvious that these shapes are very *similar* despite the difference in rotation, scale and position on the page. More generally, we can say that rotation, scale and translation (the *similarity transformations*) have little or no effect on the *perceptual distance* between any two shapes. Given this observation, it seems sensible that quantitative techniques for analyzing shapes should be invariant to the similarity transformations; this is precisely the idea behind Procrustes analysis.

[1] The term Procrustes comes from the Greek myth concerning the villain Damastes. Briefly, Damastes would offer travelers hospitality for the night and then stretch them on a rack or saw off their extremities to make them fit the bed. This practice earned Damastes the nickname Procrustes which translates as "stretcher".

[2] Many of the figures in this thesis, including Fig. 2.1, contain shapes from the MPEG-7 shape data set discussed in Section 3.3.

(a) (b) (c)

Figure 2.1: Two similar shapes for which the correct correspondence is known (a,b) aligned using Procrustes matching (c).

2.2 Procrustes Matching in Arbitrary Dimensions

Assume that we have two point sets of equal size represented by the matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{J \times M}$, where J is the number of points and M is the number of dimensions – so each row contains the coordinates of a single point. For now, we assume that the correspondence between the two shapes is known and that the j-th row of $\mathbf{X}$ corresponds to the j-th row of $\mathbf{Y}$.

To match $\mathbf{Y}$ to $\mathbf{X}$, we rotate, scale and translate $\mathbf{Y}$ in order to minimize the sum of squared distances between corresponding pairs of points. Formally, we minimize

$$D^2(\mathbf{X}, \mathbf{Y}) \equiv \|\mathbf{X} - \beta \mathbf{Y} \Gamma - \mathbf{1}_J \nu^{\mathrm{T}}\|^2, \tag{2.1}$$

where $\|\mathbf{X}\| = \mathrm{Tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{1/2}$ is the Euclidean norm, β is a scale parameter, Γ is an $M \times M$ rotation matrix, $\nu \in \mathbb{R}^M$ is a translation vector and $\mathbf{1}_J$ is the $J \times 1$ vector of ones. Without loss of generality, we can assume that both shapes have been **centered**:

$$\mathbf{X} \leftarrow \left(\mathbf{I}_J - \frac{1}{J}\mathbf{1}_J\mathbf{1}_J^{\mathrm{T}}\right)\mathbf{X}, \qquad \mathbf{Y} \leftarrow \left(\mathbf{I}_J - \frac{1}{J}\mathbf{1}_J\mathbf{1}_J^{\mathrm{T}}\right)\mathbf{Y}. \tag{2.2}$$

The matrix $\mathbf{I}_J - \frac{1}{J}\mathbf{1}_J\mathbf{1}_J^{\mathrm{T}}$ is called the **centering matrix**. For centered shapes, it can be shown that the transformations which minimize eq.(2.1) are given by

$$\hat{\Gamma} = \mathbf{U}\mathbf{V}^{\mathrm{T}}, \qquad \hat{\beta} = \frac{\mathrm{Tr}(\mathbf{X}^{\mathrm{T}}\mathbf{Y}\hat{\Gamma})}{\mathrm{Tr}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y})}, \qquad \hat{\nu} = \mathbf{0}, \tag{2.3}$$

where the matrices $\mathbf{U}$ and $\mathbf{V}$ come from the singular value decomposition (SVD)

$$\frac{\mathbf{X}^{\mathrm{T}}\mathbf{Y}}{\|\mathbf{X}\|\,\|\mathbf{Y}\|} = \mathbf{V}\Lambda\mathbf{U}^{\mathrm{T}}. \tag{2.4}$$

Fig. 2.1c shows the shape in Fig. 2.1b matched to the shape in Fig. 2.1a using this method.

2.3 Planar Procrustes Matching

For 2D point sets (*e.g.* Fig. 2.1), there is an alternative formulation of the matching problem that leads to simple expressions for the optimal transformations and does not require SVD. Specifically, we treat each 2D point as a point in the complex plane and so replace the coordinate matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{J\times 2}$ with complex vectors $\mathbf{u}, \mathbf{w} \in \mathbb{C}^J$. The minimization problem in eq.(2.1) becomes

$$D^2(\mathbf{u},\mathbf{w}) \equiv \|\mathbf{u} - \beta e^{i\theta}\mathbf{w} - (a+ib)\mathbf{1}_J\|^2, \tag{2.5}$$

where $\|\mathbf{u}\|^2 \equiv \mathbf{u}^*\mathbf{u}$ is the l^2-norm ('*' denotes the complex conjugate), $\beta \in \mathbb{R}$ is the scale parameter, $\theta \in \mathbb{R}$ is the angle of rotation and $a+ib \in \mathbb{C}$ is the translation. Assuming that the vectors $\mathbf{u}$ and $\mathbf{w}$ are centered to have mean equal to zero, the values of θ, β, a and b that minimize eq.(2.5) are given by

$$\hat{\theta} = \angle(\mathbf{w}^*\mathbf{u}), \qquad \hat{\beta} = \frac{\|\mathbf{w}^*\mathbf{u}\|}{(\mathbf{w}^*\mathbf{w})}, \qquad \hat{a} = \hat{b} = 0, \tag{2.6}$$

where $\angle(\cdot)$ denotes the complex argument.

Once matched, the sum of squared distances between corresponding points is given by

$$d^2_{\mathrm{SSD}}(\mathbf{u},\mathbf{w}) = \mathbf{u}^*\mathbf{u} - \frac{\|\mathbf{w}^*\mathbf{u}\|^2}{(\mathbf{w}^*\mathbf{w})}. \tag{2.7}$$

In some cases, it is desirable to normalize for the number of points and work with the average squared distances between corresponding points. We define the **Procrustes mean squared error (PMSE)** to be

$$d^2_{\mathrm{PMSE}}(\mathbf{u},\mathbf{w}) = \frac{1}{J}\left(\mathbf{u}^*\mathbf{u} - \frac{\|\mathbf{w}^*\mathbf{u}\|^2}{(\mathbf{w}^*\mathbf{w})}\right). \tag{2.8}$$

Note that d^2_{SSD} and d^2_{PMSE} are not symmetric in their arguments and so are not strictly distance metrics. This asymmetry is easily removed by scaling both point sets to unit size prior to matching. Substituting $\mathbf{u} \leftarrow \mathbf{u}/\|\mathbf{u}\|$ and $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|$ into eq.(2.7) gives the expression for the **full Procrustes distance (FPD)**:[3]

$$d_{\mathrm{FPD}}(\mathbf{u},\mathbf{w}) = \left(1 - \frac{\|\mathbf{w}^*\mathbf{u}\|^2}{\mathbf{u}^*\mathbf{u}\mathbf{w}^*\mathbf{w}}\right)^{1/2}. \tag{2.9}$$

[3]The substitutions incorporate the scale normalization into the computation of the FPD and hence, the *unscaled* vectors $\mathbf{u}$ and $\mathbf{w}$ should be used to evaluate eq.(2.9).

The word 'full' indicates that we have optimized over the full set of similarity transformations: rotation, scale and translation. There are many useful variants of this approach (see Dryden and Mardia (1998)); for example, when dealing with molecules (Chapter 6), one typically does not allow any scaling. Finally, it is worth noting that d_{FPD} always lies in the interval [0,1].

2.4 Weighted Procrustes Matching

In the previous sections, it was assumed that the pointwise correspondence between the two shapes is known. If the correspondence is not known, one possible approach to shape matching is to use an alternating two step algorithm:

1. Given a geometric *alignment* of the shapes, find the optimal *correspondence*.
2. Given a *correspondence* between the shapes, *align* them using Procrustes matching.

This is essentially the approach taken in Chapter 5, but importantly, *soft correspondences* are used whereby a weight between zero and one indicates the strength of correspondence between a pair of points. In this case, we are faced with a *weighted Procrustes problem* in step 2 and must replace the Euclidean distance in eq.(2.1) with

$$D^2_{\mathrm{wtd}}(\mathbf{X},\mathbf{Y}) \equiv \|\mathbf{X} - \beta\mathbf{Y}\Gamma - \mathbf{1}_J\nu^{\mathrm{T}}\|^2_{\mathrm{wtd}}, \tag{2.10}$$

where

$$\|\mathbf{X}\|_{\mathrm{wtd}} \equiv \mathrm{Tr}(\mathbf{X}^{\mathrm{T}}\Sigma\mathbf{X})^{1/2}, \quad \text{with } \Sigma \in \mathbb{R}^{J\times J} \text{ a diagonal matrix with positive entries.} \tag{2.11}$$

If we use the modified centering procedure

$$\mathbf{X} \leftarrow \left(\Sigma^{1/2} - \frac{1}{\mathrm{Tr}(\Sigma)}\mathbf{1}_J\mathbf{1}_J^{\mathrm{T}}\Sigma^{3/2}\right)\mathbf{X}, \qquad \mathbf{Y} \leftarrow \left(\Sigma^{1/2} - \frac{1}{\mathrm{Tr}(\Sigma)}\mathbf{1}_J\mathbf{1}_J^{\mathrm{T}}\Sigma^{3/2}\right)\mathbf{Y}, \tag{2.12}$$

then the optimal transformations are given by eqs.(2.3) and (2.4).

Chapter 3

Cyclic Procrustes Matching (CPM)

In this chapter, we introduce a fast, simple matching algorithm for comparing closed contours and show that it performs well on a benchmark retrieval test.[1]

3.1 Shape Representation

We represent a contour by a reasonably small number of contour points (*e.g.* $J = 50$ or 100) which facilitates efficient processing and matching. In particular, this ensures that the number of possible correspondences between any two shapes is small and that the cost of evaluating any specific correspondence is low. There are two ways in which the contour points might be chosen:

Feature-based for example, choosing points based on maximal curvature [Super, 2004a] or distance from the centroid [Zhang et al., 2003].

Equal-spacing as used in the curve matching algorithm of Sebastian et al. (2003) and the 'shape contexts' work of Belongie et al. (2002).

Note that feature-based approaches implicitly assume that some contour points are more important than others, whereas the equal-spacing representation is simply an approximation to the true contour and essentially assigns equal importance to all parts of the contour. Also, a feature-based approach may be well-suited to a specific type of shape but poorly suited to a different type. For example, points of maximal curvature are appropriate when comparing aeroplane contours [Zhang et al., 2003], but would not be

[1] The material contained in this chapter is also described in [McNeill & Vijayakumar, 2005].

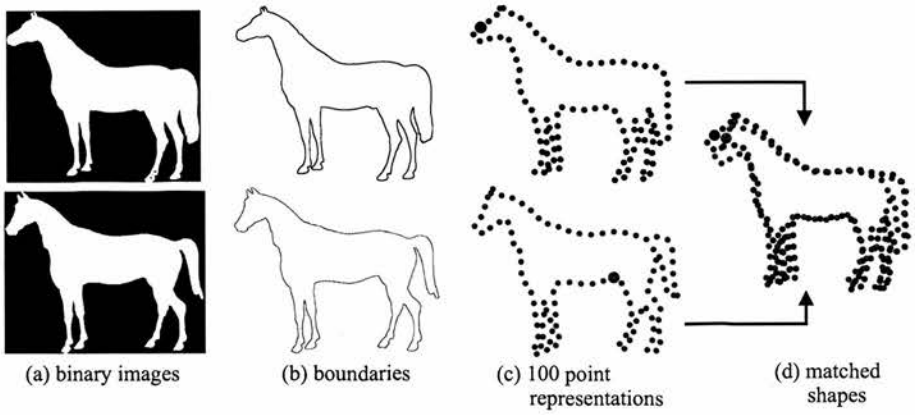

Figure 3.1: (a,b,c) Representing shape boundaries by point sets; the highlighted points in (c) indicate the arbitrary start points on each shape. (d) The optimal correspondence and alignment using Cyclic Procrustes Matching (CPM).

suitable for comparing diatoms (Fig. 4.7). Since we are interested in generic shape matching/retrieval from a potentially diverse database, we use the equal-spacing representation.

3.2 The Correspondence Problem

Fig. 3.1 shows two binary images, the contour extracted from each image and the equal-spacing representation of each contour. In Fig. 3.1c, both shapes are represented by $J = 100$ points but the starting position on each contour is arbitrary, *i.e.* the correspondence is unknown. Note that if the correct correspondence *was* known, we could assess the similarity of the two shapes using the full Procrustes distance (FPD) defined in eq.(2.9). Thus, a possible way to handle the unknown correspondence is to compute the FPD for *all* possible correspondences and define the shape distance as the minimum of these values. This brute force evaluation strategy is acceptable here because the number of evaluations grows linearly with the number of points due to the ordered nature of the point sets – see below. Furthermore, the Procrustes distance itself is very cheap to compute.

To formalize the above idea, we use the complex notation of Section 2.3 and consider two shapes $\mathbf{u}, \mathbf{w} \in \mathbb{C}^J$. For both shapes, the points are indexed in the order that they

appear on the boundary, counting clockwise from some arbitrary starting point. We assume that the 'correct' correspondence between the two shapes can be recovered by re-indexing the points of $\mathbf{w}$ using

$$\mathbf{w}_j \leftarrow \mathbf{w}_{(j+\rho)|_J}, \quad \rho \in \{0,1,2,\ldots,J-1\}, \tag{3.1}$$

where $\rho|_J$ denotes 'ρ mod J'. Forcing all the indices to change by the same shift parameter (ρ) in this way ensures that the cyclic ordering of the points is maintained. This constraint on the re-indexing also makes the implicit assumption that corresponding features are located at the same distance along the perimeter of each shape (*cf.* Chapters 4 and 7). Given the simple form of eq.(3.1), there are only J potential correspondences – the indices of $\mathbf{u}$ remain fixed and we run over the J possible values of ρ. The **cyclic Procrustes distance (CPD)** is defined as

$$d_{\mathrm{CPD}}(\mathbf{u},\mathbf{w}) \equiv \min_{\rho\in\{0,1,2,\ldots,J-1\}} d_{\mathrm{FPD}}(\mathbf{u},\mathbf{w}_{\rightarrow\rho}), \tag{3.2}$$

where $\mathbf{w}_{\rightarrow\rho}$ denotes the vector $\mathbf{w}$ with its entries cyclically shifted by ρ positions. The CPD can be made invariant to **reflection** by comparing $\mathbf{u}$ to both $\mathbf{w}$ and its complex conjugate, $\mathbf{w}^*$:

$$d'_{\mathrm{CPD}}(\mathbf{u},\mathbf{w}) \equiv \min\{d_{\mathrm{CPD}}(\mathbf{u},\mathbf{w}), d_{\mathrm{CPD}}(\mathbf{u},\mathbf{w}^*)\}. \tag{3.3}$$

We refer to the use of d'_{CPD} to match and correspond shapes as **cyclic Procrustes Matching (CPM)**. The match associated with CPM can be visualized by applying the relevant transformation (eq.(2.6)). A simple example of this is shown in Fig. 3.1; the initial correspondence is arbitrary (3.1c) but the correspondence associated with the CPM is clearly sensible (3.1d). The example in Fig. 3.2 emphasizes the invariance of CPM to the similarity transformations. We now consider the performance of CPM on a benchmark retrieval test.

3.3 Results

In this section, we demonstrate the effectiveness of CPM on the MPEG-7 shape data set[2]; see Section 4.3 for the performance of CPM on other data sets.

The "Bullseye Test" on the MPEG-7 shapes has been used extensively to assess the performance of shape retrieval algorithms. The data set is composed of 1400 binary

[2]http://www.cis.temple.edu/~latecki/research.html#shape

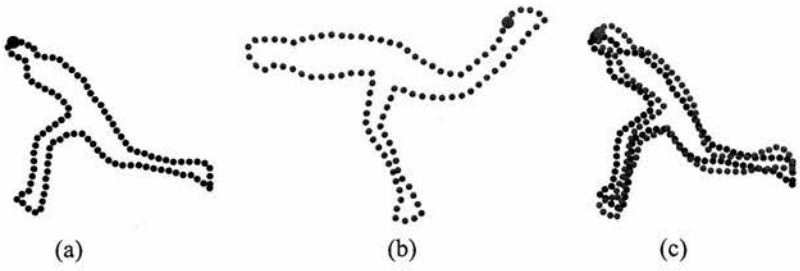

Figure 3.2: Using CPM to correspond and align shapes.

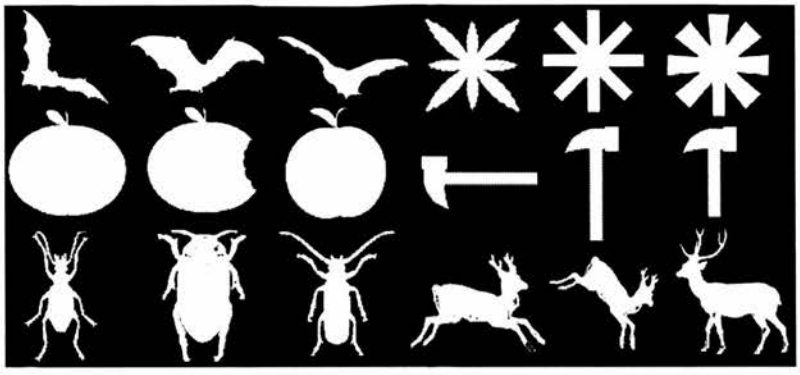

Figure 3.3: Example shapes from the MPEG-7 data set.

images with a single shape in each image. There are 70 different classes and 20 observations in each class. Some of the shapes are shown in Figure 3.3. In the bullseye test, a shape is presented as a query and the top 40 matches are retrieved (from the entire data set – the test shape is not removed). The task is repeated for each shape and the number of correct matches (out of a maximum possible 20) are noted. A perfect performance results in $1400 \times 20 = 28000$ matches. Results are given as a percentage of this maximum score.

Table 3.1 shows, to the best of our knowledge, the best published bullseye scores. The computation time required to compute a single pairwise score is also listed if available. Note that the different algorithms will have been implemented in different programming languages and ran on different computers.

Table 3.2 gives the bullseye scores achieved by CPM for various values of J – the number of points used to represent the shape. The main observation is that CPM is reasonably accurate even for small J where the computation time becomes particularly

Table 3.1: Bullseye scores for the MPEG-7 shape data set – best published results only. Times given refer to a single pairwise comparison.

Algorithm	Score (%)	Time (ms)
Internal Distances [Ling & Jacobs, 2005]	85.40	310
Multiscale Representation [Adamek & O'Connor, 2004]	84.93	7
Polygonal Multiresolution [Attalla & Siy, 2005]	84.33	10
Chance Probability Functions [Super, 2004b]	82.69	2.5
Optimized Curv. Scale Space [Mokhtarian & Bober, 2003]	81.12	0.18
Generative Model [Tu & Yuille, 2004]	80.03	200
RACER [Super, 2003]	79.09	0.4
Morphological Curv. Scale Spaces [Jalba et al., 2006]	78.8	N/A
Distance Sets [Grigorescu & Petkov, 2003]	78.38	700
Curve Edit Distance [Sebastian et al., 2003]	78.17	>1000
Contour-based Rep. [Adamek & O'Connor, 2003]	77.76	1
Eigenshapes [Super, 2004a]	76.92	0.6
Shape Contexts [Belongie et al., 2002]	76.51	200
Part Correspondence [Latecki et al., 2000]	76.45	50
Original Curv. Scale Space [Mokhtarian et al., 1997]	75.44	N/A

competitive. We note that, in contrast to the other papers cited here, the score published by Super (2004b) was not achieved using nearest neighbor retrieval and hence, is not directly comparable with the other results.

3.4 Discussion and Related Work

The CPM algorithm introduced in this chapter is similar to the approach of Zhang et al. (2003) who also evaluate the Procrustes distance for cyclic shifts of the point indices. However, their approach operates on a small set of high curvature points, was only tested on a little-known data set and was not compared to state-of-the-art algorithms. The main finding from this chapter is that, despite its simplicity, CPM is competitive in terms of speed and performance compared to modern shape matching algorithms. Also, CPM is

Table 3.2: Bullseye scores using the cyclic point matching algorithm (CPM).

Number of points (J)	Score (%)	Computation time per pairwise match (ms)
10	67.04	0.08
20	74.13	0.16
40	76.66	0.51
60	76.83	1.5
80	77.01	2.9
100	77.04	4.5
150	77.06	10
200	77.08	17

not subject to local minima problems (all possibilities are evaluated) and we have found that CPM typically selects a close to optimal correspondence even though the shape similarity score (eq.(3.3)) lacks the discriminative power of some of the approaches in Table 3.1. As we shall see in the next chapter, these properties make CPM a useful technique for initializing a more sophisticated matching algorithm.

Keogh et al. (2006) have applied algorithms from time series analysis such as Dynamic time warping (DTW) to shape matching. These techniques can be applied to generic time series style shape representations; for example, representing a shape as a 1D series, where each boundary point is represented by its distance from the centroid. An important property of this type of representation is that it enables pairs of boundary points (one from each shape) to be compared directly without any dependence on a global alignment. In contrast, the global nature of Procrustes-based approaches means that the correspondence and geometric alignment are coupled, so techniques such as DTW cannot be applied directly. However, as will be discussed in Section 5.5, this problem could be addressed using a hidden Markov model (HMM) with an expectation maximization (EM) algorithm [Dempster et al., 1977] which iterates between choosing the correspondence (essentially doing DTW) and updating the alignment transformations. It is possible that the impressive speed-ups introduced by Keogh et al. (2006) could also be adapted to Procrustes-based methods, but this is not investigated in this thesis.

Chapter 4

Hierarchical Procrustes Matching (HPM)

The CPM algorithm described in the previous chapter is an efficient technique for matching contiguous closed contours, but its reliance on global linear transformations is a significant limitation. This issue is addressed in this chapter by introducing a multiscale version of CPM.[1]

4.1 Motivation and Related Work

Simple, global point matching methods such as CPM and that of Zhang et al. (2003) work surprisingly well in many shape retrieval and classification tasks. These find a single linear transformation which best aligns one point set with the other. This approach fails when more complex transformations are present such as independent movement of parts or smooth deformations (Fig. 4.1). To deal with such cases, more local or non-linear approaches can be effective [Bookstein, 1996, Sebastian et al., 2003]. However, techniques which focus on local shape differences suffer from other problems. For example, the curve alignment algorithm of Sebastian et al. (2003) remains effective in the presence of smooth deformations and, to some extent, part articulation. However, the authors note that their algorithm regards a handwritten "6" and "U" as similar since the curvature of the two shapes only differs at a few points.

The above discussion indicates that even in apparently simple cases, there can be

[1] The material contained in this chapter is also described in [McNeill & Vijayakumar, 2006a].

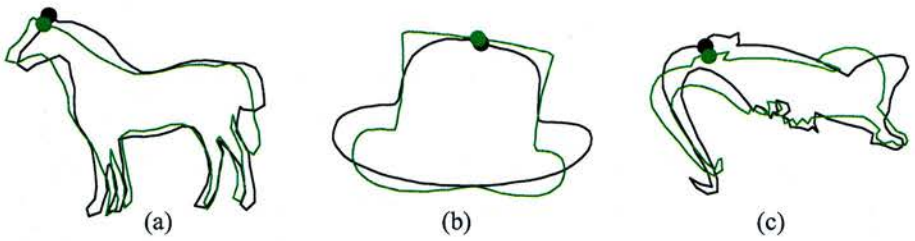

Figure 4.1: Examples of shapes matched using CPM (linear interpolation between points is used for clarity). The filled circles show corresponding points. In all cases, the correct correspondence is found, but a good alignment using only translation, rotation and scaling is not possible.

a large discrepancy between a perceived difference in shape and the approximation of this difference from a purely global or local perspective. It therefore seems natural to investigate whether a generic shape matching algorithm should lie somewhere between these two extremes – a neighborhood-based approach. More generally, one can imagine an algorithm that is not restricted to comparisons based on a single neighborhood size. The Hierarchical Procrustes Matching (HPM) algorithm presented in this chapter is a multiscale approach that investigates shape matching at a variety of different positions (segment start points) and 'scales' (segment lengths). In shape matching, 'scale' can relate to a number of related ideas: frequency (*e.g.* wavelets [Costa & Cesar, 2001]), the degree of filtering that the shape has undergone [Del Bimbo & Pala, 1999, Mokhtarian & Bober, 2003], or the resolution at which the shape is sampled [Attalla & Siy, 2005]. Here, the notion of scale arises through the matching process (rather than just the shape representation) since the least squares criteria used to assess segment similarity is dominated by the 'global' shape of the segment (see Figs. 4.3-4.5).

As noted in the previous chapter, many techniques match shapes based on specific features, such as points of zero curvature [Mokhtarian & Bober, 2003], points of minimum curvature [Del Bimbo & Pala, 1999], and convex/concave segments [Milios & Petrakis, 2000, Latecki & Lakamper, 2002]. In contrast, HPM compares segments explicitly and does not restrict itself to working with segments of a particular type. Essentially, HPM approximates a mapping which takes any segment of one shape (*i.e.* with any start point and length) to its counterpart on the other shape. Such a mapping must allow for the fact that corresponding regions of shapes often appear at slightly different

scales and positions. For example, the dorsal fin of a fish may be both larger and further forward than its counterpart on a second fish. HPM handles this problem by matching in a global to local direction. Longer segments that have already been matched provide initial matches for the shorter segments, which can then slide and stretch/contract in order to find the best matches at this smaller scale. The individual segment matches reflect the similarity of features that are relevant at the current segment length or scale. In the context of the fish example, if the backs of the fish are already corresponded, we have a good estimate as to the correct correspondence of the dorsal fins. We can then search in the neighborhood of this estimate (i.e. over segments with start points and lengths close to this estimate) to find the best match. To prevent the algorithm matching fins that are too far apart (the more candidate segments that are considered, the greater the chances of finding a spurious match), deviations from the prediction at the previous scale are penalized. A particularly simple way of implementing this penalty is via the size of the neighborhood that is searched (Section 4.2.3.1). The motivations behind HPM are illustrated using an alternative example in Fig. 4.2.[2] We now describe the algorithm in detail.

4.2 HPM: Algorithm

As in the previous chapter, we are interested in comparing two contiguous closed contours represented by the ordered point sets $\mathbf{u}, \mathbf{w} \in \mathbb{C}^J$.

4.2.1 Initialization

The first step of HPM is to find a global correspondence between the two shapes $\mathbf{u}$ and $\mathbf{w}$. If this is incorrect then any subsequent steps of HPM are meaningless, so it is important that the global correspondence algorithm is robust. The efficient CPM algorithm described in the previous chapter is well-suited to this task. As noted in Section 3.4 and demonstrated in Fig. 4.1, CPM *does* generally find the correct correspondence; the problem with CPM is that the matching score (Section 3.2) does not approximate perceptual shape similarity as well as we would wish.

[2] Procrustes matching was not used in this schematic example.

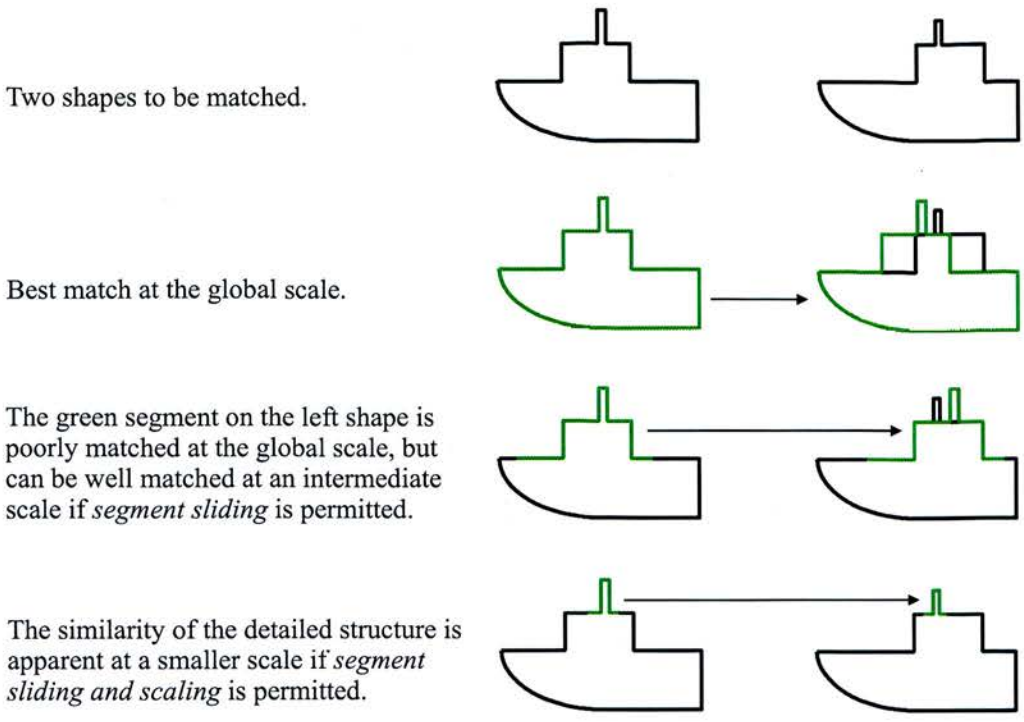

Figure 4.2: HPM computes a similarity score between two shapes by combining segment similarity scores at a variety of scales. In this example, the significant dissimilarity at the global scale will have a negative impact on the overall score, but the high degree of similarity at smaller scales will have a positive impact.

4.2.2 Segment Representation

Let us assume that CPM has been used to find a global correspondence between the shapes **u** and **w** and that the points have been relabeled accordingly so that u_1 corresponds to w_1, u_2 corresponds to w_2,... Also, assume that **u** and **w** have been scaled so that the polygons formed by their points have boundary length equal to 100. A segment of the **w** polygon is denoted by $\mathbf{w}(s_w, l_w)$, where s_w is the segment's start point and l_w is its length. A segment is represented by k equally spaced points (see below) taken from the relevant segment of the complete polygon. Note that a segment of any length and start point (*i.e.* s_w and l_w need not be natural numbers) can be represented by any number of points, k, by re-sampling from the original polygon. To represent a segment of length l, we use $k \approx \frac{l}{100}J$ points – *i.e.* the original sampling frequency of the boundary is approximately maintained.

Ideally, all possible segments of the two shapes would contribute towards the similarity score (see Section 4.4); in practice, a finite number of segments are used. There are many potential ways in which these segments might be chosen, but the following approach is simple and effective. To match $\mathbf{w}$ to $\mathbf{u}$, we consider the full shape $\mathbf{w}$, four segments of length 50, eight of length 25 and sixteen of length 12.5. At each length, neighboring segments overlap by one half. Each of these segments will, at some stage, be compared to multiple segments of $\mathbf{u}$ which have different start points and lengths. If the segment of $\mathbf{w}$ has k points, the segments of $\mathbf{u}$ that it is compared to will also be represented by k points, enabling the PMSE (eq(2.8)) between segments to be computed. Note that eq.(2.8) involves normalizing over the number of points; this makes the PMSE useful when it comes to combining segment match scores at different scales in order to compute a single match score.

The results in Section 4.3 prove that our choice of four scales and segments which overlap by one half is effective, but any number of scales or amount of overlap could be used subject to computational resource constraints.

4.2.3 Hierarchical Matching

We are now in a position to describe the HPM algorithm which is summarized in Algorithm 1 and illustrated in Fig. 4.3. Two variants of HPM are considered. The first prevents invalid segment matches by limiting the permitted change in start point and length from that predicted at the higher scale. The second uses a softer approach which requires dynamic programming.

4.2.3.1 Fast Method

Consider the task of matching the segment of $\mathbf{w}$ of length 50 and start point 0, *i.e.* $\mathbf{w}(0,50)$. The global correspondence 'predicts' that $\mathbf{w}(0,50)$ should be matched to $\mathbf{u}(0,50)$. As discussed in Section 4.1, the segment $\mathbf{w}(0,50)$ is allowed to stretch/contract and slide along the boundary of $\mathbf{u}$, but only to a limited extent. Formally, $\mathbf{w}(0,50)$ must be matched to a segment $\mathbf{u}(0 \pm \delta_s, 50 \pm \delta_l)$, where the maximum values of δ_s and δ_l determine the permitted slide and stretch/contraction, and the sums in the arguments are taken modulo 100. In practice, we consider a finite number of segments with start points close to 0 and lengths close to 50. Specifically, seven possible start points and seven

Algorithm 1 Pseudocode for HPM.

1: Initialize: match **u** and **w** using CPM
2: **for** segment lengths: $l = 50, 25, 12.5$ **do**
3: **for** segments of **w**: $m = 1, \ldots,$ # segments at length l **do**
4: compute the *predicted match* of segment m based on the matches at the previous length
5: define a 7×7 set of expansion options around the *predicted match*
6: compute PMSE between segment m and each neighborhood segment of **u**
7: **end for**
8: Select matches and get total score for length l based on:
i. PMSE between segments and candidate matches
ii. deviation from match predicted at length l-1
iii. deviation from match predicted by neighbors of same length
9: **end for**
10: shape similarity = weighted sum of scores at each length

possible lengths are considered, resulting in forty-nine candidate segments of **u** to which $\mathbf{w}(0, 50)$ might be matched (Alg. 1, steps 4-6). From this set of candidates, we select the segment that is closest to $\mathbf{w}(0, 50)$ as measured by the PMSE (Alg. 1, step 8 – only 8(i) is used here; 8(i)-(iii) are used for the dynamic programming approach discussed below). The same approach is used to match the other segments of length 50. The matched segments of length 50 provide initial estimates for matching the segments of length 25 and so on. The maximum permitted slide and stretch decreases with segment size. In our experiments, the segments of length 50 were allowed to stretch or contract by a maximum of 3% (with increments of 1% being evaluated) and to shift along the boundary by $\pm 1.5\%$. For segments of length 25, these values were halved, and so on.

The similarity of **u** and **w** is a weighted sum of the PMSEs over all the matched segments. As mentioned above, the PMSE is normalized for the number of points (eq.(2.8)), so the PMSEs corresponding to different length segments are of a similar magnitude (recall that the segments of length 50 have $\sim J/2$ points whereas the segments of length 25 have $\sim J/4$ points). However, there is less variation between shorter segments (Figs 4.3-4.5) so the PMSEs are generally smaller. We ensure that matches at every length

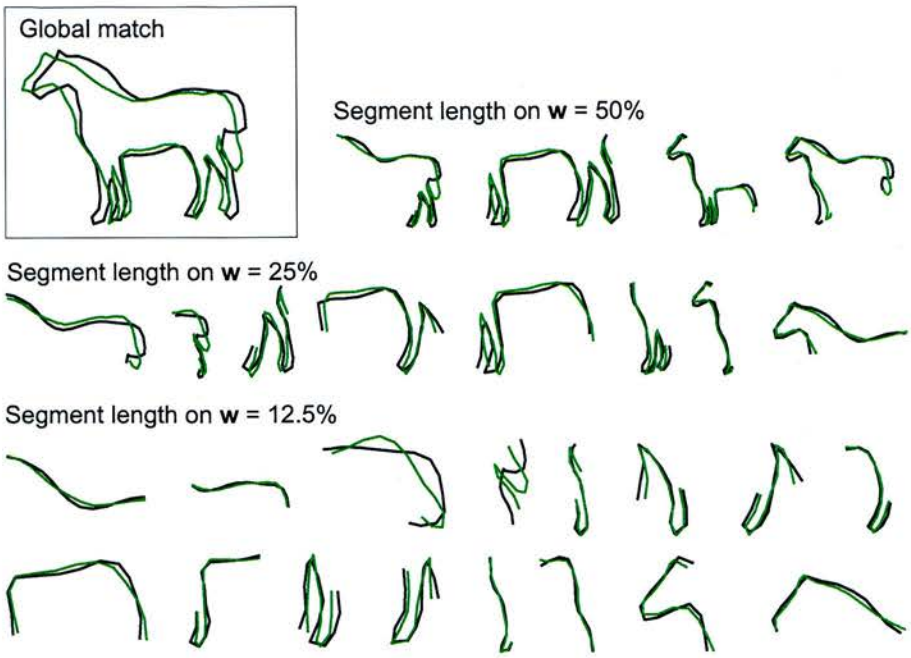

Figure 4.3: The segment matches found by HPM for the shapes in Fig. 4.1a.

make a roughly equal contribution to the final similarity score by giving a higher weight to PMSEs from shorter segments. Let S_l denote the sum of PMSEs over the matched segments at a fixed length l. The *asymmetric similarity* of $\mathbf{u}$ and $\mathbf{w}$ is defined as

$$D_{\text{asymm}}(\mathbf{u},\mathbf{w}) \equiv w_{100} d^2_{\text{PMSE}}(\mathbf{u},\mathbf{w}) + \sum_{l=50,25,12.5} w_l S_l \tag{4.1}$$

where the w_l are constant weights. The weights used for the evaluations in Section 4.3 were determined experimentally, and *the same weights were used for all data sets*. Since $D_{\text{asymm}}(\mathbf{u},\mathbf{w}) \neq D_{\text{asymm}}(\mathbf{w},\mathbf{u})$, we take the (symmetric) *similarity* of $\mathbf{u}$ and $\mathbf{w}$ to be

$$D_{\text{HPM}}(\mathbf{u},\mathbf{w}) \equiv D_{\text{asymm}}(\mathbf{u},\mathbf{w}) + D_{\text{asymm}}(\mathbf{w},\mathbf{u}). \tag{4.2}$$

Note that $D_{\text{asymm}}(\mathbf{u},\mathbf{w})$ (eq.(4.1)) depends on the position of the start point $s_u = 0$ on the polygon $\mathbf{u}$ since this determines the segments used for matching. We have not investigated whether there is any significant variation in performance associated with this choice.

4.2.3.2 Dynamic Programming Approach

This section describes a more sophisticated technique for selecting the segment-segment matches. In the "fast method" described above, the valid segment mappings were determined by a hard limit on the amount that a segment of $\mathbf{w}$ could slide and stretch away from its predicted match on $\mathbf{u}$. Here, we use a softer approach whereby the amount of slide and stretch is penalized. Consider a segment $\mathbf{w}(s_w, l_w)$, and assume that its *predicted* match on $\mathbf{u}$ (given the already matched longer segments) is $\mathbf{u}(s_u^p, l_u^p)$. If the *selected* match is $\mathbf{u}(s_u, l_u)$, then we can penalize the deviation from the predicted match using the value $|(s_u, l_u) - (s_u^p, l_u^p)|$. Rather than selecting a segment $\mathbf{u}(s_u, l_u)$ purely on the basis of PMSE, we now select it using

$$\mathbf{u}(s_u, l_u) = \arg \min_{\mathbf{u}(s_u, l_u)} \left[d^2_{\text{PMSE}}(\mathbf{u}(s_u, l_u), \mathbf{w}(s_w, l_w)) + \lambda_{l_w} |(s_u, l_u) - (s_u^p, l_u^p)| \right] \tag{4.3}$$

The parameter λ_{l_w} specifies the importance of staying close to the predictions at the current segment length, l_w. The λ_{l_w} ($w = 100, 50, 25, 12.5$) should be chosen so that the penalty terms have a similar impact at each scale. The values used for the evaluations in Section 4.3 were determined experimentally, and the same values were used for all data sets.

We may also want to encourage consistency between the matches of neighboring segments of the same length – bearing in mind that these overlap by one half. For example, if the segment $\mathbf{w}(0, 50)$ is slid far to the left of its predicted match on $\mathbf{u}$, it seems inappropriate that the neighboring segment $\mathbf{w}(25, 50)$ should slide to the right. Just as the longer segments predict a match for shorter segments, we can think of a segment $\mathbf{w}(s_w, l_w)$ predicting matches for its neighbors on either side. Specifically, if $\mathbf{w}(s_w, l)$ is matched to a segment $\mathbf{u}(s_u, l_u)$, it predicts that its neighbors of the same length, $\mathbf{w}(s_w \pm l/2, l)$, are matched to $\mathbf{u}(s_u \pm l_u/2, l_u)$.

Given a segment $\mathbf{w}(s_w, l_w)$, the segment $\mathbf{u}(s_u, l_u)$ that it is matched to will now depend on the PMSE between the segments, the penalty for choosing $\mathbf{u}(s_u, l_u)$ given the prediction from the already matched longer segments, and an additional penalty associated with the neighbors of $\mathbf{w}(s_w, l_w)$ of the same length. Using this new penalty is problematic since the neighbors of $\mathbf{w}(s_w, l_w)$ have *not* already been matched. Unlike the way that a global match provides predictions for the segments of length 50 and so on, there is no natural starting place to begin matching when considering segments of the same length – we must effectively choose all the matches simultaneously. If the

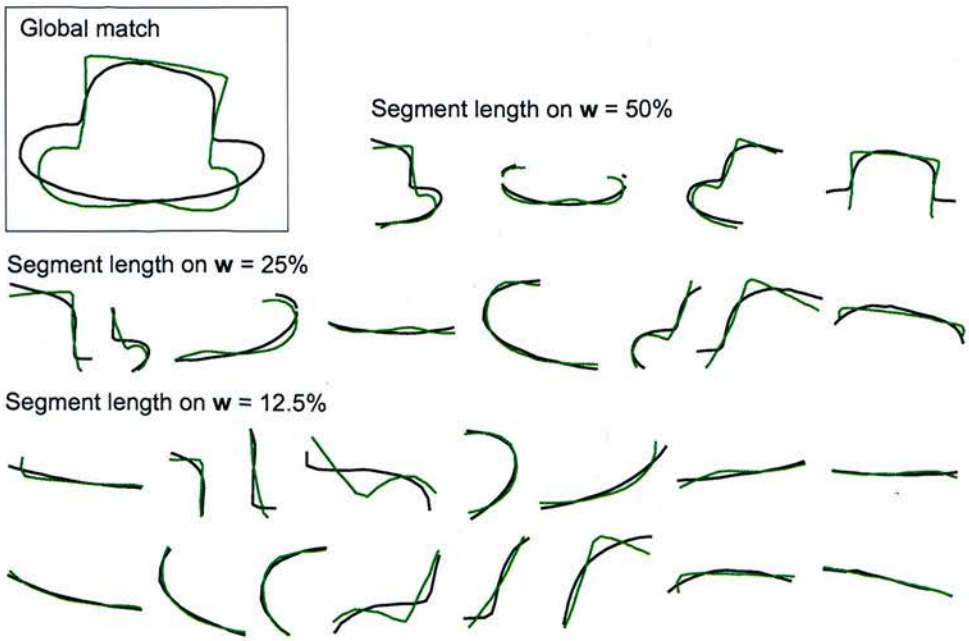

Figure 4.4: The segment matches found by HPM for the shapes in Fig. 4.1b.

M segments of $\mathbf{w}$ of a fixed length l (*e.g.* there are $M = 4$ segments of length 50) are denoted by $\mathbf{w}(s_{w_1}, l), \ldots, \mathbf{w}(s_{w_M}, l)$, then the segments of $\mathbf{u}$ that they are matched to: $\mathbf{u}(s_{u_1}, l_{u_1}), \ldots, \mathbf{u}(s_{u_M}, l_{u_M})$, are chosen so as to minimize

$$\begin{aligned} \mathcal{J} \equiv \sum_{m=1}^{M} \{ & d^2_{\mathrm{PMSE}}(\mathbf{u}(s_{u_m}, l_{u_m}), \mathbf{w}(s_{w_m}, l)) + \lambda_l |(s_{u_m}, l_{u_m}) - (s^p_{u_m}, l^p_{u_m})| \\ & + \frac{\lambda_l}{2} |(s_{u_m}, l_{u_m}) - ((s_{u_{m-1}} + l_{u_{m-1}}/2), l_{u_{m-1}})| \\ & + \frac{\lambda_l}{2} |(s_{u_m}, l_{u_m}) - ((s_{u_{m+1}} - l_{u_{m+1}}/2), l_{u_{m+1}})| \}. \end{aligned} \tag{4.4}$$

The first line of this expression is from eq.(4.3), the additional terms represent the neighbor-to-neighbor constraints and the sums m+1 and m-1 are taken modulo M. As in Section 4.2.3.1, we restrict ourselves to considering a fixed number of candidate segments (7×7=49 in our experiments) on $\mathbf{u}$ for each segment $\mathbf{w}(s_{w_m}, l)$. In this case, minimizing $\mathcal{J}$ is a cyclic shortest path problem which can be solved using Dijkstra's algorithm. The nodes of the shortest path correspond to the selected $\mathbf{u}(s_{u_1}, l_{u_1}), \ldots, \mathbf{u}(s_{u_M}, l_{u_M})$, and the cost of the shortest path is the contribution to the similarity score, S_l (eq.(4.1)), at length l. Eqs.(4.1) and (4.2) are used to compute the similarity of shapes $\mathbf{u}$ and $\mathbf{w}$.

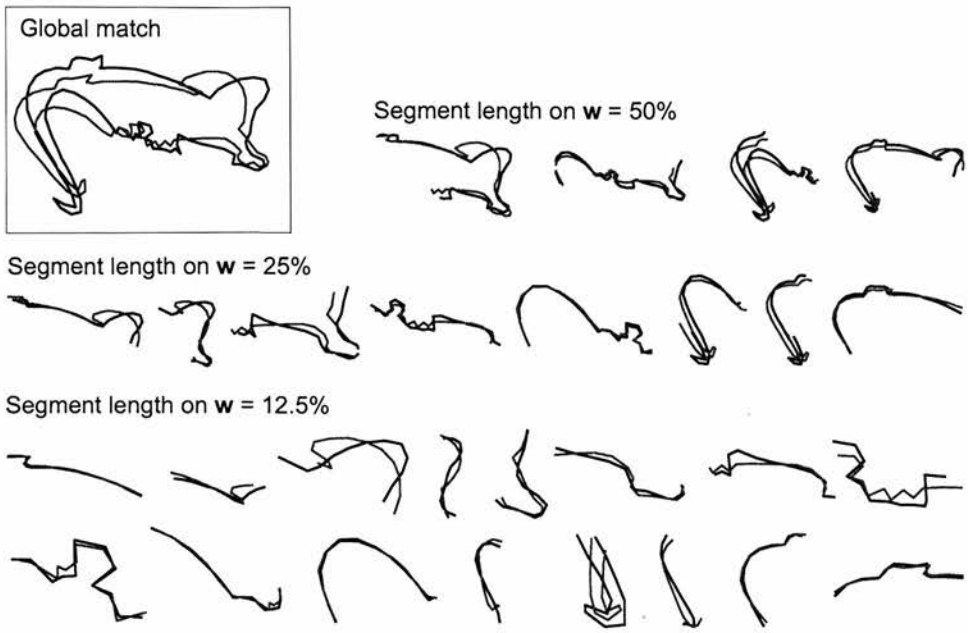

Figure 4.5: The segment matches found by HPM for the shapes in Fig. 4.1c.

4.2.4 Normalized Procrustes Distances

It is interesting to consider whether the 'confidence' with which each segment match is chosen (out of the candidate matches available) carries useful information about the quality of the overall match. To investigate this idea, we normalize the PMSEs associated with a segment of **w** and its candidate matches on **u** by making the average PMSE equal to 1, and then apply HPM as normal. This means that when a selected match is much better than the other candidate matches, it will make a very small contribution to the similarity score – recall that a small score indicates high similarity. Using this approach, the similarity of two shapes (eq.(4.2)) will be a crude measure of the average confidence with which each segment match is selected, rather than an absolute measure of shape similarity. The surprisingly good results achieved using this confidence measure (Section 4.3) highlights the strong correlation between shape similarity and the uniqueness of the optimal segment matches. We discuss this further in Section 4.5.

4.3 Experiments and Evaluations

The proposed algorithm was applied to the shapes in Fig. 4.1. The matched segments are shown in Figs. 4.3-4.5. Note that similar features are generally well matched at the scale at which they become prominent, whereas genuinely dissimilar features (*e.g.* the horses' tails, Fig. 4.3) are never well aligned.

We now consider the performance of HPM on benchmark data sets. Four variants of the algorithm are considered:

- CPM: Cyclic Procrustes matching (Chapter 3 and Section 4.2.1)
- HPM-F: fast HPM (Section 4.2.3.1)
- HPM-DP: dynamic programming HPM (Section 4.2.3.2)
- HPM-Fn: fast HPM with normalized PMSEs (Section 4.2.4)

Shapes were described by J=100 points in all cases.

4.3.1 MPEG-7 Shapes

This data set and the associated "bullseye test" are described in Section 3.3. In Table 4.1, we have added the results achieved using HPM to those listed in Table 3.1.

4.3.2 GESTURES Data

The data set consists of 980 shapes generated from 17 hand gestures. Milios and Petrakis (2000) used the 17 original shapes as queries and human relevance data as the ground truth with respect to which different algorithms can be evaluated.[3] Figure 4.6a shows the precision-recall plots[4] from Milios and Petrakis (2000) augmented with our own results – the results for HPM-DP were very similar to those for HPM-F and have been omitted for clarity. Both CPM and HPM outperform the segment-based method described by Milios and Petrakis (2000). The algorithms proposed by Belongie and Malik (2000), Sebastian

[3]GESTURES data set and relevance information for GESTURES and MARINE data available at: http://www.ced.tuc.gr/~petrakis

[4]Precision is the number of relevant shapes retrieved divided by the total number of retrieved shapes. Recall is the number of relevant shapes retrieved divided by the total number of relevant shapes in the data set.

Table 4.1: Bullseye scores for the MPEG-7 shape data set.

Algorithm	Score (%)	Time (ms)
HPM-Fn	**86.35**	**600**
Internal Distances [Ling & Jacobs, 2005]	85.40	310
Multiscale Representation [Adamek & O'Connor, 2004]	84.93	7
Polygonal Multiresolution [Attalla & Siy, 2005]	84.33	10
HPM-F	**84.07**	**600**
HPM-DP	**83.98**	**1200**
Chance Probability Functions [Super, 2004b]	82.69	2.5
Optimized Curv. Scale Space [Mokhtarian & Bober, 2003]	81.12	0.18
Generative Model [Tu & Yuille, 2004]	80.03	200
RACER [Super, 2003]	79.09	0.4
Morphological Curv. Scale Spaces [Jalba et al., 2006]	78.8	N/A
Distance Sets [Grigorescu & Petkov, 2003]	78.38	700
Curve Edit Distance [Sebastian et al., 2003]	78.17	>1000
Contour-based Rep. [Adamek & O'Connor, 2003]	77.76	1
CPM	**77.04**	**4.5**
Eigenshapes [Super, 2004a]	76.92	0.6
Shape Contexts [Belongie et al., 2002]	76.51	200
Part Correspondence [Latecki et al., 2000]	76.45	50
Original Curv. Scale Space [Mokhtarian et al., 1997]	75.44	N/A

et al. (2003), and Attalla and Siy (2005) have also been tested on this data.[5] From a visual inspection of their results, it seems that HPM outperforms the Shape Contexts approach of Belongie and Malik (2000), performs similarly to the algorithm of Attalla and Siy (2005) and performs worse than the curve alignment algorithm of Sebastian et al. (2003).

[5]We requested the relevant results from the authors but did not receive a reply.

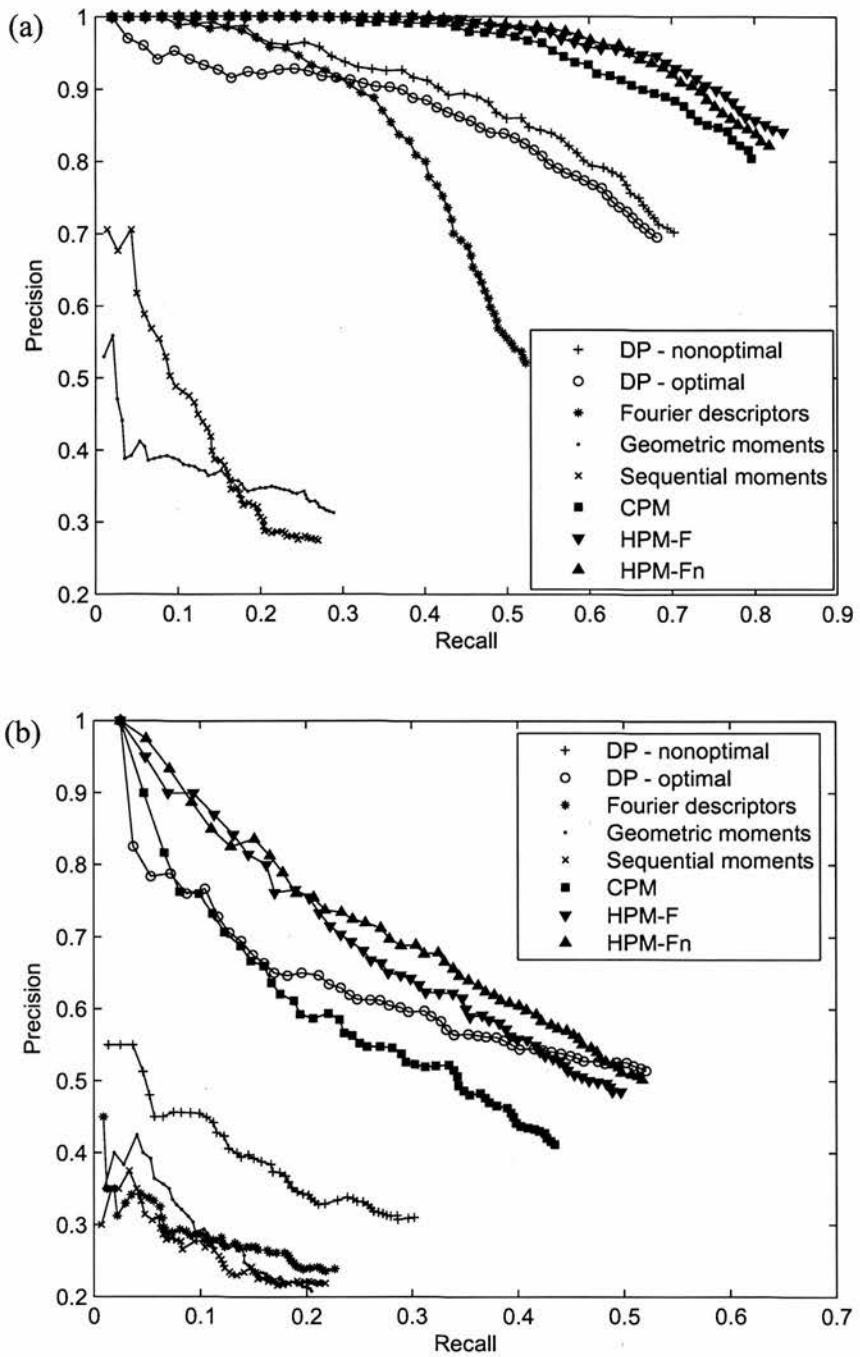

Figure 4.6: Precision-recall plots for (a) the GESTURES data, and (b) the MARINE data. Results for DP-nonoptimal, DP-optimal, Fourier descriptors, sequential moments, and geometric moments taken from Milios and Petrakis (2000).

4.3.3 MARINE Data

The data set contains the outlines of 1100 different marine species [Mokhtarian & Bober, 2003].[6] There are 20 query shapes and human relevance information is again used to evaluate performance [Milios & Petrakis, 2000]. Precision-recall plots are shown in Figure 4.6b (the results for HPM-DP are similar to those of HPM-F and are omitted). Again, HPM outperforms the technique described by Milios and Petrakis (2000) except when a large number of shapes are retrieved. Note that this time the difference in performance between CPM and HPM is more pronounced. The algorithm of Attalla and Siy (2005) has also been tested on this data set. From a visual inspection of their results, it is clear that HPM performs significantly better on this data set.

4.3.4 DIATOM Data

The data set consists of 781 outlines of individual diatoms [Jalba et al., 2006].[7] The examples are not distributed equally among the 37 species, so Jalba et al. (2006) applied a modified bullseye test whereby the number of retrieved shapes is set at twice the relevant class size. Some shapes from the data set are shown in Fig. 4.7. We achieved bullseye scores of: CPM – 63.57%, HPM-F – 74.12%, HPM-DP - 73.41% and HPM-Fn – 79.89%. HPM-Fn outperforms all but one of the techniques tested by Jalba et. al.: Curvature Scale Space (CSS) [Mokhtarian & Bober, 2003] – 66%, Fourier descriptors – 73.7%, wavelet descriptors – 74.2%, Morphological Curvature Scale Space [Jalba et al., 2006] – 82.2%. Our results for this data set are likely to have suffered due to HPM's dependence on the global correspondence. The shapes have approximate rotational and reflectional symmetry (Figure 4.7) which suggests four potential choices of global correspondence between any two shapes. Ideally, all of these should be investigated using HPM.

Looking at the results for all the data sets, it is clear that HPM, and HPM-Fn in particular, perform very well relative to existing algorithms.

[6] http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html

[7] Diatoms are single celled algae with silica shells.

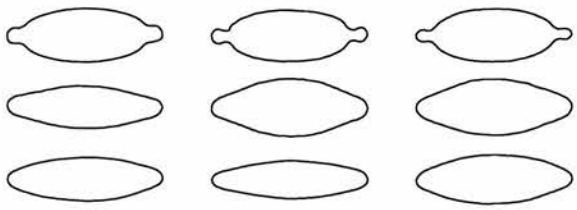

Figure 4.7: Example shapes from the DIATOM data set – shapes in the same row are in the same class.

4.4 A Continuous Segment Mapping

It is easier to qualitatively assess the performance of HPM if one can examine the match between perceptual or functional elements of shapes – rather than the segments of predefined length and start point used by the algorithm. This can be achieved by taking the matched segment pairs found by HPM as input to a regression problem which outputs a continuous mapping, *i.e.* a mapping that takes *any* segment of $\mathbf{w}$ to a segment of $\mathbf{u}$. Fig. 4.8 demonstrates this idea using thin plate spline regression.[8] Each point of the plane in Fig. 4.8a represents a segment of the contour $\mathbf{w}$ and similarly, the plane in Fig. 4.8d represents the segments of the contour $\mathbf{u}$. The continuous segment-to-segment mapping found by regression was applied to the grid points in Fig. 4.8a to produce the deformed grid in Fig. 4.8d. Figs. 4.8b,c show the segments of $\mathbf{w}$ associated with the highlighted points in Fig. 4.8a (the circle and the square). The corresponding segments of $\mathbf{u}$ are plotted in Figs. 4.8e,f and it is clear that the mapping has matched the correct segments. Note that the shorter segment of $\mathbf{w}$ (Fig. 4.8c) is a subsegment of the longer segment (Fig. 4.8b), yet the mapping shifts their start points in different directions and only changes the length of the shorter segment.

Rather than recovering a segment-segment mapping from HPM, one could formulate HPM in a continuous setting from the outset. Let us now assume that $\mathbf{w}$ and $\mathbf{u}$ are parameterized closed curves with arc length parameters $s_w, s_u \in (0, 100]$, and use the same notation $\mathbf{w}(s_w, l_w)$ to denote a segment of $\mathbf{w}$. The aim of a continuous HPM algorithm would be to find a mapping $\mathbf{f}(s_w, l_w) = (s_u, l_u)$ such that $\mathbf{w}(s_w, l_w)$ corresponds to the segment $\mathbf{u}(s_u, l_u)$ of $\mathbf{u}$. Given a suitable family of functions $\mathcal{F}$, this could be achieved by

[8] Note that the regression gives the segment mapping. We are *not* using thin plate splines to deform one shape onto the other as in Bookstein (1996).

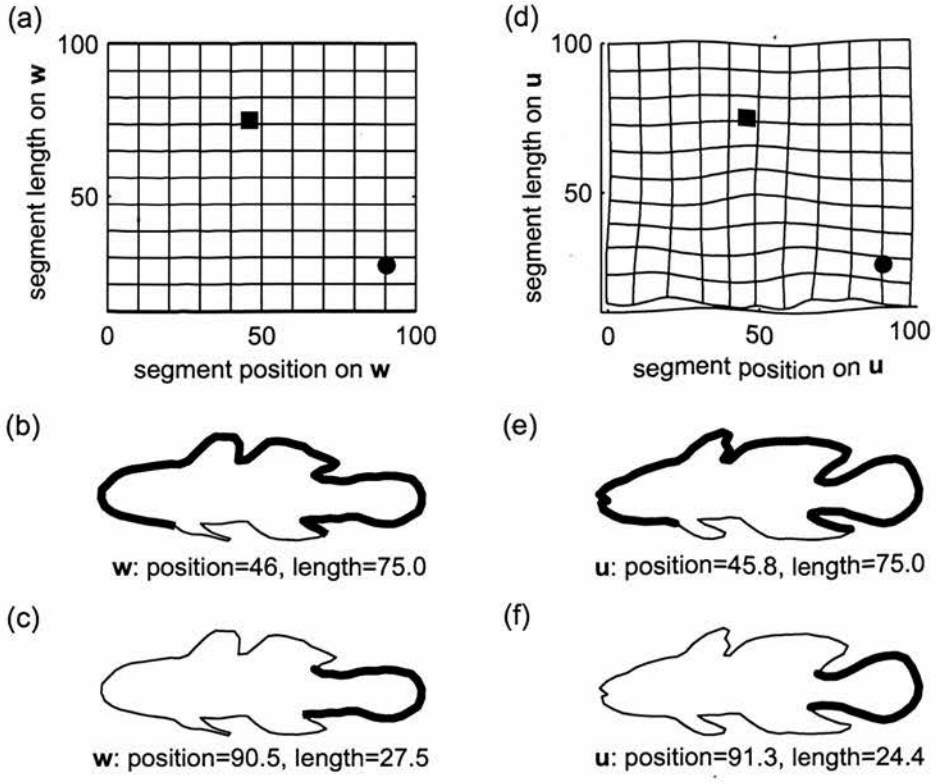

Figure 4.8: (a,d) Continuous segment mapping between two shapes. The highlighted points correspond to the segment matches shown in (b,e) – squares, and (c,f) – circles. See text for details.

selecting the $\mathbf{f} \in \mathcal{F}$ which minimizes an appropriate cost functional. For example

$$\mathbf{f} = \arg\min_{\mathbf{f}\in\mathcal{F}} \int_{l=0}^{100} \int_{s=0}^{100} \left(d(\mathbf{w}(s,l), \mathbf{u}(\mathbf{f}(s,l))) + \lambda \|\mathbf{f}''\|^2 \right) ds dl, \tag{4.5}$$

where $d(\cdot,\cdot)$ is a measure of segment similarity and the final term is a smoothness penalty on $\mathbf{f}$ – segments of $\mathbf{w}$ with similar start points and lengths should be mapped to segments of $\mathbf{u}$ with similar start points and lengths. The ideas used to construct HPM-DP in Section 4.2.3.2 were aimed at enforcing this 'smoothness' in a discrete setting. The minimum value of the functional in eq.(4.5) would indicate the similarity of $\mathbf{w}$ and $\mathbf{u}$. There are additional ideas that could be considered when selecting $\mathcal{F}$ or designing the cost functional. For example, for a fixed l_W, s_U should be a monotonically increasing function of s_W so that the cyclic order of the segments of $\mathbf{w}$ is preserved on $\mathbf{u}$.

Formulating a continuous HPM in this way and applying continuous optimization techniques may provide a way of avoiding the increase in computation associated with

searching over many candidate matches (*i.e.* using a larger or finer neighborhood grid). In particular, the regularized functional in eq.(4.5) could be faster to minimize than solving the dynamic program associated with HPM-DP when it is necessary to consider many candidate matches. This is important since it is precisely when there are many candidate matches that the soft, penalty-based approach used in HPM-DP is likely to be more appropriate than the simpler HPM-F.

4.5 Summary and Discussion

In this chapter, we have introduced a novel shape matching algorithm which performs well on benchmark shape retrieval tests. Matching boundary segments of different lengths avoids problems associated with global and local approaches. The hierarchical structure of the matching algorithm captures the intuitive notion that matching should proceed in a global to local direction. While comparison of multiscale shape representations is typically based on specific features such as curvature-zero crossings [Mokhtarian & Bober, 2003] or dominant points [Costa & Cesar, 2001], with HPM there is no need to define such features. The proposed approach generalizes the idea of finding a point-to-point correspondence between two shapes to that of finding a segment-to-segment correspondence. Indeed, we potentially have a different (though not independent) point-to-point correspondence at each different scale reflecting the fact that matching is dominated by features prominent at a given scale. This contrasts with methods which find a single point-to-point correspondence by effectively reparameterizing the curves (*e.g.* CPM and Sebastian et al. (2003)).

The results in Section 4.3 indicate that HPM is robust to a number of common transformations such as independent movement of parts and smooth deformations. Its reliance on segments makes it robust to viewpoint related changes in shape since the individual segments still match quite well even though the global alignment becomes poor (consider the bats class in Fig. 3.3). HPM requires ordered boundary information and assumes that the order in which corresponding segments appear is preserved across similar shapes. The ability of segments to stretch and slide along the boundary they are being matched to may enable HPM to handle missing features and slight occlusion, but it is not designed to handle significant occlusion or part rearrangement.

HPM-DP performed worst out of the different variants tested. A penalty-based ap-

proach to selecting the matches is likely to be more effective when segments are allowed to slide and stretch to a greater extent than was permitted here. However, it may only be necessary to use the predictions of the already matched longer segments in such cases and avoid the use of neighbor-based predictions which necessitate the use of dynamic programming. Also, the choice of penalty term has not been fully investigated – for example, the absolute difference between the predicted match and the chosen match (eq.(4.4)) may not be optimal.

The strong performance of HPM-Fn is particularly interesting. It seems likely that normalizing the PMSEs between a single segment of one shape and its candidate matches on the other shape leads to a simple form of novelty detection. For shape regions that have roughly constant curvature, all the candidate matches will be of a similar quality and all PMSEs will be similar. Consequently, the contribution of these regions to the similarity score will not be decisive. One could hypothesize that such regions are, on average, less useful for discrimination and that this effective down-weighting is desirable. On the other hand, there is often a definite best match for more complex segments (we can think of the segment clicking into place), and this will have a low normalized PMSE associated with it. Again, we might hypothesize that segments with an intricate structure (at the scale being considered) are more useful for discrimination and should have a greater impact on the similarity score. This is an interesting idea, though somewhat at odds with the notion of attaching equal significance to all parts of the shape and not identifying specific types of segment (Section 4.1).

As highlighted in Section 4.3.4, HPM is dependent on the initial correspondence chosen by CPM. More generally, the matches at a given scale must be at least approximately correct or we will end up searching for matches at the lower scale in the wrong place. Though important for efficiency, fixing the matches at higher scales before moving to the next scale is a weakness of HPM. Future work will investigate alternative formulations of HPM which avoids making these premature hard matches.

Chapter 5

Probabilistic Shape Matching

The remaining work in this thesis considers probabilistic approaches to shape matching and modeling; in this chapter, we continue to focus on pairwise matching problems.[1]

5.1 Introduction

Chapters 3 and 4 addressed the problem of matching unoccluded, closed contours. Though accurate, the proposed techniques cannot handle occlusion, are unable to utilize points from other parts of the shape (*e.g.* interior contours) and are not applicable to unordered point sets (*e.g.* comparing molecules). We begin this chapter by describing a probabilistic point matching (PPM) algorithm which operates on unordered point sets and is robust to occlusion. The approach has been proposed by various authors, but is covered in detail here since it forms the basis of the novel techniques presented in the remainder of the thesis.

Many objects have an inherent part-based structure whereby the possible configurations of the object can be realized (approximately) by applying rigid transformations to the individual parts. For example, animals, people, and many man-made objects have intuitive part structures. Molecules can take different configurations yet the relative arrangement of localized subsets of the atoms will often be approximately fixed; these subsets can be thought of as parts. In Section 5.3, we introduce a principled extension to the basic PPM algorithm specifically for part-based shapes.

[1] Publication details for this chapter: Section 5.3 – [McNeill & Vijayakumar, 2006c], Section5.4 – [McNeill & Vijayakumar, 2006d].

In Section 5.4 we extend the PPM model to the case where one of the shapes is described by a set of line segments. This removes the bias associated with point set representations of lines, and leads to faster matching for shapes composed of relatively few line segments. Just as PPM can be seen as an extension of the iterative closest point algorithm (ICP) [Besl & McKay, 1992] for matching two points sets, probabilistic line matching (PLM) extends the ICP for matching a point set to a line set.[2] The arguments in favor of PPM and PLM over the 'hard correspondence' based ICP are the same: a soft correspondence leads to fewer problems with local minima, outliers are easily handled using a background model, and the variance parameter can either be used to indicate the required match quality or annealed during matching.

5.2 Unordered Point Sets

The algorithms introduced in this and the following chapter can be applied to shapes of any dimension, so we will put aside the complex notation used in Chapters 3 and 4 (convenient for 2D shapes) and switch to standard matrix notation.[3] Our objective is to compute the correspondence and match between the unordered point sets $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{J_X})^T \in \mathbb{R}^{J_X \times M}$, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{J_Y}) \in \mathbb{R}^{J_Y \times M}$. The two point sets can have different sizes ($J_X \neq J_Y$) and the number of dimensions, M, is arbitrary. Conceptually, we think of $\mathbf{X}$ as the *model shape* or *generating shape* which is transformed and then sampled from to give the data points $\mathbf{y}_j$ of $\mathbf{Y}$. The approach involves a Gaussian mixture model (GMM) (eq.(5.3)) whereby the center of the k-th mixture component is dependent on the point $\mathbf{x}_k$ (eq.(5.4)). Given this, we refer to the probability of the k-th mixture component and the probability of the point $\mathbf{x}_k$ interchangeably.

Formally, we assume that the $\mathbf{y}_j$ are observations from the (higher level) mixture model

$$p(\mathbf{y}_j) = p(\mathbf{y}_j|v=0)p(v=0) + p(\mathbf{y}_j|v=1)p(v=1), \tag{5.1}$$

[2]The ICP algorithm for matching two point sets iterates between aligning the point sets given a putative correspondence and choosing a correspondence given the current alignment.

[3]Examples using 3D shapes are considered in Chapter 6.

where

$$p(\mathbf{y}_j|v=0) = \text{Uniform}, \tag{5.2}$$

$$p(\mathbf{y}_j|v=1) = \frac{1}{J_X}\sum_{k=1}^{J_X} p(\mathbf{y}_j|k,v=1). \tag{5.3}$$

The mixture component $v=0$ represents the 'background model' which ensures that all data points are explained to some extent, and hence, robustifies the model against outliers. The 'foreground' component $v=1$ is the interesting part. The distribution $p(\mathbf{y}_j|v=1)$ (eq.(5.3)) is the GMM referred to above, but unlike a standard GMM, the movement of the centers is controlled by a single set of transformation parameters s, Γ and $\mathbf{c}$:

$$\mathbf{y}_j|(k,v=1) \sim \mathcal{N}(s\Gamma\mathbf{x}_k + \mathbf{c}, \sigma^2\mathbf{I}), \tag{5.4}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes the Gaussian distribution, s is a scale parameter, $\mathbf{c} \in \mathbb{R}^M$ is a translation vector and Γ is an $M \times M$ rotation matrix.[4]

Note that we set $p(k|v=1) = 1/J_X$ for all k in eq.(5.3) rather than learning these mixture coefficients from the data. This prevents any single $\mathbf{x}_k$ having a large weight and thus, explaining a large number of the $\mathbf{y}_j$ – intuitively, a single point of one shape should not explain a large region of the second shape. Of course, some $\mathbf{x}_k$ may not explain any $\mathbf{y}_j$ (*e.g.* due to occlusion) and ideally $p(k|v=1)$ should be zero for these $\mathbf{x}_k$; we have found that allowing for this possibility has little impact on the results.

To recover matches from poor initial alignments yet still find a tight match where possible, we anneal the variance parameter σ^2 during matching (as in Revow et al. (1996)). Maximum likelihood estimates for the other model parameters ($\{s, \Gamma, \mathbf{c}\}$ and $p(v)$) are found using the expectation maximization (EM) algorithm [Dempster et al., 1977] as follows:

E-step: Compute the *responsibilities* using the current parameter values and eqs.(5.1)-(5.4):

$$p(k,v=1|\mathbf{y}_j) = \frac{p(\mathbf{y}_j|k,v=1)\frac{1}{J_X}p(v=1)}{p(\mathbf{y}_j)} \tag{5.5}$$

[4]The background model could be seen as the (J_X+1)-th component of a 'flat mixture model' and thus, we could avoid using the higher level variable v altogether. The hierarchical mixture model defined by eqs.(5.1)-(5.4) is used here because it generalizes easily to the parts-based and line-based cases considered in the following sections.

M-step: Update the *parameters* using the responsibilities:

$$p(v=1) = \frac{1}{J_Y}\sum_{j=1}^{J_Y} p(v=1|\mathbf{y}_j), \quad p(v=0) = 1-p(v=1), \tag{5.6}$$

$$(s,\Gamma,\mathbf{c}) = \arg\min_{s,\Gamma,\mathbf{c}} \sum_{j=1}^{J_Y}\sum_{k=1}^{J_X} p(k,v=1|\mathbf{y}_j)\|\mathbf{y}_j - s\Gamma\mathbf{x}_k - \mathbf{c}\|^2. \tag{5.7}$$

Eq.(5.7) is a *weighted Procrustes matching problem* between two points sets, each of size $J_Y \times J_X$; the importance of matching the pair $(\mathbf{x}_k, \mathbf{y}_j)$ is given by $p(k, v=1|\mathbf{y}_j)$. This problem is solved analytically as described in Section 2.4.

Fig. 5.1 shows how PPM performs on matching problems involving occlusion, irregular sampling and *local dissimilarity* – a feature of one shape is not present or is significantly different on the other shape. The black points are the $\mathbf{x}_k$ that are collectively transformed to match the green $\mathbf{y}_j$; the final matches are scaled up for clarity. See that many of the $\mathbf{x}_k$ have no counterparts in (i), whereas it is the $\mathbf{y}_j$ that have no counterparts in (iii). In all cases, the sampling frequency of corresponding sections is different, so there is no perfect point-to-point match. An example of applying PPM to real images is shown in Fig. 5.2. Note that the points with no counterparts (notably the blade in the first image and the pencil and writing on the corkscrew in the second) do not prevent the correct match being found. The results of applying PPM to the MPEG-7 data (Sections 3.3 and 4.3.1) are discussed in Section 5.4.2.

As mentioned at the beginning of the chapter, this probabilistic approach to matching unordered point sets has been proposed previously. The algorithms proposed by Chui and Rangarajan (2000a), Luo and Hancock (2002) and Taylor et al. (2003) all use a Gaussian mixture model formulation, though Luo and Hancock (2002) use an alternative approach to handling outliers whilst Chui and Rangarajan (2000a) consider both linear and nonlinear transformations. The 'Earth Mover's Distance' [Cohen & Guibas, 1999] and the approaches of Rangarajan et al. (1997) and Chui and Rangarajan (2000b) are not probabilistic but are strongly related to PPM. All these algorithms are essentially variants of the well-known Iterative Closest Point (ICP) algorithm [Besl & McKay, 1992], a key difference being that ICP assumes a *hard correspondence* whereas these use a *soft correspondence* (computed during the E-step, eq(5.5)). Another important advantage of PPM over ICP is its ability to handle outliers and occlusion. In future, we intend to quantitatively assess this ability using the 'incomplete contour representation tests' (ICR) proposed by Ghosh and Petkov (2005).

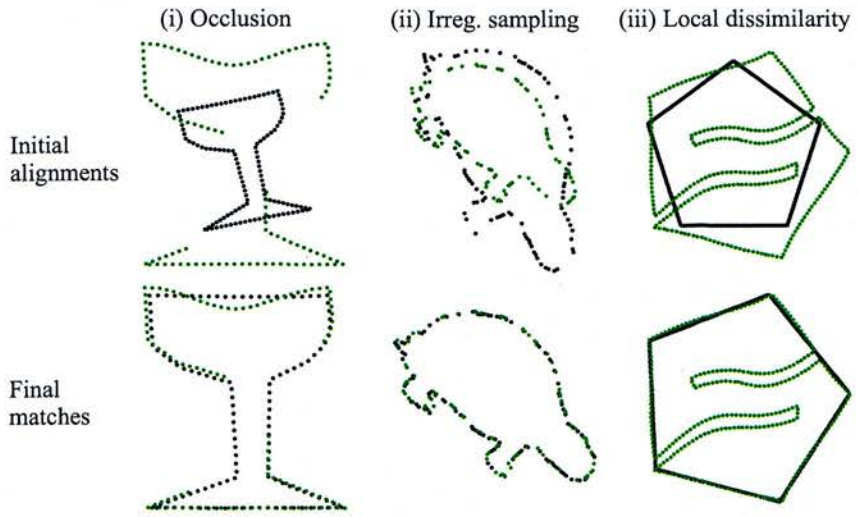

Figure 5.1: Examples – Probabilistic Point Matching (PPM).

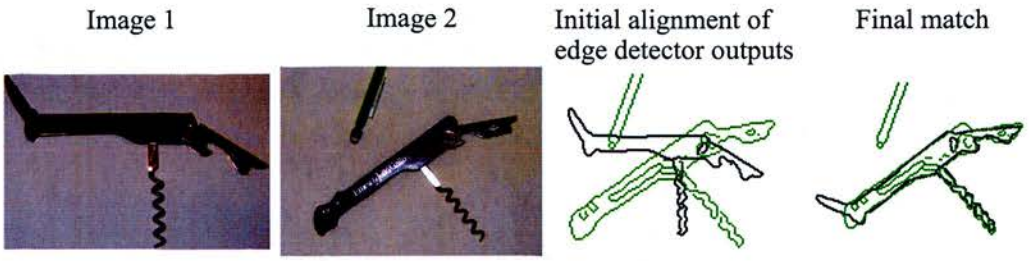

Figure 5.2: PPM applied to shapes extracted from real images.

Alternative approaches to matching unordered point sets have been proposed. Huttenlocher et al. (1993) use the alignment transformation that minimizes the Hausdorff distance.[5] Belongie et al. (2002) describe the neighborhood of each point using a radial histogram (referred to as the 'Shape Context' at that point) and correspond points based on neighborhood similarity. The Distance Sets approach of Grigorescu and Petkov (2003) uses an alternative local descriptor and has been shown to outperform Shape Contexts on both the standard MPEG-7 bullseye test (Table 4.1) and the ICR tests [Ghosh & Petkov, 2005] mentioned above. We shall see in Section 5.4.2 that PPM outperforms both of these techniques on the bullseye test; its performance on the ICR tests has not been evaluated.

[5]The Hausdorff distance: compute the distance between each point of the first shape and its nearest neighbor on the second shape and then take the maximum of these distances.

In Section 5.4, PPM is extended to handle the case where one of the shapes to be matched is ordered and the other is unordered; first, we consider the common yet notoriously difficult problem of matching part-based shapes.

5.3 Part-based Probabilistic Matching (PBPM)

The PPM algorithm described in the previous section matches two shapes using a single linear transformation. As discussed in Chapter 4, such a transformation may not be powerful enough to match perceptually similar shapes (*e.g.* Fig. 4.1). Chui and Rangarajan (2000a) have shown that replacing the linear transformation with a smooth nonlinear transformation (a thin-plate spline) is an effective solution to this problem in some cases. However, this modification is not suited to the common problem of matching part-based shapes. In the examples in Fig. 5.3, the shapes are naturally matched by applying different linear transformations to the different parts. Globally, such a transformation is discontinuous and would not be found by searching for a smooth nonlinear transformation. In this section, we extend PPM to *Part-based Probabilistic Matching* (PBPM) which handles parts explicitly by learning the most likely part structure and correspondence simultaneously. Two different notions of parts are useful here:

Natural Parts Most shapes have a *natural part decomposition* (NPD) (Fig. 5.4) and there are several algorithms available for finding NPDs (*e.g.* Bennemoun (1994), Siddiqi and Kimia (1995), Kim et al. (2005)). In object recognition and CBIR, the *query image* is frequently a template shape (a binary image or a line drawing) or a high quality image with no occlusion or clutter; either of these is a suitable input to a NPD algorithm.[6] In contrast, the *database images* to be searched are likely to be much more complex and obtaining a reliable NPD for these may not be possible. With this in mind, we focus on pairwise matching problems where we have access to a NPD of one shape only. Finally, note that a NPD is a fixed property of a single shape – it does not arise during the matching process.

Variation-based Parts A different notion of parts has been used in computer vision [Titsias, 2005], where a part is defined as a set of pixels that undergo the same

[6]The NPDs used in the examples were constructed manually.

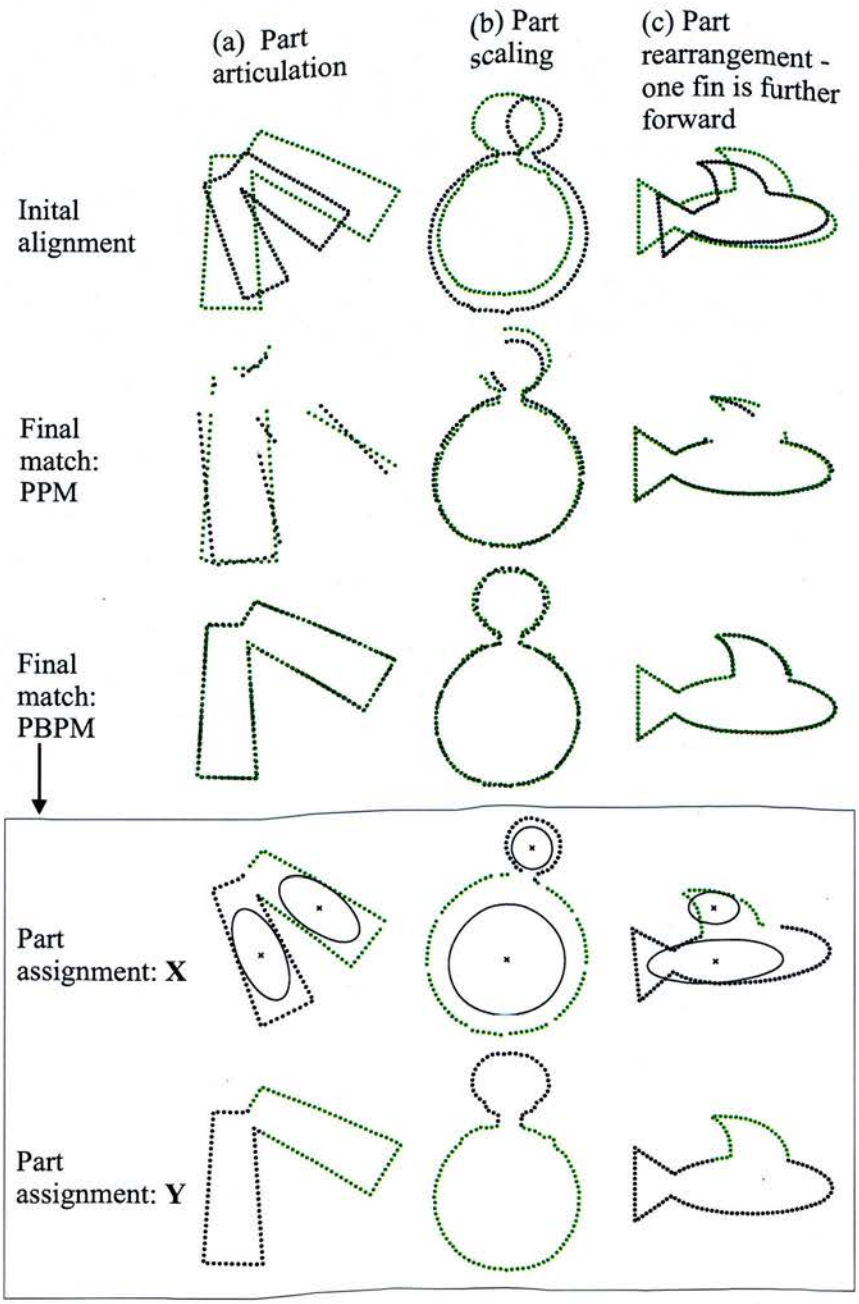

Figure 5.3: PPM fails to match shapes that display simple part-based variation whereas PBPM correctly matches the shapes by learning the correct part structure. No natural part decomposition (NPD) information was used in these examples.

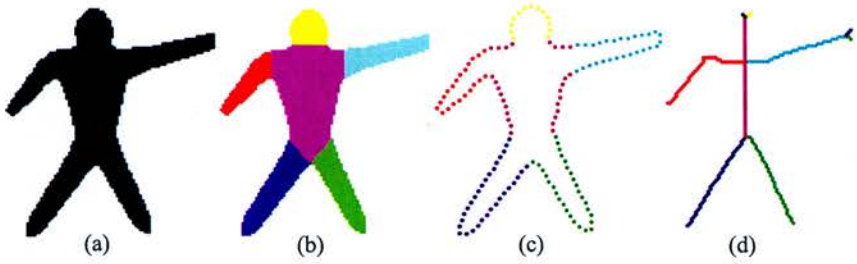

Figure 5.4: The natural part decomposition (NPD) (b-d) for different representations of a shape (a).

transformation across images. Unlike natural parts, variation-based parts only arise through the comparison of two or more shapes.

PBPM matches shapes by finding the variation-based parts. However, this is an under-constrained problem, so it is important to incorporate prior knowledge about what makes a 'perceptually valid' part. Here, two types of prior knowledge are used: Firstly, parts should be spatially localized – intuitively, a part is not composed of subparts from different locations on the shape. Secondly, we utilize any available NPD information by forcing variation-based parts to be composed of one or more *natural parts*. In other words, the natural parts (rather than the individual points) are the atomic elements which are grouped together to form variation-based parts. The basic PBPM algorithm is described in detail in the next section. In Section 5.3.3, we introduce a sequential approach for tackling model selection (the number of parts) and parameter initialization.

5.3.1 The Model

The PPM model in Section 5.2 can be seen as a special instance of a part-based model: it has one background part, $v = 0$, and one foreground part, $v = 1$. We now consider models with one background part, $v = 0$, and V foreground parts, $v \in \{1, 2, \ldots, V\}$, where (*cf.* eq.(5.4))

$$\mathbf{y}_j | k, v \sim \mathcal{N}(s_v \Gamma_v \mathbf{x}_k + \mathbf{c}_v, \sigma^2 \mathbf{I}), \quad v = 1, \ldots, V. \tag{5.8}$$

Thus, the part label v indexes transformation parameters where a part is implicitly defined as those points of $\mathbf{X}$ that undergo the same transformations. Note that parts of $\mathbf{X}$ are only defined in the context of a given $\mathbf{Y}$ that $\mathbf{X}$ is being matched to.

We could proceed by writing $p(\mathbf{y}_j, k, v) = p(\mathbf{y}_j|k,v)p(k|v)p(v)$ and learning every entry of a $J_X \times V$ conditional probability table for $p(k|v)$. However, this would place no constraints on what a part should look like so instead, the distribution $p(k|v)$ is used to incorporate the idea of spatial localization discussed above.[7] This is achieved by replacing the uniform distribution of $p(k|v=1)$ in PPM with

$$p(k|v) = \frac{\exp\{-(\mathbf{x}_k - \mu_v)^T \Sigma_v^{-1} (\mathbf{x}_k - \mu_v)/2\}}{\sum_{k'} \exp\{-(\mathbf{x}_{k'} - \mu_v)^T \Sigma_v^{-1} (\mathbf{x}_{k'} - \mu_v)/2\}}, \quad v \in \{1,2,\ldots,V\}, \tag{5.9}$$

where $\mu_v \in \mathbb{R}^2$ is a mean vector and $\Sigma_v \in \mathbb{R}^{2\times 2}$ is a covariance matrix. In words, we identify $k \in \{1,\ldots,J_X\}$ with the point $\mathbf{x}_k$ that it indexes and assume that the distribution $p(\mathbf{x}_k|v)$ is a bivariate Gaussian. Since k must take a value in $\{1,\ldots,J_X\}$, the distribution is normalized using the points $\mathbf{x}_1,\ldots,\mathbf{x}_{J_X}$ only. This assumption means that the $\mathbf{x}_k$ themselves are essentially generated by a Gaussian mixture model (GMM) with V components. However, this GMM is embedded in the larger model and maximizing the data likelihood will balance this GMM's desire for localized parts against the need for the parts and transformations to explain the actual data (the $\mathbf{y}_j$).

The next step is to incorporate the NPD information into the model. To achieve this, we use ideas from semi-supervised learning and in particular the work of Shental et al. (2004) on constrained mixture models. We briefly review this work and then describe how the idea is applied to the above model.

The standard approach to parameter estimation in a mixture model involves introducing an unobserved variable for each data point. This 'hidden label' indicates which mixture component generated the data point that it is associated with. The EM algorithm is used to estimate the model parameters which involves taking the expectation of the log-likelihood with respect to the hidden variables – *i.e.* involves summing over all valid combinations of the hidden labels. Shental et al. (2004) considered the situation where, in addition to observing data points, it was known that certain subsets of data points were generated by the same (but unknown) mixture component. They proposed a modified algorithm whereby only values of the hidden labels which are consistent with this additional "side information" are considered when taking the expectation. In other words, they essentially fix the posterior probability of the hidden labels to zero for

[7] Even if the learnt parts lack any perceptual meaning, it seems reasonable that difficult correspondence problems may be solved by splitting shapes into spatially localized pieces.

choices of labels not consistent with the side information.[8] Note that any data points not covered by the side information belong to subsets of size one, so the technique uses as much information as is available and does not require this information to be comprehensive.

The model described at the beginning of this section is a mixture model ($p(\mathbf{y}_j) = \sum_v p(\mathbf{y}_j|v)p(v)$) and here, the side information comes from the NPD of $\mathbf{Y}$ which we express as a partition: $\mathbf{Y} = \bigcup_{l=1}^{L} \mathbf{Y}_l$. When only 'valid' values of the hidden variables are permitted (*i.e.* points in the same $\mathbf{Y}_l$ are generated by the same variation-based-part, v) the expected complete log-likelihood is given by (see Shental et al. (2004) for details)

$$\mathcal{E}_c = \sum_{v=0}^{V}\sum_{l=1}^{L} p(v|\mathbf{Y}_l)\log p(v) + \sum_{v=0}^{V}\sum_{l=1}^{L}\sum_{\mathbf{y}_j\in\mathbf{Y}_l} p(v|\mathbf{Y}_l)\log p(\mathbf{y}_j|v). \tag{5.10}$$

In some applications, we may wish to discourage large differences in the part transformations in order to prevent the generating shape breaking up and explaining data in different areas of an image. This was not found to be necessary for the examples considered in Sections 5.3.2 and 5.3.3, but such a soft constraint could be implemented by including a penalty term in eq.(5.10). Note that eq.(5.10) involves $p(v|\mathbf{Y}_l)$ – the responsibility of a component v for a subset $\mathbf{Y}_l$, rather than the term $p(v|\mathbf{y}_j)$ that would be present in an unconstrained mixture model. Rearranging eq.(5.10) slightly and using $p(\mathbf{y}_j|v) = \sum_k p(\mathbf{y}_j|k,v)p(k|v)$, we have

$$\begin{aligned}\mathcal{E}_c \;=\; & \sum_{v=0}^{V}\sum_{l=1}^{L} p(v|\mathbf{Y}_l)\log p(v) + \sum_{l=1}^{L} p(v=0|\mathbf{Y}_l)\log\{u^{|\mathbf{Y}_l|}\} \\ & + \sum_{v=1}^{V}\sum_{l=1}^{L}\sum_{\mathbf{y}_j\in\mathbf{Y}_l} p(v|\mathbf{Y}_l)\log\left\{\sum_{k=1}^{K} p(\mathbf{y}_j|k,v)p(k|v)\right\},\end{aligned} \tag{5.11}$$

where u is the constant associated with the uniform distribution $p(\mathbf{y}_n|v=0)$. The parameters to be estimated are $p(v)$, μ_v, Σ_v and the transformations $\{s_v, \Gamma_v, \mathbf{c}_v\}$. With the exception of $p(v)$, these are found by maximizing the final term in eq.(5.11). For a fixed v, this term is the log-likelihood of data points $\mathbf{y}_1, \ldots, \mathbf{y}_{J_Y}$ under a mixture model, with the modification that there is a weight, $p(v|\mathbf{Y}_l)$, associated with each data point. The simplest way to proceed therefore, is to treat this subproblem as a standard mixture model maximum likelihood problem and derive the EM updates as usual. Theoretically,

[8] The derivation of the modified EM in Shental et al. (2004) is more complicated than suggested above; here we focus on the most intuitive interpretation of their approach.

this is rather simplistic since there is no mechanism to prevent a single $\mathbf{x}_k$ belonging to multiple parts, v. However, in practice we have found that this approach does not typically lead to overlapping parts.[9] The resulting EM algorithm is given below.

E-step. Compute the *responsibilities* using the current parameters:

$$p(k|\mathbf{y}_j,v) = \frac{p(\mathbf{y}_j|k,v)p(k|v)}{\sum_k p(\mathbf{y}_j|k,v)p(k|v)}, \quad v=1,2,\ldots,V \tag{5.12}$$

$$p(v|\mathbf{Y}_l) = \frac{p(v)\prod_{\mathbf{y}_j\in\mathbf{Y}_l} p(\mathbf{y}_j|v)}{\sum_v p(v)\prod_{\mathbf{y}_j\in\mathbf{Y}_l} p(\mathbf{y}_j|v)} \tag{5.13}$$

M-step. Update the *parameters* using the responsibilities:

$$p(v) = \frac{1}{L}\sum_{l=1}^{L} p(v|\mathbf{Y}_l) \tag{5.14}$$

$$\mu_v = \frac{\sum_{j,k} p(v|\mathbf{Y}_{l,j})p(k|\mathbf{y}_j,v)\mathbf{x}_k}{\sum_{j,k} p(v|\mathbf{Y}_{l,j})p(k|\mathbf{y}_j,v)} \tag{5.15}$$

$$\Sigma_v = \frac{\sum_{j,k} p(v|\mathbf{Y}_{l,j})p(k|\mathbf{y}_j,v)(\mathbf{x}_k-\mu_v)(\mathbf{x}_k-\mu_v)^T}{\sum_{j,k} p(v|\mathbf{Y}_{l,j})p(k|\mathbf{y}_j,v)} \tag{5.16}$$

$$T_v = \arg\min_T \sum_{j,k} p(v|\mathbf{Y}_{l,j})p(k|\mathbf{y}_j,v)\|\mathbf{y}_j - T_v\mathbf{x}_k\|^2 \tag{5.17}$$

where $\mathbf{Y}_{l,j}$ is the subset $\mathbf{Y}_l$ containing $\mathbf{y}_j$. As with eq.(5.7) in PPM, eq.(5.17) is a weighted Procrustes problem that can be solved analytically as described in Section 2.4.

The weights associated with the updates in eqs.(5.15)-(5.17) are similar to

$$p(v|\mathbf{y}_j)p(k|\mathbf{y}_j,v) = p(k,v|\mathbf{y}_j), \tag{5.18}$$

the responsibility of part v and point $\mathbf{x}_k$ for the observed data, $\mathbf{y}_j$. The difference is that $p(v|\mathbf{y}_j)$ is replaced by $p(v|\mathbf{Y}_{l,j})$ and hence, the impact of the side information is propagated throughout the model.

5.3.2 Examples

Matching part-based shapes is a relatively unexplored area which makes it difficult to quantitatively assess the performance of PBPM. In this and the following section, we

[9] An alternative solution would be to introduce a latent variable (a label) for each $\mathbf{x}_k$ indicating the unique part v to which it belongs.

provide illustrative examples which demonstrate the various properties of PBPM. The number of parts, V, is fixed prior to matching in this section; a technique for estimating V is described in Section 5.3.3. To visualize the matches found by PBPM, each point $\mathbf{y}_j$ is assigned to a part v using $\max_v p(v|\mathbf{y}_j)$. Points assigned to $v = 0$ are removed from the figure. For each $\mathbf{y}_j$ assigned to some $v \in \{1,\ldots,V\}$, we find $k_j \equiv \arg\max_k p(k|\mathbf{y}_j, v)$ and assign $\mathbf{x}_{k_j}$ to v. Those $\mathbf{x}_k$ not assigned to any parts are removed from the figure. The means and the ellipses of constant probability density associated with the distributions $\mathcal{N}(\mu_v, \Sigma_v)$ are plotted on the original shape $\mathbf{X}$ to show the spatial coverage of each part. We also assign the $\mathbf{x}_k$ to natural parts using the known natural part label of the $\mathbf{y}_j$ that they are assigned to.

Fig. 5.5 shows an example of matching two human body shapes using PBPM with $V = 3$. The top row shows the input to PBPM: the two shapes $\mathbf{X}$ and $\mathbf{Y}$, the known NPD of $\mathbf{Y}$ and the initial alignment of the shapes. The bottom row summarizes the results of the PBPM algorithm: the learnt variation-based part decomposition (VPD) of $\mathbf{X}$, the transformed $\mathbf{X}$, the learnt VPD of $\mathbf{Y}$, the NPD of $\mathbf{X}$ (see previous paragraph) and the final match of the two shapes. The learnt VPD in Fig. 5.5 is intuitive and the match is better than that found using PPM (Fig. 5.6, bottom left). The results obtained using different values of V are shown in Fig. 5.6. Predictably, the match improves as V increases, but the improvement is negligible beyond $V = 4$. When $V = 5$, one of the parts is effectively repeated, suggesting that four parts is sufficient to cover all the interesting variation. However, when $V = 6$ all parts are used and the VPD looks very similar to the NPD – only the lower leg and foot on each side are grouped together.

In Fig. 5.7, there are two genuine variation-based parts and $\mathbf{X}$ contains additional features. PBPM correctly identifies the two parts and and is not confused by the extra features.

In Fig. 5.8, a template shape is matched to the edge detector output from a real image. Six parts were used for matching, but two of these are initially assigned to clutter and end up playing no role in the final match. The object of interest in $\mathbf{X}$ is well matched to the template using the other four parts.

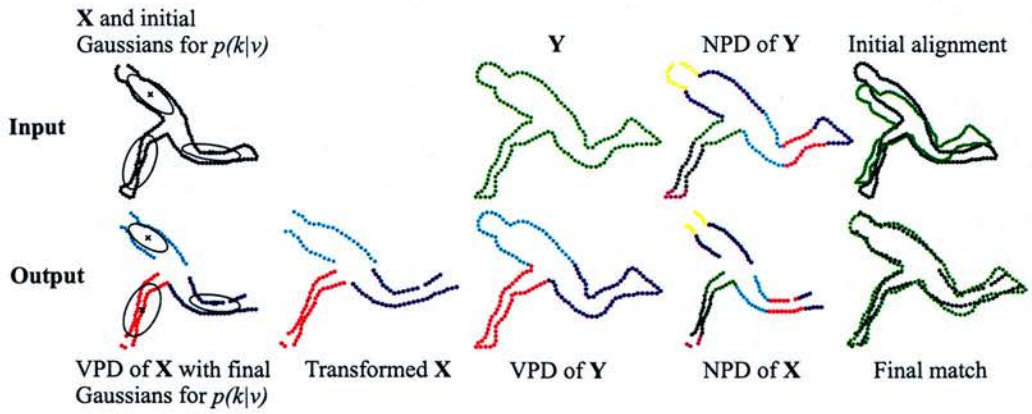

Figure 5.5: An example of applying PBPM with $V = 3$.

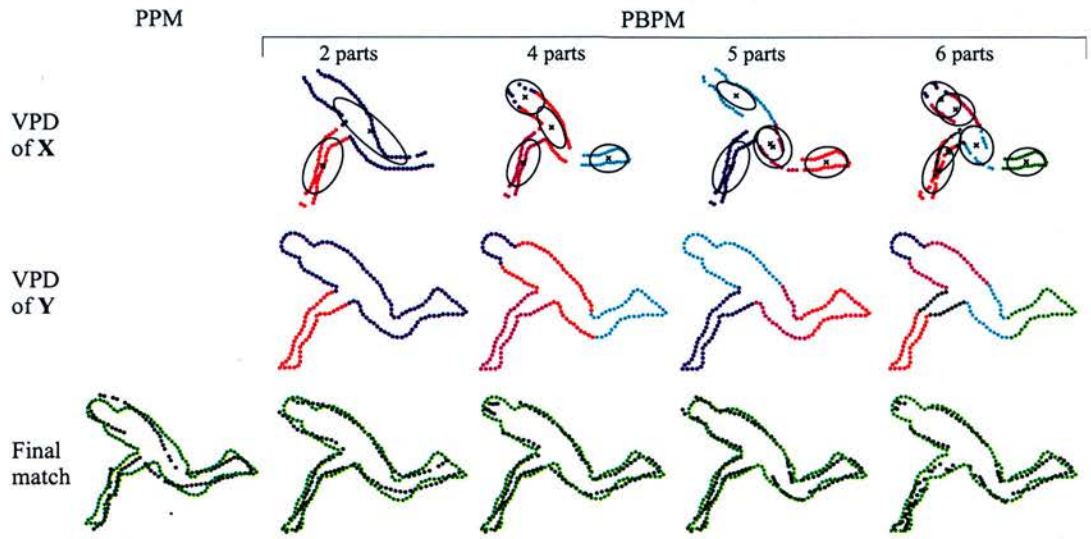

Figure 5.6: Results for the problem in Fig. 5.5 using PPM and PBPM with $V = 2, 4, 5$ and 6.

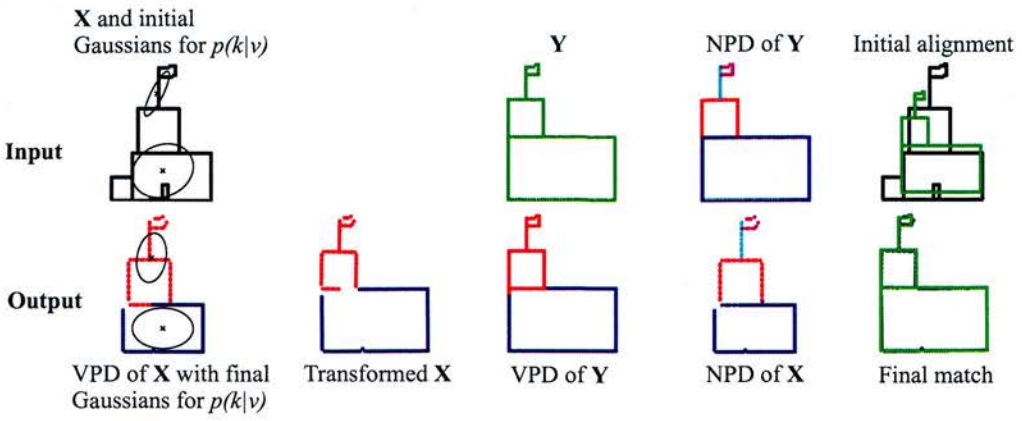

Figure 5.7: Some features of $\mathbf{X}$ are not present on $\mathbf{Y}$; the main building of $\mathbf{X}$ is smaller and the tower is more central.

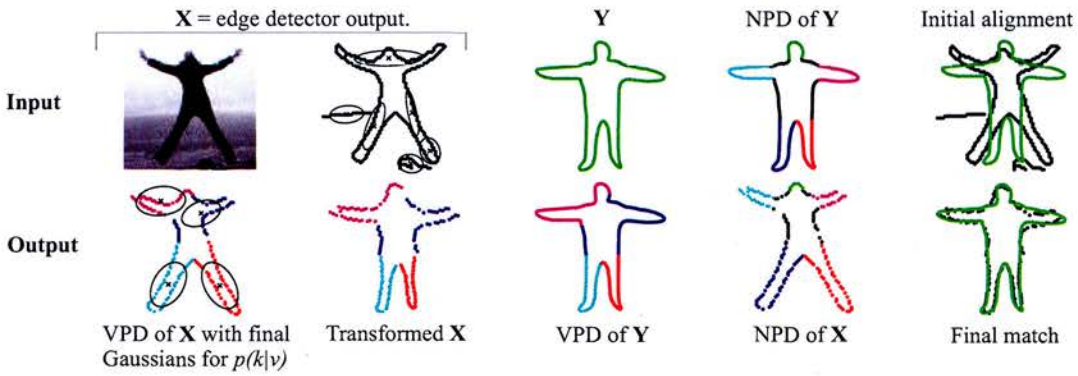

Figure 5.8: Matching a template shape to a real image.

5.3.3 Sequential Algorithm for Initialization

When part variation is present, one would expect PBPM with $V = 1$ to find the most significant part and allow the background to explain the remaining parts. This suggests a sequential approach to finding V whereby, at each stage, a single new part is used to explain points that currently belong to the background. This is achieved by modifying the technique described by Titsias (2005) for fitting mixture models sequentially. Specifically, assume that the first part ($v = 1$) has been learnt and now learn the second part using the weighted log-likelihood

$$\mathcal{L} = \sum_{l=1}^{L} z_l^1 \log\{p(\mathbf{Y}_l|v=2)p(v=2) + u^{|\mathbf{Y}_l|}(1 - p(v=1) - p(v=2))\}. \tag{5.19}$$

Here, $p(v=1)$ is known and

$$z_l^1 \equiv \frac{u^{|\mathbf{Y}_l|}(1-p(v=1))}{p(\mathbf{Y}_l|v=1)p(v=1) + u^{|\mathbf{Y}_l|}(1-p(v=1))} \tag{5.20}$$

is the responsibility of the background component for the subset $\mathbf{Y}_l$ after learning the first part – the superscript of z indicates the number of components that have already been learnt. Using the modified log-likelihood in eq.(5.19) has the desired effect of forcing the new component ($v = 2$) to explain the data currently explained by the uniform (background) component. Note that we use the responsibilities for the subsets $\mathbf{Y}_l$ rather than the individual $\mathbf{y}_j$ [Titsias, 2005], in line with the assumption that complete subsets belong to the same part. Also, note that eq.(5.19) is a weighted sum of log-likelihoods over the subsets, it cannot be written as a sum over data points since these are not sampled i.i.d. due to the side information. Maximizing eq.(5.19) leads to similar EM updates to those given in eqs.(5.12)-(5.17). Having learnt the second part, additional components $v = 3,4,\ldots$ are learnt in the same way except for minor adjustments to eqs.(5.19) and (5.20) to incorporate all previously learnt components. The sequential algorithm terminates when the background is not significantly responsible for any data or the most recently learnt component is not significantly responsible for any data. Just as in Titsias (2005), the sequential approach is likely to encounter fewer problems with local minima than the full PBPM model since the objective function will be smoother (a single component competes against a uniform component at each stage) and the search space smaller (fewer parameters are learnt at each stage).

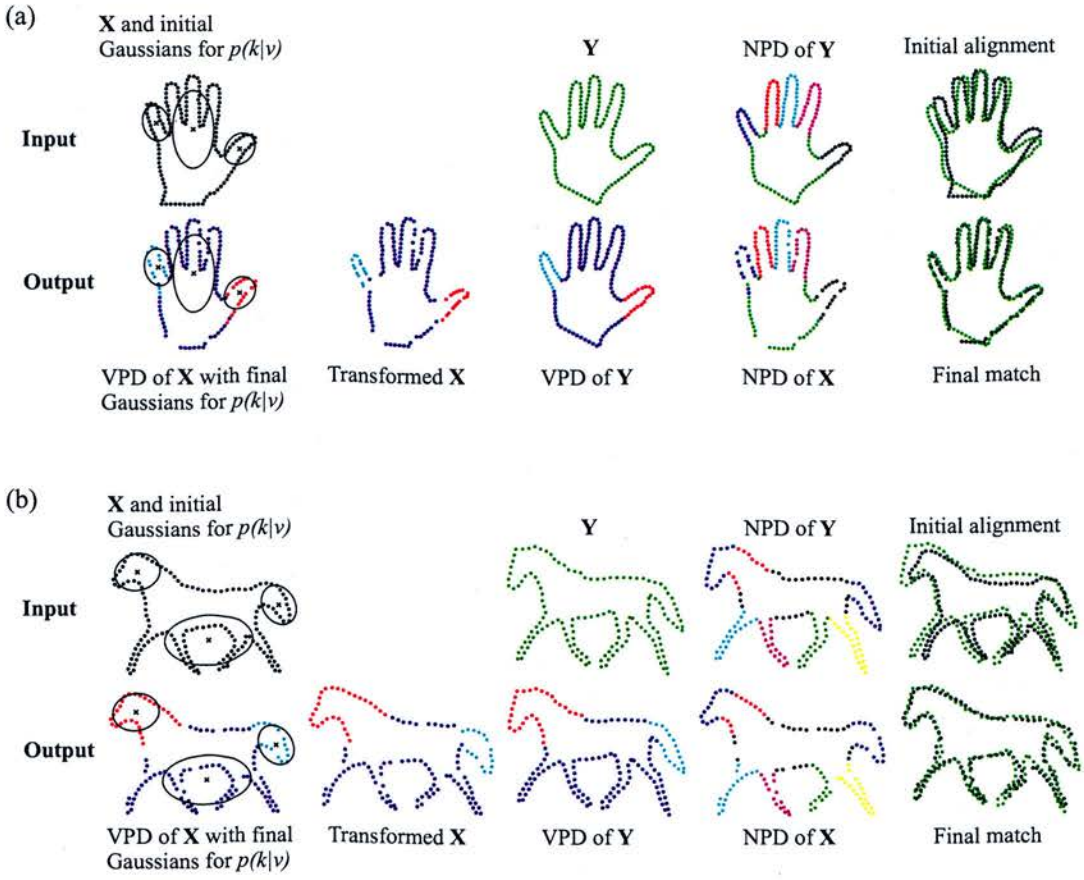

Figure 5.9: Examples of PBPM; V and initial parameters were found using the sequential approach.

Fig. 5.9(a) shows the results of applying the standard PBPM algorithm after using the sequential approach to select the number of parts and initial parameter values. From the plot of the initial alignment, it is clear that the two hand shapes only differ significantly at the thumbs and little fingers. Accordingly, the sequential algorithm chooses $V = 3$ and appropriate estimates for the Gaussians $p(k|v)$ (top left of Fig. 5.9) and the transformations. A similar situation occurs in Fig. 5.9(b), where the most notable difference is between the heads and tails of the horses.

A weakness of the sequential algorithm is that its behavior is highly dependent on the value of the uniform density, u. We are currently investigating how the model can be made more robust to this value and also how the used $\mathbf{x}_k$ should be subtracted (in a probabilistic sense) at each step.

5.3.4 Discussion

Matching a pair of part-based shapes is a severely under-constrained problem and consequently, the PBPM algorithm is more susceptible to local minima than PPM. When available, the NPD provides valuable additional constraints, enabling PBPM to solve matching problems such as those shown in Figs. 5.5-5.9. However, robustness remains an issue and it would be interesting to see if other constraints could be used to further improve reliability. For example, the technique of Luo and Hancock (2003) for encouraging neighborhood consistency between corresponding points might be extended to the part-based case. Alternatively, one could argue that even if further constraints are included, it is unrealistic to expect that part-based shapes can be matched reliably without stronger prior information about the part structure of the shapes beng considered. In other words, that PBPM is too generic and contains too many open parameters relative to the amount of data (two shapes) and constraints available.

Given the above discussion and the inability of existing algorithms to handle part-based shapes, it is clear that part-based matching is an open and difficult problem. We believe that PBPM (and the ideas that it is based on) are an important step towards fully understanding and solving the problem, but recognize that more testing and development is needed. Given the limited time available and that a key objective of the thesis is to extend pairwise matching to class models (Chapters 6 and 7), we have chosen not to investigate the problem any further in this thesis.

5.4 Matching a Line Set to a Point Set

In some matching problems, one of the shapes may be more naturally represented as a set of lines rather than a set of points. For example, in a retrieval scenario the query shape/image may be a sketch, line drawing, or high quality image from which the shape's boundary can be extracted. Ideally, the model should reflect the continuous nature of the shape in such cases – there should be a continuous tube of probability mass around each line, rather than blobs of probability mass placed at equal intervals along the line as in

PPM (Section 5.2). Also, we will see that a model which uses a line-based generating shape is more computationally efficient in cases where few line segments are required to accurately approximate the shape.

We will focus on the case where a shape's boundary is described by a polygon with G vertices, but the same algorithm can be used whenever one of the shapes is represented by an arbitrary set of line segments. Let $\mathbf{r}(k)$ be the polygon associated with the vertex set $(\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_G)^T \in \mathbb{R}^{G\times 2}$, where $k \in [0,1)$ is the curve parameter and $0 = k_1 < k_2 < \ldots < k_G < 1$ are the values of k that correspond to the vertices – *e.g.* if the vertex $\mathbf{x}_g$ lies 70% of the way round the perimeter of $\mathbf{r}$, then k_g=0.7. The probabilistic line matching (PLM) model is the same as PPM except that the variable k, which indexes mixture components, is now continuous. Hence, we generate a data point by first selecting a point on the polygon using $p(k)$ and then generating $\mathbf{y}$ from an isotropic Gaussian centered at this point. As before, we assume k is uniformly distributed but now this uniform distribution is over the continuous interval $[0,1)$ rather then the discrete set $\{1,2,\ldots,K\}$.

The EM updates for PLM are the same as those for PPM (eqs.(5.5)-(5.7)) except that k is now continuous and hence, summations over k become integrals. This creates the following problems. Firstly, we must evaluate

$$p(\mathbf{y}_j|v=1) = \int_0^1 p(\mathbf{y}_j|k,v=1)dk \tag{5.21}$$

in order to compute $p(\mathbf{y}_j)$ (where we have used $p(k|v=1) \sim \text{Uniform}(0,1)$, so the value of the probability density function is 1 in the interval $[0,1]$). The conditional distribution for $p(\mathbf{y}_j|k,v=1)$ in eq.(5.4) becomes

$$\mathbf{y}_j|(k,v=1) \sim \mathcal{N}(s\Gamma\mathbf{r}(k)+\mathbf{c}, \sigma^2\mathbf{I}), \tag{5.22}$$

in the continuous setting; substituting this into eq.(5.21) and defining $T\mathbf{r}(k) \equiv s\Gamma\mathbf{r}(k)+\mathbf{c}$ to prevent the algebra from becoming too cumbersome, we can write

$$\begin{aligned} p(\mathbf{y}_j|v=1) &= \frac{1}{2\pi\sigma^2}\int_0^1 \exp\left\{\frac{-1}{2\sigma^2}\|\mathbf{y}_j - T\mathbf{r}(k)\|^2\right\}dk \\ &= \frac{1}{2\pi\sigma^2}\sum_{g=1}^{G}\int_{k_g}^{k_{g+1}} \exp\left\{\frac{-1}{2\sigma^2}\|\mathbf{y}_j - T\mathbf{r}(k)\|^2\right\}dk \\ &= \frac{1}{2\pi\sigma^2}\sum_{g=1}^{G} I_{gj}. \end{aligned} \tag{5.23}$$

Here, we have split up the integral around the polygon (*i.e.* over k) into segments and defined I_{gj} to be the integral along the segment between $\mathbf{x}_g$ and $\mathbf{x}_{g+1}$ (where $\mathbf{x}_{G+1} \equiv \mathbf{x}_1$ and $k_{G+1} \equiv 1$). The expression for I_{gj} can be rewritten as

$$I_{gj} = (k_{g+1} - k_g) \int_0^1 \exp\left\{ \frac{-1}{2\sigma^2} \|\mathbf{y}_j - T(\mathbf{x}_g + (\mathbf{x}_{g+1} - \mathbf{x}_g)t)\|^2 \right\} dt, \tag{5.24}$$

and after some manipulations we find that

$$I_{gj} = \frac{A_{gj}\sqrt{\pi B_g}}{2} \left[\operatorname{erf}\left(\frac{1}{\sqrt{B_g}}(1 - \lambda_{gj}) \right) - \operatorname{erf}\left(\frac{-\lambda_{gj}}{\sqrt{B_g}} \right) \right], \tag{5.25}$$

where

$$\operatorname{erf}(z) \equiv \frac{2}{\sqrt{\pi}} \int_0^z e^{-z^2} dz \quad \text{(known as the error function)} \tag{5.26}$$

$$\lambda_{gj} \equiv \frac{|\mathbf{y}_j - T\mathbf{x}_g|}{|T\mathbf{x}_{g+1} - T\mathbf{x}_g|} \cos\left(\angle(T\mathbf{x}_{g+1} - T\mathbf{x}_g) - \angle(\mathbf{y}_j - T\mathbf{x}_g) \right) \tag{5.27}$$

$$A_{gj} \equiv (k_{g+1} - k_g) \exp\left\{ \frac{-|T\mathbf{x}_g + (T\mathbf{x}_{g+1} - T\mathbf{x}_g)\lambda_{jg} - \mathbf{y}_j|^2}{2\sigma^2} | \right\} \tag{5.28}$$

$$B_g \equiv \frac{2\sigma^2}{(s(k_{g+1} - k_g))^2} \tag{5.29}$$

and $\angle(\mathbf{x}, \mathbf{y})$ denotes the angle between $\mathbf{x}$ and $\mathbf{y}$.[10] Given eqs.(5.25)-(5.29), we can evaluate I_{gj} and hence, evaluate $p(\mathbf{y}_j | v = 1)$ in eq.(5.23).

The second difficulty associated with using a continuous mixture model is that we need to solve eq.(5.7) in the continuous setting:

$$(s, \Gamma, \mathbf{c}) = \arg\min_{s,\Gamma,\mathbf{c}} \sum_j \int_0^1 p(k, v = 1 | \mathbf{y}_j) \|\mathbf{y}_j - s\Gamma\mathbf{r}(k) - \mathbf{c}\|^2 dk. \tag{5.30}$$

The method of solution is analogous to the point-based case (Section 2.4) and all integrals can be evaluated using the segment-wise approach described above.

5.4.1 Examples

Fig. 5.10 shows how the PLM model performs on the matching tasks considered in Fig. 5.1. The data shapes are the same, but the generating shapes are polygons. Note that the number of vertices is much smaller than the number of points used in Fig. 5.1 – here we have used a simple algorithm based on the change in curvature to select the vertices. Fig. 5.11 demonstrates how PLM can be used to find a template shape in a real image.

[10] Eq.(5.25) is not the simplest 'erf-based' expression for evaluating eq.(5.24), but it avoids numerical issues associated with evaluating the error function.

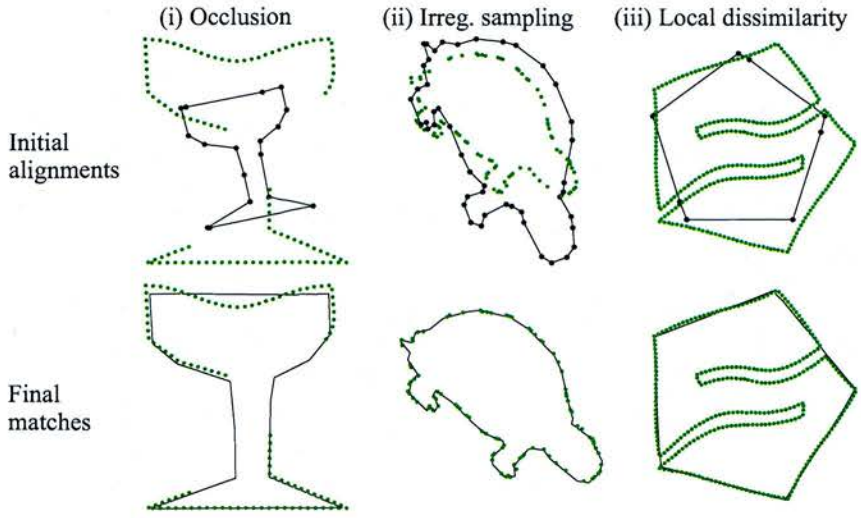

Figure 5.10: Examples – Probabilistic Line Matching (PLM).

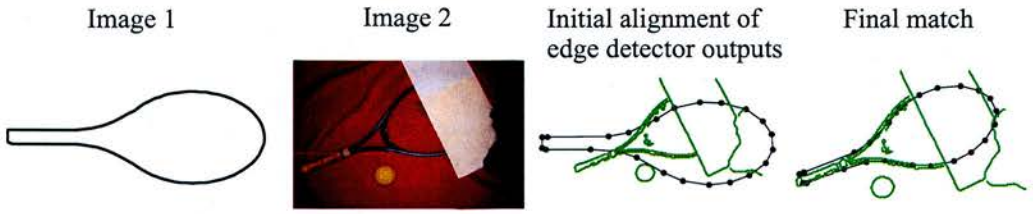

Figure 5.11: Using PLM to find a template shape in a real image.

5.4.2 Evaluation

The PPM algorithm described in Section 5.2 operates on a pair of unordered points sets, whereas PLM requires that one of the point sets is ordered and the HPM algorithm described in Chapter 4 requires that both of the point sets are ordered. Thus, PPM and PLM are more widely applicable than HPM and other dedicated contour matching algorithms yet are at a theoretical disadvantage to such algorithms when matching contours since they are unable to utilize the available ordering information. With this in mind, it is interesting to see how PPM and PLM perform on the MPEG-7 bullseye test considered in Sections 3.3 and 4.3.1.

The 100 point shape representations from Sections 3.3 and 4.3.1 were used for the bullseye tests and to increase the probability of PPM/PLM finding the correct match (*i.e.* to prevent the EM algorithm from getting trapped in a local minimum), the following

procedure was used to initialize the transformation parameters: $\mathbf{X}$ and $\mathbf{Y}$ were centered, normalized for size (where the size of $\mathbf{X}$ was defined as $\sum_k \|\mathbf{x}_k\|^2$), their principal components were aligned and then a range of transformed versions of $\mathbf{Y}$ were considered. We identified the transformed $\mathbf{Y}$ that was most similar to $\mathbf{X}$ based on a hard assignment of points and the associated transformation parameters were taken as the initial parameter values for PPM/PLM. Note that no ordering information was used in this initialization. To apply PLM, the vertices of the polygon $\mathbf{r}(k)$ were chosen from the 100 boundary points using a fast heuristic. The same initial $p(v)$ and σ were used for PPM and PLM, and 16 iterations of EM were used in both cases.

We found the data likelihood to be an unreliable measure of shape similarity and instead used the following heuristic score function to rank the shapes for retrieval. Each foreground $\mathbf{y}_j$ (*i.e.* $p(v=1|\mathbf{y}_j) > p(v=0|\mathbf{y}_j)$) was assigned to an $\mathbf{x}_k$ using $p(k|\mathbf{y}_j, v=1)$. The similarity score $\psi_{\mathbf{X}}(\mathbf{Y})$ was defined as the number of $\mathbf{x}_k$ that had at least one $\mathbf{y}_j$ assigned to them. This value is dependent on the number of outliers (if q of the $\mathbf{y}_j$ are explained by $v=0$, the score cannot be greater than 100-q) and how uniformly spread across the $\mathbf{x}_k$ the non-outlying $\mathbf{y}_j$ are. When using PLM, we substituted the polygon for the original 100 point shape after matching and computed $\psi_{\mathbf{X}}(\mathbf{Y})$ as above. The symmetric score function $\Psi(\mathbf{X},\mathbf{Y}) \equiv \psi_{\mathbf{X}}(\mathbf{Y}) + \psi_{\mathbf{Y}}(\mathbf{X})$ was used for retrieval.

PPM scored 81.98% on the bullseye test and PLM scored 81.88%. This compares favorably with other algorithms that match unordered point sets: [Tu & Yuille, 2004]-80.03%,[11] [Grigorescu & Petkov, 2003]-78.38%, [Belongie et al., 2002]-76.51%. It also compares favorably with many of the other algorithms listed in Table 4.1, all of which operate exclusively on ordered point sets. We note that the similarity score defined above works under the implicit assumption that occlusion is not present and also, that the boundaries have been sampled uniformly (so boundary information has been utilized to some extent). However, the key point is that the underlying matching algorithms remain applicable and effective when no such assumptions are valid (Figs. 5.1, 5.2, 5.10, 5.11).

Despite PLM using a continuous shape representation for the generating shape, the PPM and PLM algorithms are comparable in terms of speed. For 100 point shapes, it takes $\sim$0.22s to match shapes using PPM and 0.07-0.83s using PLM – the time varies

[11] The features used by Tu and Yuille (2004) *did* incorporate ordering information, but their algorithm is also applicable to unordered point sets.

depending on the number of vertices required for a reasonable polygonal approximation to the shape. Times are for a 1.4GHz Pentium-M machine using Matlab code.

5.5 Matching Ordered Point Sets

A probabilistic approach to matching two ordered point sets could use a *discrete* or a *continuous* approach. These alternative possibilities are summarized below; a detailed treatment of the continuous approach is given in Chapter 7 in the context of learning class models.

Discrete Approach: Hidden Markov Model

The PPM model in Section 5.2 is a mixture model and uses the standard assumption that data points are generated i.i.d. Thus, knowing which mixture component generated the n-th data point provides no information as to which component generated the $(n+1)$-th data point. However, in many real world problems, the temporal or sequential ordering of the data is important and the i.i.d. assumption is not suitable. A common approach in such cases is to use a *Hidden Markov Model (HMM)* which uses a *transition matrix* to explicitly model the probability of choosing a mixture component at the current time step given the component at the previous time step.

The switch from i.i.d. data to sequential data described above is essentially what occurs when switching from unordered point sets to ordered point sets. For example, consider the problem of matching two ordered point sets each composed of 100 equally spaced points. If the 20th component (*i.e.* point) of the generating shape is known to be responsible for the 75th point of the data shape, then there is a high probability that say, the 20th, 21st, 22nd or 23rd component is responsible for the 76th point – and note that any of these choices will preserve the monotonicity of the correspondence (Chapter 3). Given this sequential interpretation of ordered point sets, it is natural to use a HMM to match contours rather than the non-sequential mixture model of PPM. In this setting, the structure of the transition matrix will enforce the monotonicity constraints on the correspondence. Specifically, a *circular HMM* [Arica & Yarman-Vural, 1999] could be used for closed shapes and a *left-to-right HMM* [Bishop, 2006] for open shapes. Just as the EM updates for PPM are similar to those for a standard mixture model, the EM updates for matching contours would be similar to those for a standard HMM (*e.g.* Bishop

(2006)).[12] In a shape matching/retreival scenario where only one data shape/sequence is considered, the transition matrix should either be user-specified (or at least have a strong prior) or learnt from training data where a common transition matrix is used for all classes. This contrasts with classification tasks where each class model would have its own transition matrix.

Previous HMM-based approaches to shape matching and classification have involved chain codes [Arica & Yarman-Vural, 1999, Bicego & Murino, 2001] or curvature [Abd-Almageed & Smith, 2002, Bicego & Murino, 2004]. In the approach proposed by Bicego and Murino (2004), the hidden states are effectively curvature values and the transition matrix contains the likelihoods of jumping from one curvature value to another as a shape contour is traversed. Both this and the chain code based approaches are likely to suffer from the same problems as other 'purely local approaches' – see Section 4.1.

Continuous Approach: Curve Reparameterization

Even if a contour is represented by a finite number of points (or only a finite number of points are observed), the 'real shape' is the underlying *continuous* contour upon which these points lie. Thus, we may prefer to use a continuous approach to matching contours whereby the shapes are described by functions, *e.g.* Fourier expansions or splines. Let us assume that we have two shapes represented by the continuous functions $f_1(t)$ and $f_2(t)$, where t is the arc-length parameter. The monotonic relabeling used for ordered point sets (see previous section and Chapter 3) is replaced by *monotonic curve reparameterization* in this continuous setting. For example, $f_2(t) \rightarrow f_2(g(t))$, where $g(t)$ is a monotonic function chosen such that $f_2(g(t))$ resembles $f_1(t)$ as closely as possible. Conceptually, one can think of traversing curve f_1 at constant speed and f_2 at non-uniform speed such that at any given time, the points reached on the two curves are as close together as possible – with respect to some global alignment of the curves which must be estimated.

In this thesis, we will investigate the continuous approach. However, while the material in Sections 5.3 and 5.4 are naturally seen as extensions to PPM, our probabilistic approach to matching ordered contours is more easily seen as a special case of the "probabilistic contour model" described in Chapter 7. Accordingly, we do not include details here and direct the reader to Chapter 7 and Section 7.6.3 in particular.

[12] The EM algorithm is often called the *Baum-Welch algorithm* for the specific case of a HMM.

5.6 Discussion

In this chapter, we have reviewed an effective probabilistic approach to matching unordered point sets [Chui & Rangarajan, 2000a, Luo & Hancock, 2002, Taylor et al., 2003, Kent et al., 2004] and shown how it can be modified to handle various other matching problems. The material on part-based shapes is novel; the algorithm for matching a line set to a point set (Section 5.4) was briefly discussed by Krishnapuram et al. (2004), though few details were given and no examples shown. Much of the relevant work on matching ordered point sets has focused on the problem of learning class models and is discussed in Chapter 7. Other approaches to matching ordered point sets are discussed in Chapters 3 and 4.

Chapter 6

Class Models for Unordered Point Sets

A single shape contains no information about the *variation* that is present in the shape class to which it belongs. Indeed, this was precisely the problem encountered in the pairwise retrieval problems considered in the previous chapters where we were given a single query shape and asked to decide which database shapes were similar to it. In the absence of any information telling us how to compute shape similarity *for that particular query shape*, we were forced to construct a generic similarity measure to use in all cases. However, if we had been given multiple query shapes from the same class, we could have estimated the within-class variation and used this to improve retrieval accuracy. For example, given the contours of seven fish that belong to the same species, we might find that the shape of the dorsal fin is highly variable but the shape of the tail changes little. Based on this finding, we might weight the tail similarity more heavily in the similarity score.

More commonly, multiple training shapes arise in data visualization problems (*e.g.* a biologist wishes to understand the type and extent of variation found in a certain species of fish) and classification tasks (*e.g.* a biologist wishes to know which species of fish a 'test' contour belongs to). In all such problems and regardless of whether the shapes are ordered or unordered, the fundamental challenge is to extract and utilize the within-class variation. In the next two chapters we interpret this challenge in the context of learning class models. In this framework, the groupwise analysis problem can be seen as an extension of the probabilistic approaches to pairwise matching covered in the previous chapter: the data shapes (now there are more than one) are still generated by a generating shape, the correspondence problem(s) must be solved and the alignment transforma-

tion(s) must be estimated. However, when learning a class model, the generating shape is neither known nor fixed and hence, we estimate additional model parameters which specify a mean generating shape and the ways in which this can deform to produce the data shapes. In this chapter, we present a novel approach to learning a class model from unordered point sets; the next chapter focuses on class models of ordered point sets.

6.1 Introduction

We are interested in learning a shape model from unordered point sets which can be of different sizes and where the correspondence is unknown. For example, in chemoinformatics the aim might be to investigate the variation in a set of molecules from the same family (Sections 6.3 and 6.5). Dryden et al. (2007) have recently proposed a technique for matching multiple shapes simultaneously using a 'mean shape plus noise' generative model. Only linear alignment transformations are considered and the transformations themselves are optimized over rather than modeled. Consequently, their approach is suited to data sets containing little systematic variation in shape beyond the presence/absence of individual points. Wang et al. (2006) have proposed a technique that uses nonlinear transformations to match shapes to an emerging mean and which accounts for the possibility that there may not exist an exact pointwise correspondence between training shapes.[1] Despite the elegant probabilistic formulation and the use of nonlinear transformations, their approach shares an important property with that of Dryden et al. (2007): only a mean shape emerges during learning rather than a more expressive model of shape variation. Since it is the *systematic variation* in a shape data set that is typically of interest, this is a significant limitation. To address this issue, we present a low-dimensional latent variable model able to describe complex shape variation.

The approach taken in this chapter can be seen as extending the *pairwise* PPM algorithm of Section 5.2. On the other hand, it is related to existing techniques for modeling *multiple* point sets where the correspondence between training shapes is known *a priori*. The point distribution model [Cootes et al., 1992] is perhaps the best known of these methods; this involves aligning the shapes using Procrustes analysis then applying principal component analysis (PCA). Of course, here the correspondences are *not* known and

[1] This occurs when matching points from edges in images, but not in the chemoinformatics problems in Sections 6.3 and 6.5.

also, the model may need to be nonlinear in order to explain complex shape variation. The *probabilistic shape model* (PSM) introduced in this chapter is a generative probabilistic model that can be linear or nonlinear and which explicitly accounts for noise in the data. Just as in PPM, the soft correspondences which naturally arise in a probabilistic setting help to avoid local minima during parameter estimation. However, in PSM the EM algorithm finds a soft *groupwise correspondence* during the E-step and estimates both the alignment transformations and the *model parameters* during the M-step.

The PSM is related to the work of Williams, Revow and Hinton on digit recognition [Williams, 1994, Revow et al., 1996] which involves learning a linear model over the control points of a spline. In their work, ordering information about the 'inked pixels' is unknown (or not used) so a digit is treated as an unordered point set. Here, we are interested in linear and nonlinear models for 'genuinely unordered point sets', *i.e.* the points do not lie along a curve and there is no correct ordering, either known or unknown. We now describe the PSM in detail.

6.2 Probabilistic Shape Model (PSM)

Assume that we have observed N unordered point sets $\mathbf{Y}^1, \ldots, \mathbf{Y}^N$, of size $J^1, \ldots, J^N$ respectively. We will use the notation $\mathbf{y}_j^n \in \mathbb{R}^M$ to refer to point j of shape n, where M is the number of dimensions – typically 2 or 3. Note that superscript is used to index shapes (and points in latent space – see below) and subscript to index points. Our first task is to specify the model that generated these shapes.

6.2.1 The Model

For a single shape $\mathbf{Y}^n$, the generative model can be summarized in the following steps

1. Sample a point $\mathbf{z}$ from a low-dimensional latent shape space.
2. Map $\mathbf{z}$ to a K-point shape, where typically K is set to $\max\{J^1, \ldots, J^N\}$ as in Dryden et al. (2007).
3. Generate the data points $\mathbf{y}_1^n, \ldots, \mathbf{y}_{J^n}^n$ of $\mathbf{Y}^n$ from a GMM with centers at the K-points of the 'generating shape' constructed in step 2.

Steps 1 and 2 represent the shape model. For example, sampling $\mathbf{z}$ from a Gaussian prior over latent space and using a *linear* mapping in step 2 gives a probabilistic principal component analysis (PPCA, [Tipping & Bishop, 1999]) based shape model. However, the lower level mixture model in step 3 and the possibility that the mapping in step 2 may be nonlinear makes parameter estimation in our model more difficult than in standard PPCA. Specifically, the integrals over the latent variables needed to compute $p(\mathbf{Y}^n)$ and the M-step updates are intractable. To circumvent this problem, we adopt the approach taken by Bishop, Svensen and Williams in their work on the Generative Topographic Mapping (GTM) [Bishop et al., 1998] (essentially a nonlinear version of PPCA) and use a discrete rather than a continuous latent space. For an L-dimensional latent space, this involves discretizing the hypercube $[-1,1]^L$ using a uniform grid of G points, $\mathbf{z}_1,\ldots,\mathbf{z}_G$, and assuming a uniform prior over these points:

$$p(\mathbf{z}) = \frac{1}{G}\sum_{g=1}^{G}\delta(\mathbf{z}-\mathbf{z}_g). \tag{6.1}$$

With this discrete latent space, the problematic integrals discussed above become sums which are easily computed. Note that each grid point in latent space is mapped to a generating shape via the mapping in step 2. Thus, there are now finitely many (G) generating shapes and our model has become a hierarchical mixture model:

Higher level mixture model for shapes The data shapes $\mathbf{Y}^1,\ldots,\mathbf{Y}^N$ are sampled i.i.d. from a G component mixture model.

Lower level mixture model for points Given a generating shape, the data points $\mathbf{y}_1^n,\ldots,\mathbf{y}_{J^n}^n$ are generated as described in step 3 above – though see the restrictions on the latent variables discussed in Section 6.2.2.

The log-likelihood of the data given this hierarchical model is

$$\ln p(\mathbf{Y}^1,\ldots,\mathbf{Y}^N) = \ln\left\{\frac{1}{G}\prod_{n=1}^{N}\sum_{g=1}^{G}\prod_{j=1}^{J^n}\sum_{k=1}^{K+1}p(\mathbf{y}_j^n|k,g)p(k|g)\right\}, \tag{6.2}$$

Note that the product over j (*i.e.* over points from the same data shape) is inside the sum over g. This captures the idea that the same generating shape is responsible for all points from the same data shape; we are unlikely to learn a sensible model of shape variation if this is not enforced.

The conditional distribution of a point is assumed to be

$$\mathbf{y}_j^n|k,g \sim \mathcal{N}(T^n(\mathbf{W}_k\phi(\mathbf{z}^g)+\mu_k),\sigma^2\mathbf{I}_M), \quad k=1,\ldots,K \tag{6.3}$$

$$\mathbf{y}_j^n|k,g \sim \text{Uniform}, \quad k=K+1 \tag{6.4}$$

where $\phi(\cdot):\mathbb{R}^L\rightarrow\mathbb{R}^U$ is a nonlinear mapping (see below), $\mathbf{W}_k\in\mathbb{R}^{M\times U}$ and $\mu_k\in\mathbb{R}^M$ is the k-th point of the mean generating shape. Finally, T^n is a similarity transformation: $T^n\mathbf{x}\equiv\beta^n\Gamma^n\mathbf{x}+\gamma^n$, where β^n is a scale parameter, $\Gamma^n\in\mathbb{R}^{M\times M}$ is a rotation matrix and $\gamma^n\in\mathbb{R}^M$ is a translation vector. Note that stacking the matrices $\mathbf{W}_1,\ldots,\mathbf{W}_K$ into a single $MK\times U$ matrix would give a linear mapping from (transformed) latent space to (MK-dimensional) data space analogous to the mapping found in PPCA, factor analysis or GTM.

The only distribution still to be defined is $p(k|g)$ – the probability of choosing point $k\in\{1,\ldots,K\}$ of generating shape g or the outlier component $k=K+1$. We start by fixing the probability of the outlier component to be small (with respect to the number of non-outlying points/components, K) and make it independent of the generating shape:

$$p(K+1|g)=p(K+1)=1/10K. \tag{6.5}$$

The only other constraints on the G-by-(K+1) conditional probability table for $p(k|g)$ are that each entry is in $[0,1-p(K+1)]$ for $k\neq K+1$ and that $\sum_{k=1}^{K+1}p(k|g)=1$, *i.e.* the distribution is not parameterized. Note that $p(k|g)$ indicates the likely presence or absence of an individual point as a function of shape – *e.g.* the presence/absence of a specific atom as function of molecular shape. Having considered all the distributions, the next step is to estimate the model parameters.

6.2.2 ECM Algorithm for Parameter Estimation

As in Chapter 5, we turn to the EM algorithm to estimate the model parameters. Unfortunately, even with the introduction of a discrete latent space, maximizing the expected complete log-likelihood with respect to all the parameters simultaneously is intractable for the model introduced above. To avoid this problem, we use the Expectation Conditional Maximization (ECM) algorithm [Meng & Rubin, 1993] whereby, at each step, a subset of the parameters are updated whilst the others remain fixed.

The first step towards deriving the ECM algorithm is to introduce latent variables for both the high level shape model and the low level point model:

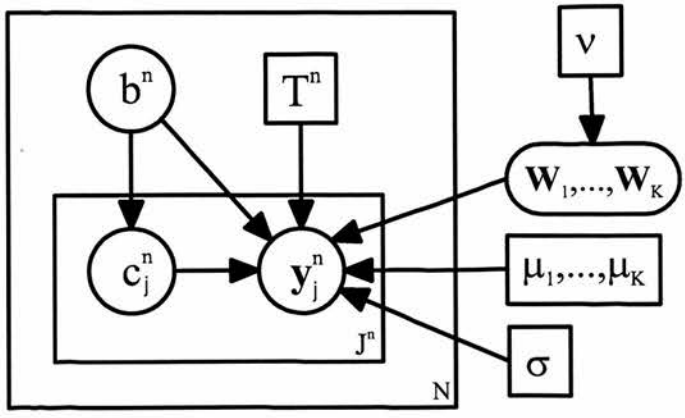

Figure 6.1: The probabilistic shape model (PSM).

- $b^n \in \{1,\ldots,G\}$ indicates which generating shape is responsible for the data shape $\mathbf{Y}^n$.
- $c_j^n \in \{1,\ldots,K,K+1\}$ indicates which point of the generating shape is responsible for the data point $\mathbf{y}_j^n$.

Note that there is only a single c_j^n per data point which gives the pointwise correspondence between the data point and the model, *i.e.* we do not match a data shape to each generating shape independently as might occur under a naive combination of a shape model (steps 1 and 2 in Section 6.2.1) and the PPM correspondence algorithm described in Section 5.2.

The graphical model for PSM is shown in Fig. 6.1 where 'plate notation' [Bishop, 2006] has been used to denote i.i.d. variables. Readers not familiar with graphical model representations should simply note that the diagram represents the conditional independence assumptions that have been made when constructing the model and that as a consequence of these assumptions, we need only specify the conditional distribution of each variable given its parents in the graph.[2] The required distributions were all defined in the previous section, though note that the variables g and k are now replaced by the latent variables b^n and c_j^n respectively.

The complete log-likelihood of all the variables is given by

$$\mathcal{L}_c = N \ln \frac{1}{G} + \sum_{n=1}^{N} \sum_{j=1}^{J^n} \ln p(\mathbf{y}_j^n, c_j^n | b^n) \tag{6.6}$$

[2]See Bishop (2006) for an introduction to graphical models.

and it is easy to show that the expectation (taken with respect to the distribution of the latent variables given the data) of $\mathcal{L}_c$ is

$$\mathcal{E}_c = \sum_{n=1}^{N} \sum_{j=1}^{J^n} \sum_{g=1}^{G} \sum_{k=1}^{K+1} p^{old}(g|\mathbf{Y}^n) p^{old}(k|\mathbf{y}_j^n, g) \left(\ln p(\mathbf{y}_j^n|k,g) + \ln p(k|g) \right), \tag{6.7}$$

where we have ignored the constant term in eq.(6.6) and used the notation p^{old} to indicate distributions evaluated using the old parameter values. The term $p^{old}(k|\mathbf{y}_j^n, g)$ gives the strength of correspondence between point $\mathbf{y}_j^n$ and point k on generating shape g. However, the higher level shape-to-shape correspondence, $p^{old}(g|\mathbf{Y}^n)$, ensures that the model prefers to explain all points from the same data shape using the same generating shape as discussed above.

The ECM algorithm for parameter estimation is given in Table 6.1. The expressions for the parameter updates in eqs.(6.10)-(6.13) are found by maximizing $\mathcal{E}_c$ with respect to the specified subset of parameters while holding the other parameters fixed. We currently set the value of σ^2 by hand but this could also be estimated from the data if desired. In our experiments, we either learn a linear model by setting $\phi(\mathbf{z}) = \mathbf{z}$ or a nonlinear model by setting $\phi(\mathbf{z}) = (\phi_1(\mathbf{z}), \ldots, \phi_U(\mathbf{z}))^T$, where the $\phi_u(\cdot)$ are Gaussian shaped radial basis functions (RBFs) with centers positioned on a uniform grid in $[-1,1]^L$ and a common width parameter chosen so that the centers of neighboring basis functions are two standard deviations apart. This is the same type of nonlinear mapping that is used in GTM [Bishop et al., 1998] and as in that work, we regularize it by placing a Gaussian prior with variance ν^2 over the $\mathbf{W}_k$:

$$p(\mathbf{W}_k|\nu^2) = \left(\frac{1}{2\pi\nu^2} \right)^{MU/2} \exp\left\{ -\frac{1}{2\nu^2} \sum_{u=1}^{U} \sum_{m=1}^{M} [\mathbf{W}_k]_{mu}^2 \right\}, \tag{6.14}$$

and then taking the maximum *a posteriori* (MAP) estimate of $\mathbf{W}_k$ in the M-step (eq.(6.11)). The impact of this prior can be seen in eq.(6.11), where the $\mathbf{W}_k$ update is a regularized least squares estimate with regularization parameter σ^2/ν^2 – the ratio of the noise variance to the prior variance.

6.2.3 Implementation

In all experiments, we initialize the entries of the $\mathbf{W}_k$ to zero and set the mean generating shape equal to the data shape with the greatest number of points; this implicitly

Table 6.1: ECM algorithm for learning a Probabilistic Shape Model (PSM).

The E-step is repeated after each CM-step. Sums and products over j are from 1 to J^n and, unless otherwise stated, sums over k are from 1 to K+1. For readability, we define the quantities

$$H \equiv \sum_{n=1}^{N} J^n = \text{ total number of data points,} \qquad \Omega^n_{jgk} \equiv p^{old}(g|\mathbf{Y}^n)p^{old}(k|\mathbf{y}^n_j,g)$$

E-step Evaluate the posteriors using the current parameter values and eqs.(6.3) and (6.4):

$$p^{old}(k|\mathbf{y}^n_j,g) = \frac{p(\mathbf{y}^n_j|k,g)p(k|g)}{\sum_k p(\mathbf{y}^n_j|k,g)p(k|g)} \tag{6.8}$$

$$p^{old}(g|\mathbf{Y}^n) = \frac{\prod_j \sum_k p(\mathbf{y}^n_j|k,g)p(k|g)}{\sum_g \prod_j \sum_k p(\mathbf{y}^n_j|k,g)p(k|g)} \tag{6.9}$$

CM-step 1

Probability of choosing generating point k given generating shape g:

$$p(k|g) = (1-p(K+1))\frac{\sum_{n,j}\Omega^n_{jgk}}{\sum_{n,j}\sum_{k=1}^{K}\Omega^n_{jgk}}, \qquad k=1,\ldots,K,\ g=1,\ldots,G. \tag{6.10}$$

Mapping from the (transformed) latent space onto point k of the associated generating shape (see eq.(6.3)):

$$\mathbf{W}_k^T = \left(\mathbf{R}^T\mathbf{R} + \frac{\sigma^2}{\nu^2}\mathbf{I}_U\right)^{-1}\mathbf{R}^T\mathbf{S}, \qquad k=1,\ldots,K \tag{6.11}$$

where $\mathbf{R} \in \mathbb{R}^{GH\times U}$ with rows $(\Omega^n_{jgk})^{1/2}\beta^n\phi(\mathbf{z}^g)^T$ and $\mathbf{S} \in \mathbb{R}^{GH\times M}$ with rows $((\Omega^n_{jgk})^{1/2}(\Gamma^n)^T(\mathbf{y}^n_j - \gamma^n - \beta^n\Gamma^n\mu_k))^T$.

CM-step 2

Point k of the mean generating shape:

$$\mu_k = \frac{\sum_{n,j,g}\Omega^n_{jgk}\beta^n(\Gamma^n)^T(\mathbf{y}^n_j - \gamma^n - \beta^n\Gamma^n\mathbf{W}_k\phi(\mathbf{z}^g))}{\sum_{n,j,g}\Omega^n_{jgk}(\beta^n)^2}, \qquad k=1,\ldots,K. \tag{6.12}$$

CM-step 3

Rotation, scale and translation parameters for matching the generating shape to the n-th data shape:

$$\Gamma^n = \mathbf{V}\mathbf{U}^T, \quad \beta^n = \frac{Tr(\mathbf{X}^T\mathbf{Y}(\Gamma^n)^T)}{Tr(\mathbf{Y}^T\mathbf{Y})}, \quad \gamma^n = \overline{\mathbf{X}}^T - \beta^n\Gamma^n\overline{\mathbf{Y}}^T, \quad n=1,\ldots,N \tag{6.13}$$

where $\mathbf{X} \in \mathbb{R}^{GKJ^n\times M}$ with rows $(\Omega^n_{jgk})^{1/2}(\mathbf{y}^n_j)^T$ and $\mathbf{Y} \in \mathbb{R}^{GKJ^n\times M}$ with rows $(\Omega^n_{jgk})^{1/2}(\mathbf{W}_k\phi(\mathbf{z}^g)+\mu_k)^T$ and the columns of both $\mathbf{X}$ and $\mathbf{Y}$ have been centered. The matrices $\mathbf{U}$ and $\mathbf{V}$ are from the singular value decomposition: $\frac{\mathbf{X}^T\mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|} = \mathbf{V}\Lambda\mathbf{U}^T$, and the row vectors $\overline{\mathbf{X}}, \overline{\mathbf{Y}} \in \mathbb{R}^M$ are the column means of the uncentered $\mathbf{X}$ and $\mathbf{Y}$.

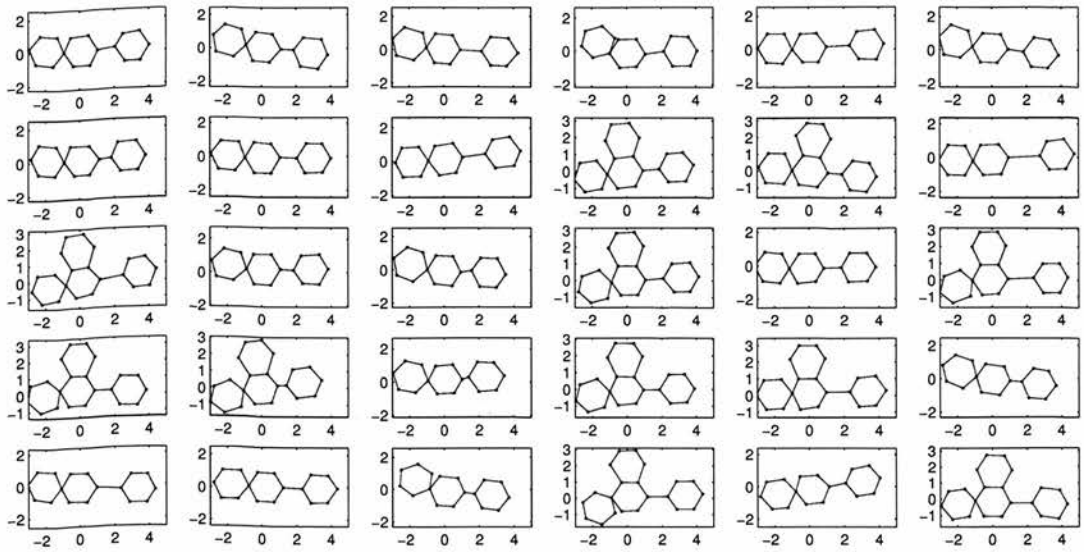

Figure 6.2: The artificial molecules data set (N=30).

sets $K = \max\{J^1, \ldots, J^N\}$. The first E-step provides approximate point-wise correspondences ($p(k|\mathbf{y}_j^n, g)$), however, these are equal for all g as a consequence of initializing the $\mathbf{W}_k$ to zero, so a modified $\mathbf{W}_k$ update is needed in the first iteration. Specifically, the $p(k|\mathbf{y}_j^n, g)$ are used to make a hard assignment of non-outlying data points to generating points and, for a linear model, PCA is used to 're-initialize' the $\mathbf{W}_k$ directly. In the nonlinear case, regression is used to initialize $\mathbf{W}_k$ so that the nonlinear mapping approximates the linear PCA mapping.

The number of grid points in latent space (G) should be large in order to accurately approximate a continuous latent space, but this must be balanced against the associated rise in computation time.

6.3 Artificial Example

Fig. 6.2 shows an artificial 2D molecules data set, where each molecule consists of either 17 or 21 atoms. We stress that these are simply molecule-like objects that are useful for illustrating the algorithm described in the previous section. There is no atom type or properties associated with the atoms and the 'bonds' are added purely to aid visualization. The molecules were constructed in the following way:

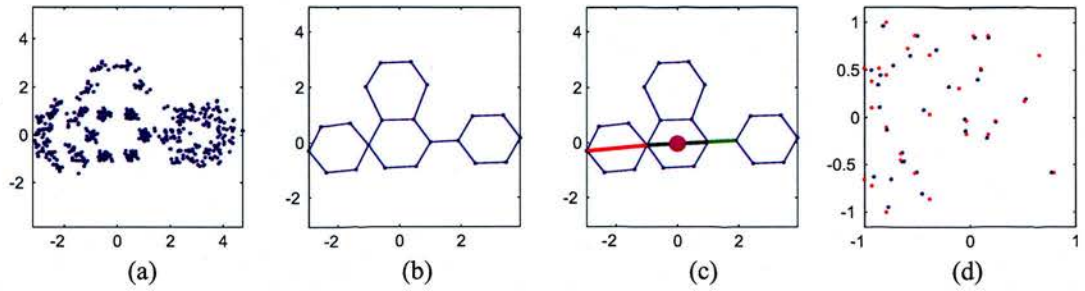

Figure 6.3: (a) Initial alignment of the point sets. (b) Learnt mean shape. (c) Learnt mean shape including the 3-line representation used in Fig. 6.5 (see text). (d) The posterior means (blue) and modes (red).

1. The central hexagon is common to all molecules.

2. The left hexagon rotates about the atom it shares with the central hexagon. The angle of rotation is randomly sampled from a zero-mean Gaussian with standard deviation 0.3.

3. The distance between the right hexagon and the central hexagon is randomly sampled from a Gaussian with mean 1 and standard deviation 0.3.

4. An upper hexagon is present if the rotation angle from step 2 is ≥ 0.

After constructing the molecules, independent Gaussian noise of standard deviation 0.03 is added to each point; this explains the slight irregularity of the hexagons in Fig. 6.2. The global alignment of the shapes resulting from the construction procedure described above is perturbed by applying a small random translation and rotation to each point set. The initial alignment of all shapes is shown in Fig. 6.3a; see that some of the 21 point clouds overlap. We note that in many applications, the initial alignment may be very poor and application specific heuristics will be required for initialization – see Section 6.5.

We applied the approach described in the previous section using a linear mapping, a two-dimensional latent space (L=2), a 30-by-30 grid (G=900) and $\sigma^2 = 0.05$ (this value of σ^2 was used in all experiments). Scale invariance is not desirable when considering molecules, so only translation and rotation were updated in CM-step 3 (Table 6.1). The learnt mean shape is shown in Fig. 6.3b. Shapes associated with randomly sampled

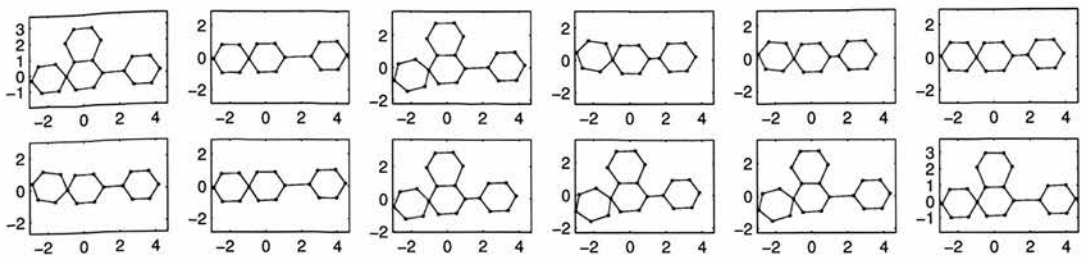

Figure 6.4: Shapes corresponding to randomly selected grid points in latent space.

grid points in latent space (*i.e.* generating shapes) are plotted in Fig. 6.4, where points with $p(k|g) < 1/10K$ have been removed from the figure. It is clear that the model has captured the variation known to be present in the data set and note that the upper hexagon is only present when the rotation of the left hexagon is anti-clockwise. Sampling from a continuous uniform prior over latent space (*i.e.* choosing non-grid points) produces similar results to those in Fig. 6.4 so examples are omitted.

In Fig. 6.5, generating shapes are plotted at the position of their corresponding grid point using the 3-line representation demonstrated in Fig. 6.3c. Note that the angle between the red and black lines indicates the rotation angle described in step 2 above and the length of the green line is the distance between the central and right hexagon (step 3). The magenta circle is only included when $p(k|g) > p(K+1)$ (the probability of choosing the outlier component) for all the upper four points of the top hexagon (step 4). We make the following observations about Fig. 6.5:

- The variation in the length of the green line corresponds to a principal component running (approximately) top-left to bottom-right in latent space.
- The angle between the red and black lines corresponds to a principal component along the other diagonal. Further, the same diagonal roughly captures the presence/absence behavior of the magenta circle as we would hope (since we know this has a deterministic dependence on the angle).
- The two systematic types of variation are represented by orthogonal directions in latent space consistent with the independence of steps 2 and 3 above.

This simple interpretation of Fig. 6.5 is possible due to the variation present in this particular data set and the fact that the latent space to data space mapping is linear.

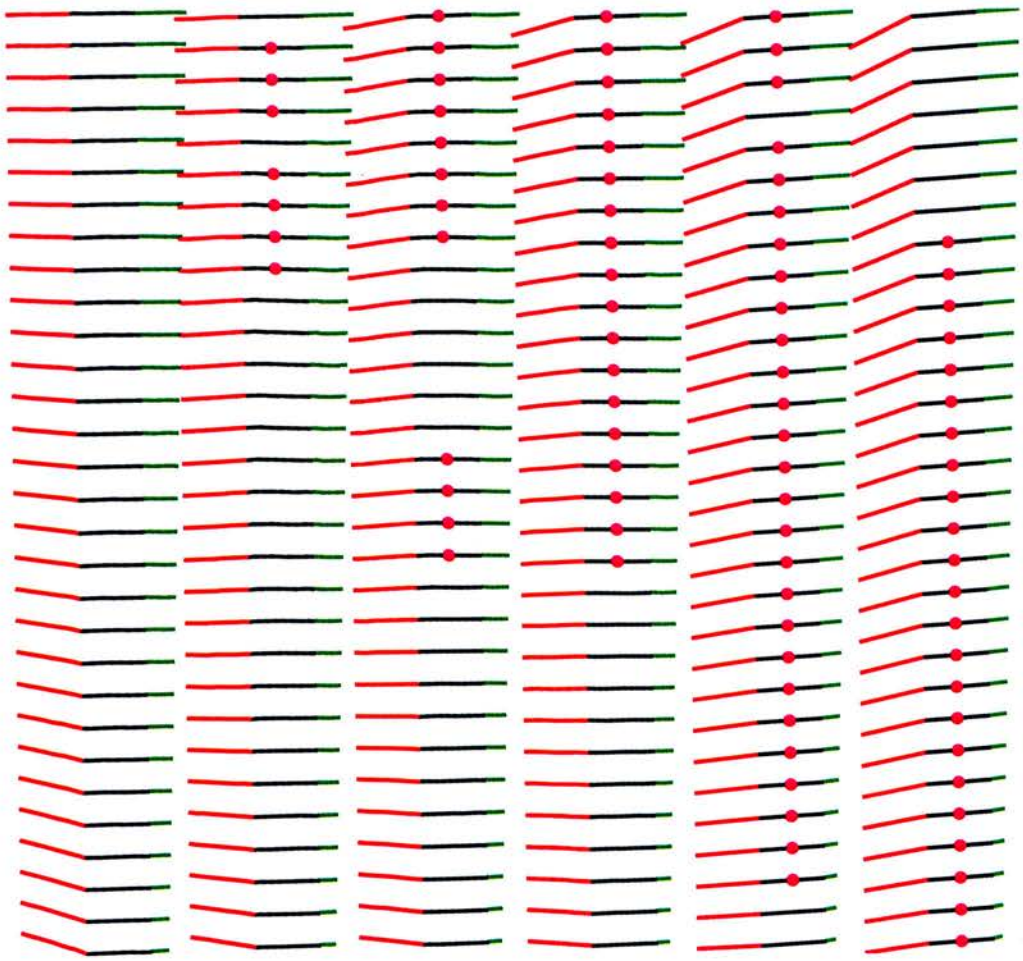

Figure 6.5: Each generating shape is shown at the position of the corresponding grid point in latent space; only one fifth of the shapes are plotted for clarity.

To appreciate this, imagine traversing a straight line in latent space and observing the movement that this induces in the K points of the generating shape. Since the mapping is linear, each point traces out a straight line in $\mathbb{R}^2$. For the artificial data set considered here, the points do indeed trace out roughly straight lines – the range of rotation for the left hexagon is small so the curved paths traced by the points can be accurately approximated by a straight line. Thus, assuming the learnt linear model is accurate, we will observe straight principal directions in latent space.

6.4 Nonlinear Example

We briefly consider a second, artificial data set (Fig. 6.6a) to demonstrate the importance of using a nonlinear model where appropriate. Each shape consists of only four points; all shapes have an identical triangular base, but the position of the highest point is sampled from a curve. There is only one mode of variation here, so a 1D shape model (*i.e.* a 1D latent space, $L = 1$) is appropriate. Two models are considered:

1. A **linear** 1D model with $G = 50$ grid points.
2. A **nonlinear** 1D model with $G = 50$ grid points, $U = 5$ RBFs for the nonlinear mapping and a prior variance (the regularization parameter) of $\nu^2 = 10^6$.

Let us label the 'moving point' of the generating shape $k = 1$ (this is arbitrary) and refer to the point itself as $\mathbf{x}_1$. The theoretical limitations of the above models are easily understood by considering the change in position of $\mathbf{x}_1 \in \mathbb{R}^2$ as we traverse the latent space – the discretized line between -1 and 1 with points indexed by $g = 1, \ldots, G$. In the linear model, $\mathbf{x}_1 = \mathbf{W}_1 \mathbf{z}^g + \mu_1$ (set $\phi(\mathbf{z}^g) = \mathbf{z}^g$ in eq.(6.3)), so the straight line latent space is mapped to a *straight line* in $\mathbb{R}^2$. This is demonstrated in Fig. 6.6b which shows the generating shapes for the linear model learnt using the data set in Fig. 6.6a. It is clear that this model is inadequate. In contrast, the (discretized) *curve* along which $\mathbf{x}_1$ moves for a nonlinear model is given by $\mathbf{x}_1 \equiv \mathbf{W}_k \phi(\mathbf{z}^g) + \mu_k$, $g = 1, \ldots, G$ – note the nonlinear transformation of the latent space point, $\phi(\mathbf{z}^g)$. Fig. 6.6c-e shows the generating shapes for the nonlinear model at various stages of learning. Though not perfect, the learnt model (Fig. 6.6e) is obviously more accurate than the linear model (Fig. 6.6b).

The purpose of this example is to demonstrate the difference between linear and nonlinear models in the simplest possible setting. The data is noise-free, the initial alignment of the shared triangular base is perfect and the Procrustes alignment step (CM-step 3 in Table 6.1) is omitted to make Fig. 6.6 as clear as possible.[3] In real-world problems, it will be more difficult to decide whether a nonlinear model is required. This issue, and the challenge of model selection for PSM in general, will be addressed in future work.

[3]The results when this step is included are similar but slightly messier due to the small alignment transformations. Generally, the correct alignment is not known and the alignment step is required.

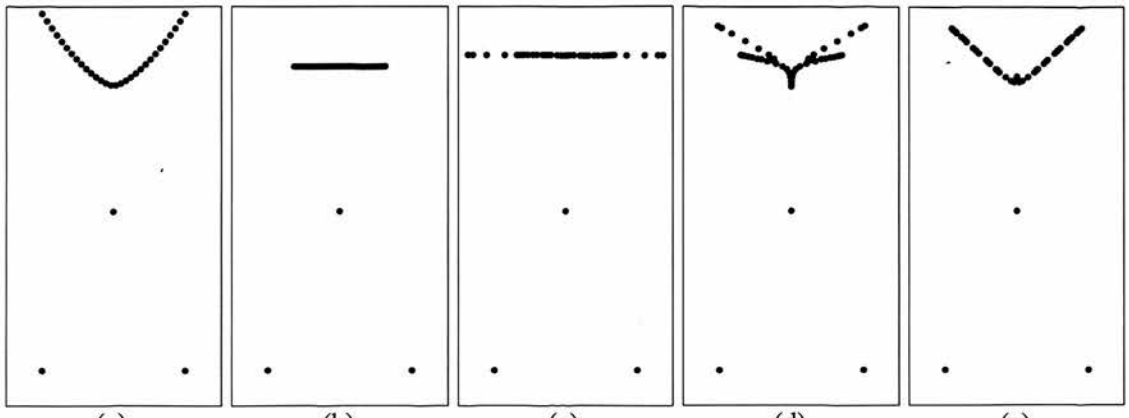

Figure 6.6: (a) The data set; all shapes share the same triangular base, but the position of the top point is variable. (b) The generating shapes of a *linear* 1D model. (c-e) The generating shapes of a *nonlinear* 1D model after 2, 15 and 150 iterations of the ECM algorithm.

6.5 Steroid Data

We now consider a steroids data set which has been used to test other modeling algorithms [Wagener et al., 1995, Coats, 1998, Dryden et al., 2007].[4] The objective in this section is not a detailed biochemical analysis. Rather, we wish to demonstrate how easily our approach can be applied to real chemoinformatics data in the hope that researchers in this area will apply it to their own data.

The data set contains information on 31 steroid molecules, including the 3D coordinates of each atom – Fig. 6.7. Each molecule contains 4 carbon rings which can be used to find a good initial alignment of the molecules [Dryden et al., 2007]. We first analyze the data using a linear shape model with a 1D latent space ($G = 50$). Fig. 6.8 shows the mean shape which is similar to its initial configuration – the steroid consisting of the maximal number of atoms: $n = 23$ (cortisolacetat), $J^{23} = K = 61$. The red lines at each point show the trajectory of the points as we move across latent space. The most noticeable change is in the leftmost carbon loop; the upward motion on one side and downward on the other results in this hexagon being folded away from the approximate plane upon which the other carbon loops lie (also see Fig. 6.10). The CH_3 group just to

[4] http://www2.ccc.uni-erlangen.de/services/steroids

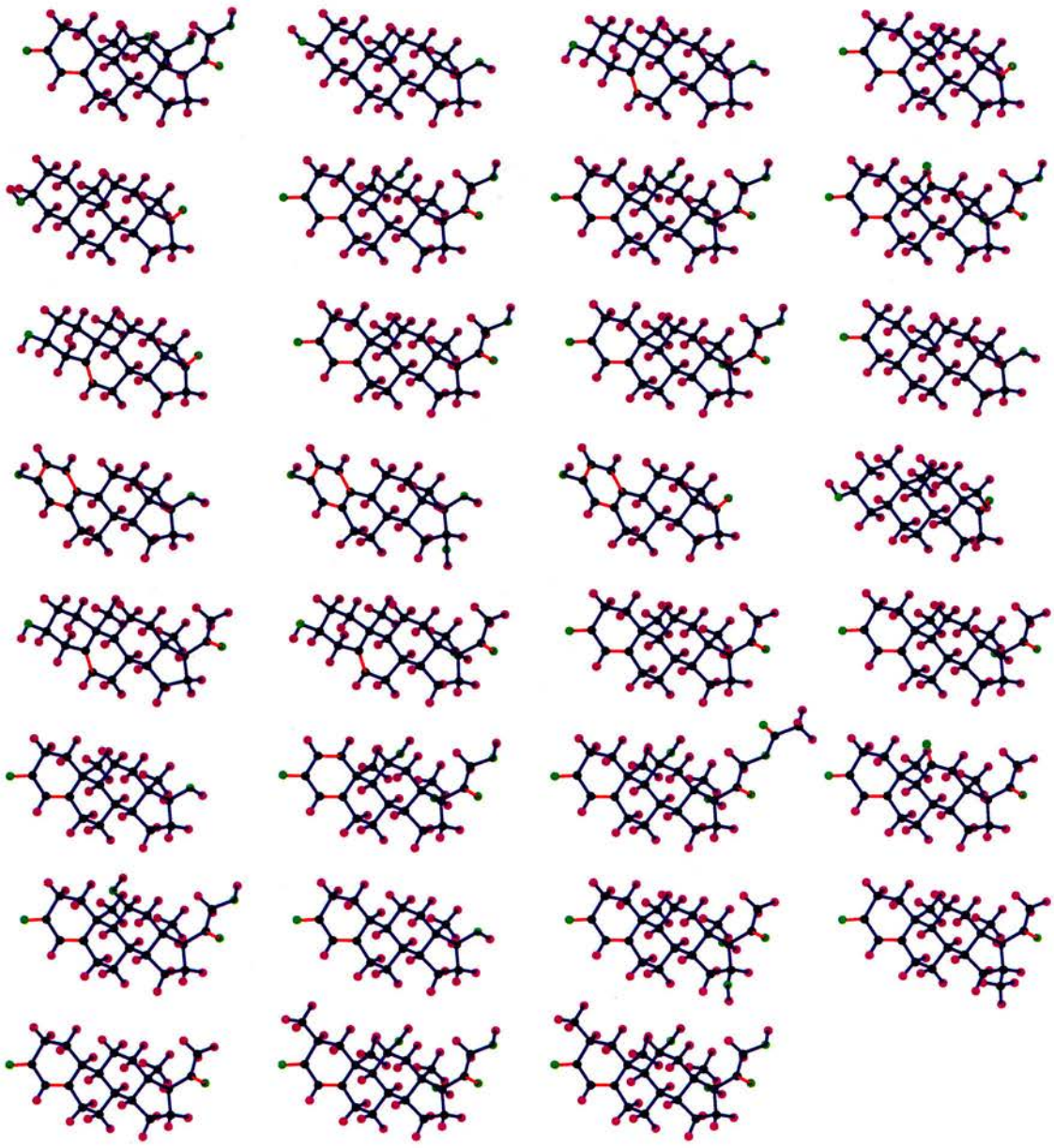

Figure 6.7: The steroid data set (N=31). Black – carbon, green – oxygen, magenta – hydrogen; double bonds are shown in red.

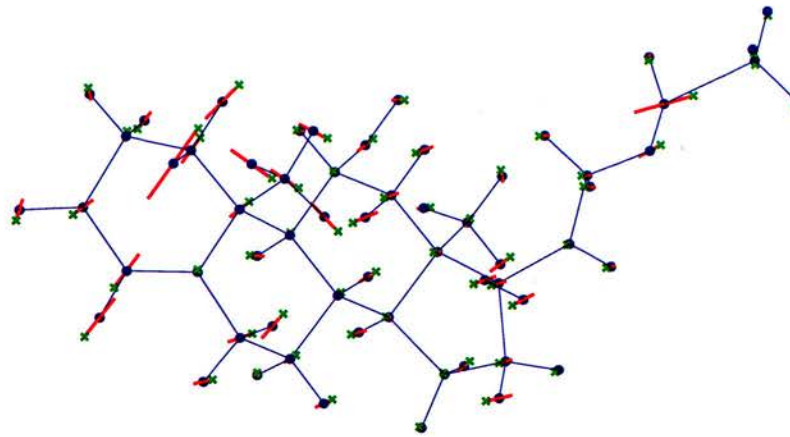

Figure 6.8: The mean steroid shape for a 1D linear PSM model. The red lines show the movement of the atoms as we traverse latent space; the green crosses are the generating shape corresponding to $\mathbf{z}$=-1 in latent space.

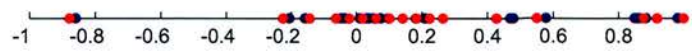

Figure 6.9: The posterior means (blue) and modes (red) for the linear 1D steroid model.

the right of this loop also undergoes significant motion. The posterior means and modes are plotted in Fig. 6.9. The clear outlier on the extreme left corresponds to molecule 16 (etiocholanolone). Visual inspection of this molecule (Fig. 6.7, row 4, column 4) confirms that the left hexagon is indeed folded away from the approximate plane of the other loops. In Fig. 6.10 we consider a model with a two-dimensional latent space (with a 20-by-20 grid) and plot some generating shapes at their corresponding position in this latent space. The horizontal axis captures roughly the same variation as the 1D model illustrated in Fig. 6.8 – the end carbon loop (closest to us from this view) is folded over on the leftmost shapes. We have focused on the movement of this carbon loop because it is the most visually striking component of variation. However, in real chemoinformatics problems, the importance of any variation in the movement or presence/absence behavior of atoms will be dependent on problem-specific information and domain knowledge. For example, a researcher might be particularly interested in a specific region of the molecules.

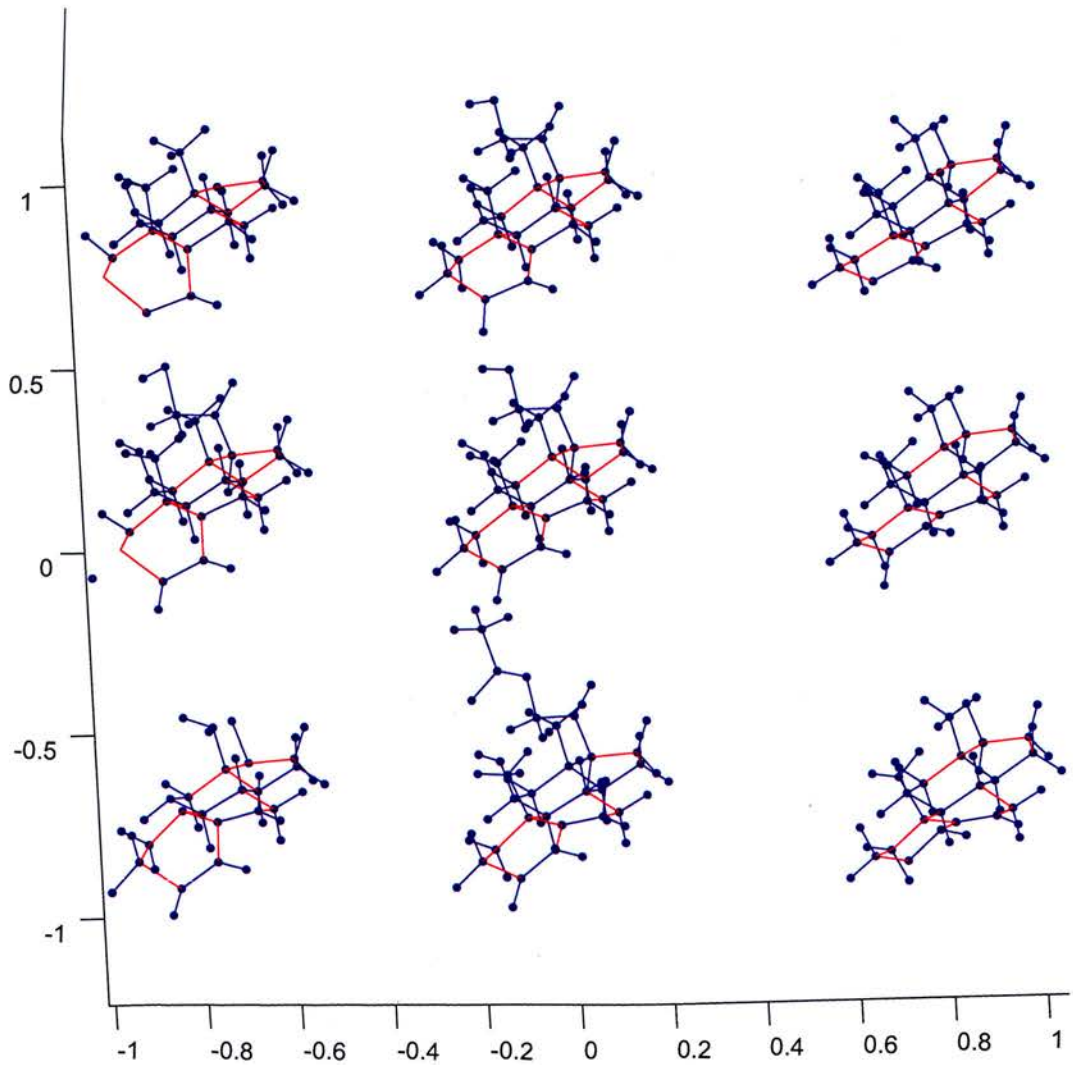

Figure 6.10: Latent space for the steroid data; each shape is shown at the position of the corresponding grid point in latent space. The bonds of the carbon rings are shown in red to aid interpretation.

6.6 Discussion and Future Work

We have introduced a principled approach to modeling unordered point sets and shown that it can learn accurate models from data sets for which the true generative process is known (Sections 6.3 and 6.4). While the main contribution of this chapter is theoretical, we note that point set modeling has applications in a variety of fields. Section 6.5 gives an example of how PSM can be applied to chemoinformatics data. In computer vision, modeling the positions of features sets such as edge point, corners or SIFT keypoints [Lowe, 2004] could be used in object recognition.

PSM is based solely on the spatial locations of points, but could be extended to include feature values at these points – *e.g.* atom type in chemoinformatics or local appearance in computer vision. The simplest way to incorporate such information would be to follow the approach of Dryden et al. (2007) and assume that point position and feature value are conditionally independent given the model parameters. Under this assumption, the likelihood in eq.(6.3) for the location of a point would simply be multiplied by the likelihood of the feature value. However, there are many potential applications where this independence assumption will not be suitable and hence, where the joint distribution over position and feature values will not factorize. Future work on PSM will investigate different approaches to incorporating feature information as well as addressing the important problem of model selection.

While PSM can be applied to any collection of unordered point sets, it has not been tested on *dense point sets* where exact pointwise correspondences may not exist (*e.g.* as a result of sampling from lines or surfaces) or where the notion of a pointwise correspondence may be less natural (*e.g.* 2D shape silhouettes) . There is a danger that PSM may find it more difficult to distinguish between genuine shape variation and noise with this type of data. If the noise variance (*i.e.* the variance of the small Gaussian about each of the generating points) is too large, the algorithm could simply spread out the many points of the mean shape and explain the data using a 'mean shape plus noise' model. On the other hand, if the noise variance is not large enough to capture the additional 'correspondence noise' associated with dense point sets, the model will incorrectly treat this noise as genuine shape variation. A possible solution to this problem would be to use a relatively large noise variance and a mean shape with fewer points – essentially sacrifice resolution for robustness. In particular, this strategy may be effective for dense

point sets such as 2D silhouettes where the correspondence noise can be assumed to be isotropic. For point sets representing lines or surfaces embedded in a higher-dimensional space, the correspondence noise will be tangential to the mean line/surface and it may be necessary to learn non-spherical noise Gaussians during parameter estimation.

Chapter 7

Class Models for Ordered Point Sets

In this chapter, we consider the problem of learning a class model from *ordered* point sets – *i.e.* curves or contours. The Probabilistic *Contour* Model (PCM) introduced here is conceptually similar to the Probabilistic Shape Model (PSM) introduced in the previous chapter. However, the generative model now involves a 'generating curve' rather than a 'generating point set' and accordingly, the correspondence problems involve reparameterizing curves rather than labeling individual points.[1]

7.1 Introduction

Statistical shape models of 2D contours are used in medical image analysis and object recognition. Many of the previous approaches to this problem are conceptually similar [Kotcheff & Taylor, 1998, Taylor et al., 2000, Davies et al., 2002, Ericsson & Astrom, 2003b, Thodberg & Olafsdottir, 2003, Wang et al., 2004, Hladuvka & Buhler, 2005]: given some observed points from multiple contours, fit curves (often polylines) to the data and then slide points around each curve to find the optimal correspondence with respect to an objective function. Noise in the data is usually not modeled and a 'reference shape' with fixed points/parameterization is typically required to prevent degenerate solutions whereby points cluster about a single region of the contour.

The method proposed by Kotcheff and Taylor (1998) forms the basis of much recent work in this area. Given a Procrustes alignment of the training shapes, they monotonically reparameterize each training shape so as to minimize the determinant of the sample

[1] The material contained in this chapter is also described in [McNeill & Vijayakumar, 2007].

covariance matrix. A Gaussian noise model with *pre-specified variance* is introduced to avoid numerical problems. Davies et al. (2002) use a similar formulation, assuming a Gaussian model over the training shapes and using the minimum description length (MDL) framework to learn smooth reparameterization functions. MDL is typically used for model selection whereby model complexity is balanced against the ability of the model to explain the data. However, in the approach of Davies et al. (2002) the model is fixed and the objective function is similar to that used by Kotcheff and Taylor (1998).

A number of modifications to the MDL approach have been proposed. Thodberg has incorporated curvature information [Thodberg & Olafsdottir, 2003] and extended the MDL technique to appearance models [Thodberg, 2003]. Efficient gradient-based techniques for learning reparameterization functions have been introduced for discrete [Ericsson & Astrom, 2003b] and continuous shape representations [Hladuvka & Buhler, 2005]. Ericsson and Astrom (2003a) have proposed an alternative formulation to MDL which is invariant to affine transformations and Cates et al. (2006) have recently proposed a novel particle-based approach.

In this chapter, we introduce the Probabilistic Contour Model (PCM) and the Nonlinear Probabilistic Contour Model (NPCM).[2] The generative model for PCM/NPCM can be summarized as follows:

1. Sample a single point from a low-dimensional latent space.
2. Map the point to a 2D curve using a linear (PCM) or nonlinear (NPCM) mapping.
3. Reparameterize the curve, sample it and add Gaussian noise to the sample points to generate the observed data.

The inclusion of an explicit low-dimensional latent space and noise model contrasts with techniques which treat major and minor components of variation separately in the objective function [Davies et al., 2002, Ericsson & Astrom, 2003b, Thodberg & Olafsdottir, 2003, Hladuvka & Buhler, 2005]. Neither the existing techniques nor PCM are designed to handle nonlinear patterns of variation and we extend PCM to NPCM to address this limitation. Since many data sets contain noise of *unknown* magnitude associated with either the shapes themselves (*e.g.* due to random factors in the biological processes that

[2] In contrast to the previous chapter, it is natural to consider linear and nonlinear models separately here – see Sections 7.3 and 7.4 for more details.

generated them) or the image capturing and processing techniques used for shape extraction, we estimate the noise variance during learning rather than specifying it *a priori*. In PCM/NPCM, the underlying (noise-free) contours are effectively latent variables so unlike in other approaches, curves are not fitted to the data prior to learning the model. Also, the observed data points remain fixed rather than being slid around, so there is no danger of point clustering and hence, no need for a reference shape.

7.2 Shapes, Curves and Reparameterization Functions (RFs)

We start by introducing the main components that will be used to construct a contour model. An ordered 2D point set is represented by the matrix $\mathbf{Y} \in \mathbb{R}^{J\times 2}$, where each row contains the coordinates of a single point and points are stacked in the order they appear on the underlying contour. We will often consider the long vector $\mathbf{y} \in \mathbb{R}^{2J}$ formed by concatenating the x-coordinates and the y-coordinates:

$$\mathbf{y} \equiv \text{vec}(\mathbf{Y}). \tag{7.1}$$

A shape model is learnt from N ordered point sets, $\mathbf{Y}_1, \ldots, \mathbf{Y}_N$ – note that shapes are indexed using subscript in this chapter. The PCM model introduced in the next section can easily accommodate point sets of unequal size, but to avoid complicating the notation, we will assume that each point set contains the same number of points, J. The 2D curve responsible for generating $\mathbf{Y}_n$ is described by a linear combination of basis functions:

$$f_n(t) = \phi^T(t)[\mathbf{v}_n^x, \mathbf{v}_n^y], \tag{7.2}$$

where t is the curve parameter, $\mathbf{v}_n^x, \mathbf{v}_n^y \in \mathbb{R}^K$ are vectors of coefficients and

$$\phi(t) \equiv (\phi_1(t), \ldots, \phi_K(t))^T \tag{7.3}$$

is a vector of basis functions. Again, we frequently consider the long vectors $\mathbf{v}_n \in \mathbb{R}^{2K}$ formed by concatenating $\mathbf{v}_n^x$ and $\mathbf{v}_n^y$:

$$\mathbf{v}_n \equiv \text{vec}([\mathbf{v}_n^x, \mathbf{v}_n^y]). \tag{7.4}$$

For closed curves, the basis functions are periodic, *e.g.* Fourier, wrapped Cauchy, von Mises, or periodic B-spline.

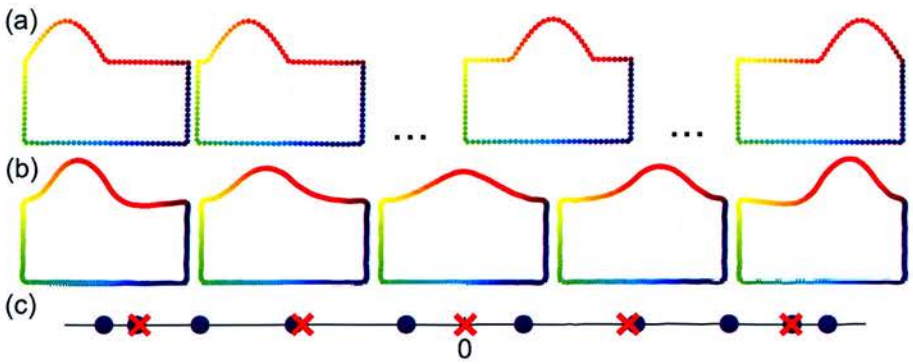

Figure 7.1: (a) 4 of the 10 point sets from the box-bump data set; color represents the value of the arc length parameter. (b) Curves associated with equal increments along the 1-dimensional latent space of a probabilistic contour model (PCM) that does not learn reparameterization functions (RFs). (c) Latent space: the circles are the posterior means of the 10 data shapes, the crosses are the latent variables of the curves in (b).

Given a training set, we assume that the *same point* (*e.g.* the top of the bump for the shapes in Fig. 7.1a) varies in both its spatial location and the distance along the shape perimeter at which it appears. The idea is to solve the correspondence problem by reparameterizing the curves from which the points were sampled, allowing us to construct a shape model over the spatial locations of corresponding points. This is demonstrated in Figs. 7.1 and 7.2 where an initial poor correspondence between training shapes (Fig. 7.1a – the color of the bump on the leftmost shape is noticeably different from that on the rightmost) is improved (Fig. 7.2a) leading to a better shape model – the generated shapes in Fig. 7.2b resemble the training shapes more than those in Fig. 7.1b. For further details of the box-bump experiment, refer to Section 7.6.1.

We now consider curve reparameterization in more detail and introduce a monotonic reparameterization function (RF), $g_n(t)$, for each curve, $f_n(t)$. RFs can be expressed as the integral of a linear combination of basis functions where both the basis functions and the coefficients are non-negative [Davies et al., 2002, Hladuvka & Buhler, 2005]. To avoid these non-negativity constraints, we use a 'monotonicity operator' [Ramsay & Silverman, 2005]: take an unconstrained linear combination of arbitrary basis functions, exponentiate to ensure positivity and then integrate to ensure monotonicity. Ramsay and Silverman (2005) have successfully applied this idea to curve registration using an intuitive least-squares error function. Here, registration is carried out with respect to the

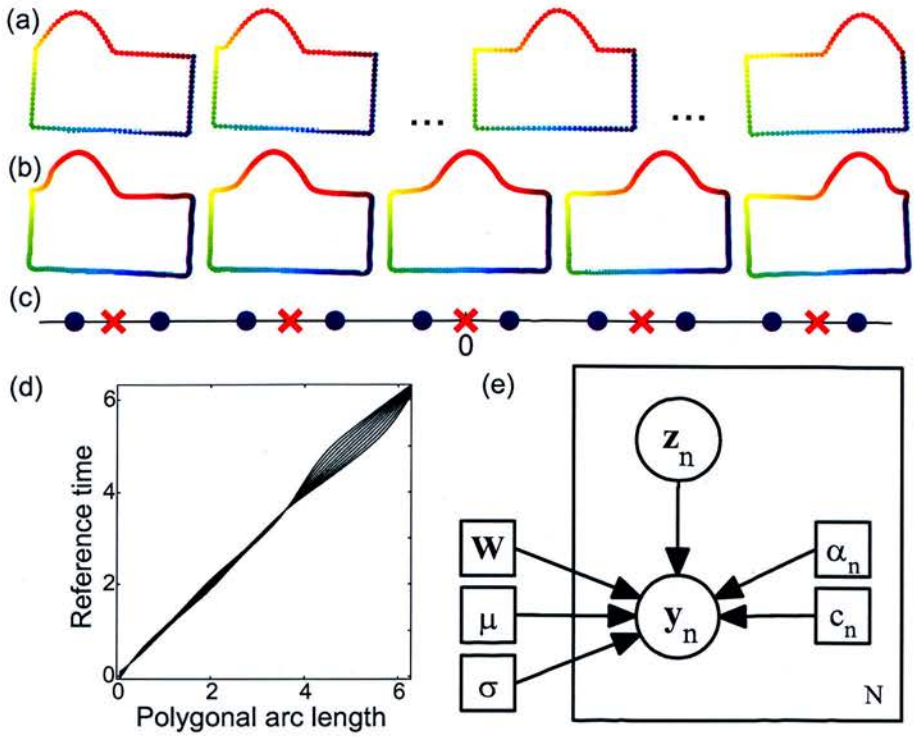

Figure 7.2: PCM for the bump-box data: (a) The reparameterized data shapes; color represents the value of the 'reference time' parameter. (b) Curves associated with equal increments along the latent space. (c) The latent space. (d) The RFs. (e) Graphical model for PCM.

latent variable models introduced in the following sections.

Closed curves are more difficult to model than open curves since the start point on each shape is generally unknown and hence, each RF must contain a shift parameter. Focusing on the closed curve problem, we define the RF for the n-th shape, $g_n(t) : [0, 2\pi] \rightarrow [c_n, 2\pi + c_n]$ as

$$g_n(t) \equiv 2\pi \frac{\int_0^t \exp(\psi^T(\tau)\alpha_n)d\tau}{\int_0^{2\pi} \exp(\psi^T(\tau)\alpha_n)d\tau} + c_n, \tag{7.5}$$

where $\psi = (\psi_1(t), \ldots, \psi_Q(t))^T$ is a vector of basis functions and $\alpha_n \in \mathbb{R}^Q$ is a vector of coefficients. The values $t = c_n$ and $t = 2\pi + c_n$ correspond to the same point on the curve so, assuming that smooth RFs are desired, we should ensure that the derivative of $g_n(t)$ is equal at c_n and $2\pi + c_n$. This is most easily achieved by using smooth 2π-periodic basis functions.

For each curve n, we approximate the value of the arc length parameter at each observed point using the polygonal approximation to the contour (this is only used for initialization and need not be particularly accurate). These values are normalized to lie in $[0, 2\pi]$ and then stored in the vector

$$\mathbf{t}_n = (t_{n1}, t_{n2}, \ldots, t_{nJ})^T. \tag{7.6}$$

Combining eqs.(7.2),(7.5) and (7.6), we can see that the point set $\mathbf{Y}_n$ is approximated by evaluating the composite function $f_n(g_n(t))$ at the entries of $\mathbf{t}_n$. The same approximation can be written in terms of the long vectors $\mathbf{y}_n$ (eq.(7.1)) and $\mathbf{v}_n$ (eq.(7.4)) and expressed as a generalized linear regression on $g_n(t)$:

$$\mathbf{y}_n \approx \Phi_n \mathbf{v}_n, \tag{7.7}$$

where $\Phi_n \in \mathbb{R}^{2J \times 2K}$ is the design matrix defined as

$$\Phi_n \equiv \begin{pmatrix} \Omega_n & \mathbf{0} \\ \mathbf{0} & \Omega_n \end{pmatrix}; \quad [\Omega_n]_{jk} \equiv \phi_k(g_n(t_{nj})). \tag{7.8}$$

By initializing each g_n to the identity function, we initially estimate the correspondences using arc length. During learning, the $\mathbf{t}_n$ remain fixed but the g_n change. We are now in a position to define the probabilistic model.

7.3 Probabilistic Contour Model (PCM)

As in the previous chapter, we assume that the intrinsic dimensionality of the data is low and accordingly, use a latent variable model with a low-dimensional latent space – Fig. 7.2e. Letting $\mathbf{z}_n \in \mathbb{R}^L$ be the latent variable associated with shape n (typically $L \ll K \ll J$), the prior distribution of the i.i.d. $\mathbf{z}_n$ is assumed to be a spherical Gaussian

$$\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_L). \tag{7.9}$$

The key component of the model is the conditional distribution

$$\mathbf{y}_n | \mathbf{z}_n \sim \mathcal{N}(\Phi_n(\mathbf{W}\mathbf{z}_n + \mu), \sigma^2 \mathbf{I}_{2J}), \tag{7.10}$$

where $\sigma^2 \in \mathbb{R}$ is the noise variance, $\mathbf{W} \in \mathbb{R}^{2K \times L}$ is a linear mapping from latent space to coefficient space (which contains the $\mathbf{v}_n$ introduced in the previous section) and $\mu \in \mathbb{R}^{2K}$

is the mean coefficient vector. There is independent Gaussian noise of equal magnitude on all output dimensions which amounts to a circular Gaussian about each of the J observed 2D points. It is easily shown that the marginal distribution of $\mathbf{y}_n$ is

$$\mathbf{y}_n \sim \mathcal{N}(\Phi_n \mu, \Phi_n \mathbf{W}\mathbf{W}^T \Phi_n^T + \sigma^2 \mathbf{I}_{2J}), \tag{7.11}$$

and that the posterior distribution is given by

$$\mathbf{z}_n | \mathbf{y}_n \sim \mathcal{N}(\mathbf{M}_n^{-1} \mathbf{W}^T \Phi_n^T (\mathbf{y}_n - \Phi_n \mu), \sigma^2 \mathbf{M}_n^{-1}), \tag{7.12}$$

where

$$\mathbf{M}_n \equiv \mathbf{W}^T \Phi_n^T \Phi_n \mathbf{W} + \sigma^2 \mathbf{I}_L. \tag{7.13}$$

The generative model is strongly related to probabilistic principal component analysis (PPCA) [Bishop, 2006]. However, rather than learning a linear mapping from the latent space directly to the data space, we map the latent space to the coefficients of the curve. The curve associated with these coefficients is then evaluated at the 'warped time points' $g_n(t_{n1}), g_n(t_{n2}), \ldots, g_n(t_{nJ})$ and the data points are generated by small isotropic Gaussians centered at each of the evaluated points. Note that once the model parameters have been estimated, all new curves generated from the model are functions of the same 'reference time' parameter (*i.e.* they are correctly corresponded) and need not be reparameterized.

Having considered all the distributions, the next step is to estimate the model parameters. In PPCA, the maximum likelihood estimates (MLEs) of the parameters can be computed in closed form or using the EM algorithm [Bishop, 2006]. Here, the presence of the RFs complicates parameter estimation and there is no closed form solution for the MLEs and no exact M-step for the EM algorithm. However, minimizing the expectation of the complete log-likelihood with respect to one of $\mathbf{W}, \mu, \sigma^2, \{\alpha_1, c_1, \ldots, \alpha_N, c_N\}$ while holding the others fixed is reasonably straightforward and leads to the ECM algorithm in Table 7.1. Note that there is no closed form expression for $\{\alpha_n, c_n\}_{new}$ (eq.(7.18)) so a nonlinear optimization technique is required. In all experiments, the Levenberg-Marquardt algorithm was used for this minimization and the trapezoidal rule was used to evaluate the RFs (eq.(7.5)).

As with other approaches, it is assumed that scale, translation and rotation are nuisance transformations that do not alter the actual shape of a contour. Rather than complicating the model by including these transformations in the conditional distribution

$p(\mathbf{y}_n|\mathbf{z}_n)$ (eq.(7.10)), we have found that transforming each point set $\mathbf{Y}_n$ so as to maximize the expected complete log-likelihood works well in practice, *i.e.* we maximize the same objective function used for parameter estimation, but transform the data shapes rather than the model. Since the posterior distribution $p(\mathbf{z}_n|\mathbf{y}_n)$ is Gaussian, this maximization reduces to a standard Procrustes matching problem between $\mathbf{Y}_n$ and the point set corresponding to the maximum *a posteriori* (MAP) estimate of $\mathbf{z}_n$. Scale is not included in the optimization since this would enable the model to produce a high likelihood by simply shrinking all the point sets. To remove the impact of scale, all point sets are normalized to have equal size, where size is defined as the mean squared distance from the centroid.

Note that in contrast to the PSM model of the previous chapter, there is no need to discretize the latent space here – *i.e.* we are not faced with intractable integrals over the latent space. Indeed, by combining a linear model with the Gaussian distributions in eqs.(7.9) and (7.10), the marginal and posterior distributions are also Gaussian (eqs.(7.11) and (7.12)) and the M-step updates in the ECM algorithm are easily derived (Table 7.1).[3] In the next section, we consider nonlinear models where it will once again be necessary to discretize the latent space.

7.4 Nonlinear PCM (NPCM)

In PCM, the marginal distribution $p(\mathbf{y}_n)$ is Gaussian (eq.(7.11)) and hence, the distribution of each 2D boundary point is Gaussian. Such a model is not suitable for data sets displaying non-linear variation (*e.g.* where a single point traces out a curved path), but this limitation is not unique to PCM; to the best of our knowledge, none of the existing algorithms for *learning the correspondences and the model simultaneously* are designed to handle nonlinear shape variation. In contrast, a variety of techniques have been proposed for handling complex shape variation *when the correspondence between training shapes is known.* For example, one can fit a mixture of Gaussians to a low dimensional representation of the data rather than a single Gaussian [Cootes & Taylor, 1999]. Alternatively, one can focus on the mapping between data space and latent space. This is the approach taken by Twining and Taylor (2001), where standard PCA is replaced with

[3]Recall that *mixtures of Gaussians* were used in Chapter 6 and that the model could be linear or nonlinear.

Table 7.1: ECM algorithm for learning a Probabilistic Contour Model (PCM).

E-step Compute the sufficient statistics using the current parameter values (the following expectations are taken with respect to the distributions $p(\mathbf{z}_n|\mathbf{y}_n)$):

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}_n^{-1}\mathbf{W}^T\Phi_n^T(\mathbf{y}_n - \Phi_n\mu), \qquad \mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \sigma^2\mathbf{M}_n^{-1} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^T. \tag{7.14}$$

CM-steps Update the parameters using the sufficient statistics; repeat E-step between each CM step. Numerical optimization (Levenberg-Marquardt algorithm) is used in eq.(7.18). The $\otimes$ symbol denotes the Kronecker product.

1. Noise variance (see eq.(7.10)):

$$\sigma^2_{new} = \frac{1}{2NJ}\sum_{n=1}^{N}\left(\|\mathbf{y}_n - \Phi_n\mu\|^2 - 2\mathbb{E}[\mathbf{z}_n]^T\mathbf{W}^T\Phi_n^T(\mathbf{y}_n - \Phi_n\mu) + \mathrm{Tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]\mathbf{W}^T\Phi_n^T\Phi_n\mathbf{W})\right). \tag{7.15}$$

2. Mean curve coefficient vector (see eq.(7.10)):

$$\mu_{new} = \left(\sum_{n=1}^{N}\Phi_n^T\Phi_n\right)^{-1}\sum_{n=1}^{N}\Phi_n^T(\mathbf{y}_n - \Phi_n\mathbf{W}\mathbb{E}[\mathbf{z}_n]). \tag{7.16}$$

3. Mapping from latent space to curve coefficient space (see eq.(7.10)):

$$\mathrm{vec}(\mathbf{W}_{new}) = \left(\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]\otimes\Phi_n^T\Phi_n\right)^{-1}\mathrm{vec}\left(\sum_{n=1}^{N}\Phi_n^T(\mathbf{y}_n - \Phi_n\mu)\mathbb{E}[\mathbf{z}_n]^T\right). \tag{7.17}$$

4. Parameters of reparameterization functions (see eq.(7.5)):

$$\{\alpha_n, c_n\}_{new} = \arg\min_{\{\alpha_n,c_n\}} -\|\mathbf{y}_n - \Phi_n\mu\|^2 + 2\mathbb{E}[\mathbf{z}_n]^T\mathbf{W}^T\Phi_n^T(\mathbf{y}_n - \Phi_n\mu) - \mathrm{Tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]\mathbf{W}^T\Phi_n^T\Phi_n\mathbf{W}). \tag{7.18}$$

Procrustes Step Rotate and translate each point set $\mathbf{Y}_n$ so as to minimize $\|\mathbf{y}_n - \Phi_n(\mathbf{W}\mathbb{E}[\mathbf{z}_n] - \mu)\|^2$ (recall that $\mathbf{y}_n \equiv \mathrm{vec}(\mathbf{Y}_n)$). This is a Procrustes problem which can be solved as described in Section 2.3, but where scale is excluded from the optimization.

---

kernel PCA.

In this section, we extend PCM to nonlinear PCM (NPCM) by allowing the mapping from latent space to coefficient space to be nonlinear. As in Chapter 6, this is achieved by passing the latent variable through a set of nonlinear basis functions. The modified conditional distribution (*cf.* eq.(7.10)) is given by

$$\mathbf{y}_n|\mathbf{z}_n \sim \mathcal{N}(\Phi_n(\mathbf{W}\gamma(\mathbf{z}_n)+\mu), \sigma^2\mathbf{I}_{2J}), \tag{7.19}$$

where

$$\gamma(\mathbf{z}_n) \equiv (\gamma_1(\mathbf{z}_n), \gamma_2(\mathbf{z}_n), \ldots, \gamma_U(\mathbf{z}_n))^T \tag{7.20}$$

is a vector of basis functions with $\gamma_u(\mathbf{z})$:$\mathbb{R}^L \rightarrow \mathbb{R}$ and $\mathbf{W}$ is now a $2K \times U$ matrix.

The nonlinearity introduced in eq.(7.19) allows the marginal distribution $p(\mathbf{y}_n)$ to be non-Gaussian, enabling us to capture more complex types of shape variation. However, eq.(7.19) also complicates the expression for the expected complete log-likelihood and we can no longer derive a clean ECM algorithm (*cf.* Table 7.1). As in the previous chapter, we overcome this by problem by switching from a Gaussian prior to a discretized uniform prior:

$$p(\mathbf{z}_n) = \frac{1}{G}\sum_{g=1}^{G}\delta(\mathbf{z}_n - \mathbf{z}_g), \tag{7.21}$$

where the $\mathbf{z}_g$ are grid points of the latent space. For each n, the grid prior gives rise to a constrained G-component Gaussian mixture model in data space: each $\mathbf{z}_g$ is mapped to a coefficient vector $\mathbf{W}\gamma(\mathbf{z}_g)+\mu$, the curve associated with these coefficients is evaluated at $g_n(t_{nj})$ $(j = 1,2,\ldots,J)$ and a spherical Gaussian is placed at each of the J 2D points. Since $\mathbf{y}_n \in \mathbb{R}^{2J}$, each component of the mixture model is formally a spherical Gaussian of dimension $2J$.

As discussed above, the linear PCM model may be *too simple* and hence, unable to explain shape variation even *given the correct RFs*. Conversely, with NPCM there is the danger of overfitting: the model may be *too complex* and will be able explain shape variation *given incorrect RFs*. The natural solution to this problem is to guide the algorithm towards smooth mappings by regularizing $\mathbf{W}$. As in GTM [Bishop et al., 1998] and the PSM model of Chapter 6, we define a radially-symmetric Gaussian prior over the entries of $\mathbf{W}$:

$$p(\mathbf{W}|\nu^2) = \left(\frac{1}{2\pi\nu^2}\right)^{UK}\exp\left\{-\frac{1}{2\nu^2}\sum_{k=1}^{2K}\sum_{u=1}^{U}w_{ku}^2\right\}. \tag{7.22}$$

and use the MAP estimate to update $\mathbf{W}$ in the M-step (eq.(7.26)). The model parameters are estimated using the ECM algorithm in Table 7.2.[4]

7.5 Implementation

To implement PCM, the user must specify the following parameters:

L – the number of latent space dimensions (1, except in Section 7.6.2).

K – the number of curve basis functions for x and y coordinates (50).

Q – the number of RF basis functions (8).

The numbers in brackets are the values used in our experiments. The low-dimension of the latent space combined with the noise model essentially regularizes the curves – it decides what constitutes genuine shape variation. Thus, we can choose a large K and not worry about overfitting. The RFs are not regularized but this could easily be incorporated. The choice of L is an important model selection problem and future work will consider automatic selection of this parameter.

In all experiments, von Mises shaped unimodal periodic basis functions were used for both the curves and the RFs:

$$\phi(t;\rho_q,\kappa) = e^{\kappa\cos(t-\rho_q)} / e^{\kappa}, \tag{7.28}$$

where ρ_q is the center and κ is the width parameter. Given K (or Q), the centers were placed at equal intervals in $[0,2\pi]$ and κ was fixed at $K/2$ (or $Q/2$). Note that multi-scale RFs could be investigated (without changing the model or its implementation) by including basis functions of different widths.

To initialize PCM, the point sets are aligned to the mean (with respect to the initial correspondences) using generalized Procrustes matching, a 2D curve is fitted to each point set and then PPCA on the coefficient vectors is used to initialize μ, σ and $\mathbf{W}$.[5] The RFs are initialized to the identity function by setting $\alpha_n = \mathbf{0}$ and $c_n = 0$ (eq.(7.5)). The

[4] A hard assignment is used during the Procrustes step to improve efficiency.

[5] For many choices of basis function, isotropic noise on the coefficients does not equate to noise of equal magnitude at each boundary point. This is one of the problems associated with fitting fixed curves to the data points and then modeling the curve coefficients directly. Here, the fitted curves are only used for initialization.

Table 7.2: ECM algorithm for learning a Nonlinear Probabilistic Contour Model (NPCM).

E-step Compute the responsibilities using the current parameter values and eq.(7.19):

$$p(\mathbf{z}_g|\mathbf{y}_n) = \frac{p(\mathbf{y}_n|\mathbf{z}_g)}{\sum_g p(\mathbf{y}_n|\mathbf{z}_g)}. \tag{7.23}$$

CM-steps Update the parameters; numerical optimization (Levenberg-Marquardt algorithm) is used in eq.(7.27).

1. Noise variance (see eq.(7.10)):

$$\sigma^2_{new} = \frac{1}{2NJ}\sum_{n,g} p(\mathbf{z}_g|\mathbf{y}_n)\|\mathbf{y}_n - \Phi_n(W\gamma(\mathbf{z}_g)+\mu)\|^2. \tag{7.24}$$

2. Mean curve coefficient vector (see eq.(7.10)):

$$\mu_{new} = \left(\sum_{n=1}^{N}\left(\sum_{g=1}^{G} p(\mathbf{z}_g|\mathbf{y}_n)\right)\Phi_n^T\Phi_n\right)^{-1}\sum_{n,g} p(\mathbf{z}_g|\mathbf{y}_n)\Phi_n^T(\mathbf{y}_n - \Phi_n W\gamma(\mathbf{z}_g)). \tag{7.25}$$

3. Mapping from *transformed* latent space to curve coefficient space (see eq.(7.10)):

$$\begin{aligned}\mathrm{vec}(\mathbf{W}_{new}) \;=\; & \left(\sum_{n,g} p(\mathbf{z}_g|\mathbf{y}_n)\left(\gamma(\mathbf{z}_g)\gamma^T(\mathbf{z}_g)\right)\otimes\left(\Phi_n^T\Phi_n\right) + \frac{\sigma^2}{\nu^2}\mathbf{I}_{2KU}\right)^{-1} \\ & \times \quad \mathrm{vec}\left(\sum_{n,g} p(\mathbf{z}_g|\mathbf{y}_n)\Phi_n^T(\mathbf{y}_n - \Phi_n\mu)\gamma^T(\mathbf{z}_g)\right).\end{aligned} \tag{7.26}$$

4. Parameters of reparameterization functions (see eq.(7.5)):

$$\{\alpha_n, c_n\}_{new} = \arg\min_{\{\alpha_n,c_n\}} \sum_g p(\mathbf{z}_g|\mathbf{y}_n)\|\mathbf{y}_n - \Phi_n(W\gamma(\mathbf{z}_g)+\mu)\|^2. \tag{7.27}$$

Procrustes Step Rotate and translate each $\mathbf{Y}_n$ to match the point set associated with $\arg\max_{\mathbf{z}_g} p(\mathbf{z}_g|\mathbf{y}_n)$.

data sets used in Section 7.6 were chosen to enable comparison with other techniques and in all cases, the first point of each shape is correctly corresponded across training examples. Since this information may not be available in some applications, it is worth noting that PCM/NPCM only requires rough initial correspondences and that there are accurate, efficient algorithms for finding these (*e.g.* CPM, Chapter 3).

PCM is surprisingly robust and if a data set contains nonlinear variation, it tends to produce as good a linear model as we might hope for rather than failing completely. This suggests using PCM to initialize NPCM which can be achieved by setting the first L entries of $\gamma(\mathbf{z})$ in eq.(7.20) equal to the entries of $\mathbf{z}$, setting the first L columns of $\mathbf{W}$ in eq.(7.19) equal to the appropriately scaled $\mathbf{W}$ from PCM (eq.(7.10)) and the remaining entries to zero. There are alternative ways in which PCM could be used to initialize NPCM but we have found this approach to work well in practice. It is straightforward to assign a different prior variance to the entries of $\mathbf{W}$ associated with different types of basis function, but in our experiments we used the single default value given below. The additional parameters required for NPCM are:

G – the number of grid points in latent space (50, except in Section 7.6.2).

U – the number of basis functions for the latent space $\rightarrow$ coefficient space mapping (1 linear + 8 radial basis functions (RBFs), except in Section 7.6.2).

ν^2 – the prior variance on the entries of $\mathbf{W}$ (0.1).

As noted in Section 6.2.3, G should be large in order to accurately approximate a continuous latent space. Gaussian shaped RBFs were used with centers placed on a uniform grid containing $U - L$ vertices and with the variance fixed so that neighboring centers were two standard deviations apart.

The algorithms were implemented in Matlab on a 1.6GHz Intel Centrino Duo machine. In all experiments, 200 EM iterations were used for both PCM and NPCM (200 E-steps, 40 each of the CM/Procrustes steps). Learning the shape model for a data set of 22 shapes, each described by 128 points and using the default parameters above took <3min for PCM and <9min for NPCM.

7.6 Experiments and Evaluations

7.6.1 Illustrative Examples: Box-bump Data

Fig. 7.1a shows a 'box-bump' data set similar to that used by Davies (2002), Hladuvka and Buhler (2005) and Ericsson and Karlsson (2006). Shapes generated from a model with fixed arc-length RFs are shown in Fig. 7.1b. The resemblance between the data shapes and the generated shapes is poor (*e.g.* the bump of the mean shape plotted in the center of Fig. 7.1b is low and elongated) demonstrating, as has been done in the past, that arc length parameterization produces a poor shape model for this type of data.

Note that unlike most natural data sets, the distribution of the shapes in Fig. 7.1a is uniform given the correct correspondence. For example, the point at the top of the bump is located at equal increments along a straight line as the bump moves from left to right. The same is true of all points on the top of the shape (including non-bump points), whereas points on the side and base of the shapes do not move at all. Given this uniform distribution in data space and the fact that PCM is a linear model, we would expect the posterior means of the training shapes to be uniformly distributed in latent space if the learnt model is accurate. As can be seen in Fig. 7.1c, the posterior means (circles) are neither uniformly distributed nor do they reflect the Gaussian prior .

The results of applying PCM to the box-bumps are summarized in Fig. 7.2. The slight rotations of the shapes in the figures are those applied by PCM during the Procrustes step; as would be expected, the generated shapes are similarly rotated. These alignment transformations are included in the figures for completeness, but note that the orientation of the shapes is of no importance when assessing the accuracy of the shape model. In contrast to Fig. 7.1b, the generated shapes in Fig. 7.2b closely resemble the data shapes indicating that an accurate shape model has been learnt. Also, the learnt correspondences in Fig. 7.2a are better than those associated with arc length in Fig. 7.1a – the color distribution on each bump is approximately the same in Fig. 7.2a. The RFs responsible for these improved correspondences are plotted in Fig. 7.2d. Finally, Fig. 7.2c (circles) shows the 1D latent space with each training shape represented by its posterior mean – the mean of the Gaussian in eq.(7.12). Note that the circles are now uniformly spaced.

In Fig. 7.3a, each point of the box-bump shapes has been contaminated with independent Gaussian noise of standard deviation 0.3. PCM is robust to this type of noise by

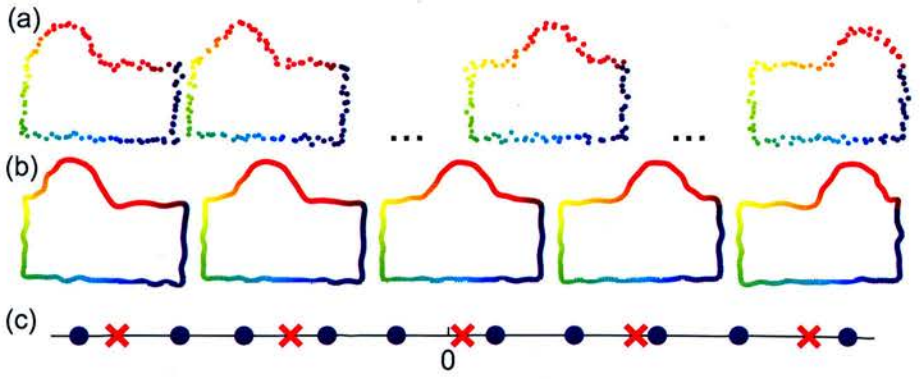

Figure 7.3: (a) Noisy box-bump data with learnt parameterizations. (b) Curves associated with equal increments in latent space. (c) Latent space.

construction and learns a reasonably accurate model with an estimated standard deviation of 0.26. Note that the generated shapes in Fig. 7.3b reflect the true shape variation despite the large noise variance and there being only 10 training shapes.

7.6.2 Dimensionality Reduction/Visualization

Standard dimensionality techniques such as PCA, PPCA [Tipping & Bishop, 1999], factor analysis [Mardia et al., 1979], multidimensional scaling [Cox & Cox, 2000], and GTM [Bishop et al., 1998] can be used to visualize high-dimensional data sets by projecting the data points into a lower dimensional space. In just the same way, we can use PCM/NPCM to visualize high-dimensional shape data (here the data has dimension $2J$) by projecting each data shape onto the latent space. An example of this is shown in Fig. 7.4 using a data set comprised of the 12 shark shapes used in Section 7.6.4 and the 11 fish/shark shapes from Kimia's data set [Sebastian et al., 2004]. Each shape is plotted at the position of its posterior mean, $\mathbb{E}[\mathbf{z}_n]$, where the expectation is taken with respect to the distribution $p(\mathbf{z}_n|\mathbf{y}_n)$. For the 2D NPCM model, we used G=25^2, U=27 (2 linear basis functions and $5^2 = 25$ RBFs) and 200 iterations of the ECM algorithm.

In real-world problems, the reasons for carrying out data visualization are task-specific. Here, we are merely demonstrating *how* PCM/NPCM can be used for visualization; our objective is not to draw conclusions or generate hypothesis about this particular data set.

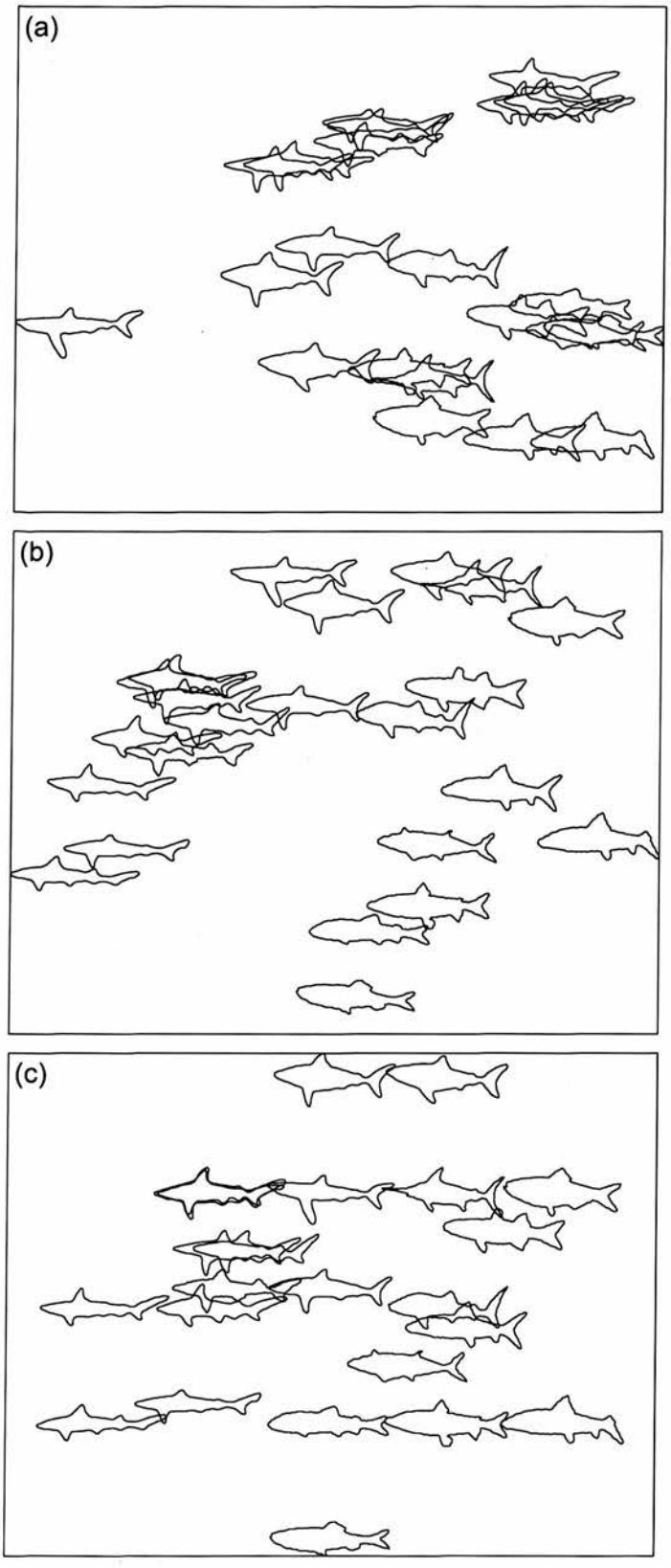

Figure 7.4: Each data shape is shown at the position of its posterior mean. (a) PCM with no RFs. (b) PCM. (c) NPCM.

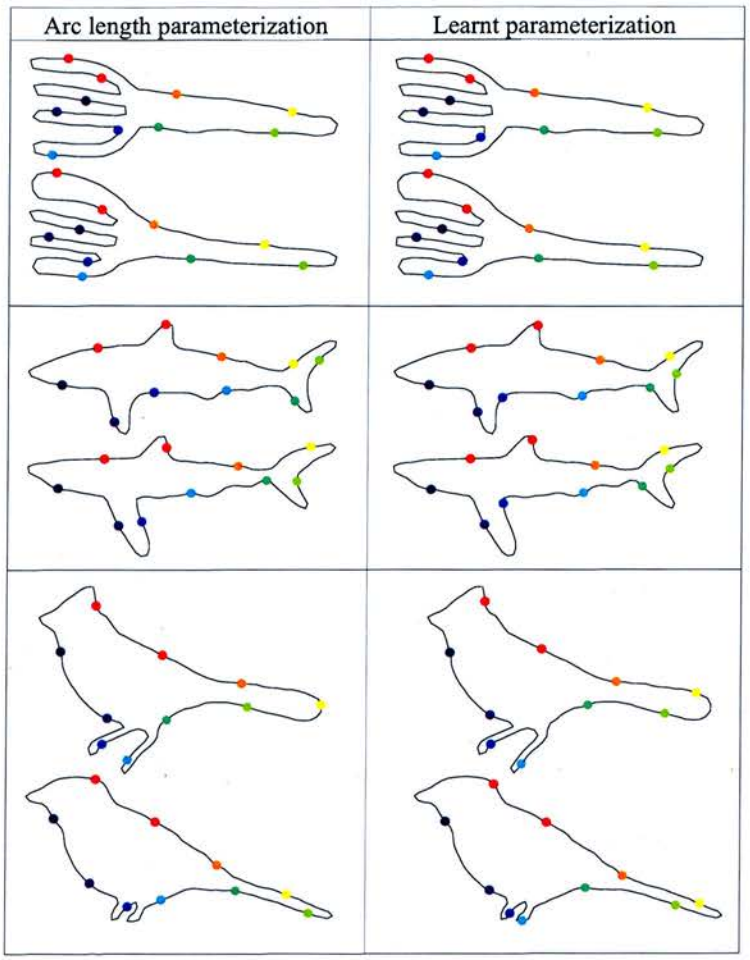

Figure 7.5: Examples of using PCM for pairwise matching.

7.6.3 Pairwise Matching

PCM can be used to solve pairwise matching problems by simply treating the two shapes as a data set of size 2. A detailed model of shape variation cannot be learnt from such a small data set, so all entries of the matrix $\mathbf{W}$ in eq.(7.10) are set equal to zero. This effectively simplifies PCM to a mean shape plus noise model. The approach is illustrated in Fig. 7.5 on pairs of shapes from the data sets used by Ericsson and Karlsson (2006) – these are discussed in detail in the next section. For each pair of shapes, we have plotted the contours and then highlighted points at regular increments of either the arc length parameter (left column) or the learnt 'reference time' parameter (right column). It is clear that many pairs of points are not correctly corresponded in the left column but all

the learnt correspondences in the right column seem reasonable. Note that the matching algorithm is symmetric and involves reparameterizing both shapes in order to maximize the likelihood.

The illustrative examples in Fig. 7.5 indicate that PCM can improve upon arc length based correspondence in non-trivial pairwise matching problems. In the next section, we quantify this improvement for the more complex problem of learning class models from multiple training shapes.

7.6.4 Benchmark Data Sets

Ericsson and Karlsson (2006) recently evaluated state-of-the-art algorithms using their "ground truth correspondence measure" (GCM) which avoids problems associated with the compactness, specificity and generality measures used by Davies (2002). To compute the GCM, multiple independent observers identify a pre-specified set of landmarks on each shape. The resulting distribution of landmarks is then compared to the estimated landmark positions given by the algorithm under evaluation. In simple terms, the GCM is the error in an algorithm's approximation of the ground truth. Table 7.3 shows the GCM values for five shape classes. The second row gives the GCM for an arc length parameterization and subsequent rows give the GCM for different algorithms *as a percentage of the arc length GCM*. The algorithms tested include variants of the MDL approach: the "cur" algorithms use curvature, algorithms 3-5 use different techniques to prevent degenerate solutions (point clustering – a problem that does not arise with PCM/NPCM) and the AIAS+MDL algorithms combine MDL with an affine invariant approach [Ericsson & Astrom, 2003a]. Note that different algorithms perform well on different data sets. For example, the AIAS+MDL algorithms perform very well on the birds data and curvature information seems to be useful for the flightbirds data. This suggests that low-level choices regarding transformation invariance and shape features are important but *application dependent*.

The 128-point shapes and ground truth information used by Ericsson and Karlsson (2006) were used to compute the results in Table 7.4.[6] Aside from the poor performance

[6]Six closed curve data sets were provided but we had difficulty processing the box-bump data and ground truth, so this data set is omitted (but see Section 7.6.1). Also, we are trying to identify the cause of small discrepancies between the arc length GCM values reported here and those published by Ericsson and Karlsson (2006) (0.1-3.4% – 2nd row, Tables 7.3 and 7.4).

Table 7.3: Ground Truth Correspondence Measure for existing algorithms – from Ericsson and Karlsson (2006).

Algorithm	sharks	birds	flightbirds	rats	forks
Arc length	15.55	22.88	7.12	13.37	19.11
Residual error (%):					
1. MDL	27	65	56	29	19
2. MDL,cur	22	80	45	27	23
3. MDL,me	29	92	62	30	20
4. MDL,nodecost	26	67	56	29	19
5. MDL,par	24	62	48	28	18
6. AIAS,MDL	22	23	58	28	20
7. AIAS,MDL,cur	22	24	48	27	24
8. Eucl	44	60	59	37	27
9. Eucl,cur	29	55	54	35	29
10. Cur	22	111	46	29	31

on the rats data, PCM performs reasonably well and NPCM performs very well. It is important to note at this point that PCM/NPCM is *not* a variant of an existing technique (*c.f.* Table 7.3). Rather, it is a novel approach to learning shape models with *application independent* advantages over existing techniques (Section 7.1). As with MDL, variants of the basic PCM/NPCM algorithms (*e.g.* incorporating curvature) could easily be introduced.

Table 7.4: Ground Truth Correspondence Measure: PCM/NPCM.

Algorithm	sharks	birds	flightbirds	rats	forks
Arc length	15.65	22.91	6.93	13.84	19.19
Residual error (%):					
1. PCM	27	56	52	76	20
2. NPCM	21	42	47	51	15

7.7 Discussion

We have presented a generative probabilistic approach to modeling shape contours which overcomes many of the problems associated with existing algorithms. Future work will focus on model selection, extensions to 3D data and investigating mechanisms for handling outliers in both latent space (outlying shapes), and data space (outlying points, often due to occlusion or missing parts).

Chapter 8

Contributions and Future Work

In this chapter, we review the contributions of this thesis and our plans for future work.

8.1 Main Contributions

A multiscale segment-based contour matching algorithm

The hierarchical Procrustes matching (HPM) algorithm described in Chapter 4 is intuitive and effective. Matching segments at a variety of scales avoids problems with purely global or local approaches and the coarse-to-fine matching strategy (*i.e.* matching ever shorter segments) increases robustness and speed. For a fixed set of parameter values, HPM performed very well on several benchmark retrieval tests. This contrasts with many other algorithms reported in the literature where results have only been reported for a single data set, or different parameter values were used for each data set.

Extension of probabilistic shape matching to part-based shapes

Shapes which display part-based variation are common but notoriously difficult to match. In Section 5.3, we extended probabilistic point matching (PPM) to handle part-based shapes and presented a sequential initialization algorithm for selecting the number of parts. This was successfully applied to examples involving both isolated shapes and real images, though further work is needed to assess the robustness of the approach.

A probabilistic model for unordered point sets

Most interesting data sets contain coherent linear or nonlinear shape variation. How-

ever, previous work on modeling unordered point sets has focused on corresponding the training shapes to a single mean shape – *i.e.* the shape variation of interest has not been included in the model. In Chapter 6, we presented the probabilistic shape model (PSM) which can be used to simultaneously correspond shapes and extract the significant components of shape variation. The approach was found to be effective on both artificial (2D) and real (3D) chemoinformatics data sets.

A probabilistic model for ordered point sets

Despite the large amount of research on modeling shape contours, the popular curve reparameterization approaches are typically unable to handle nonlinear variation or noise and also, require a hand-labeled reference shape to prevent degenerate solutions. The (nonlinear) probabilistic contour model ((N)PCM) presented in Chapter 7 is not subject to any of these limitations and performed well on benchmark data sets.

8.2 Future Work

We have made extensive use of the EM algorithm in Chapters 5-7. As discussed in Chapter 5, the soft correspondence used by EM makes it less prone to local minima problems than techniques such as ICP [Besl & McKay, 1992] that use a hard correspondence. However, EM is still an iterative algorithm that can produce sub-optimal parameter estimates and we would like to carry out further empirical investigations to assess the sensitivity of our techniques to initialization. For example, the robustness of the matching algorithms in Chapter 5 could be investigated by considering multiple initial alignments for each problem (either chosen at random or relating to fixed increments in parameter space) and observing how often the correct alignment is recovered. This type of analysis could indicate how many random starts should be used in real-world problems or alternatively, how accurate any initialization heuristic should be.

Chapters 4-7 contain details of how the work described in individual chapters might be extended. There are also interesting possibilities for future work which involve combining ideas from various chapters:

Modeling part-based shapes

In Chapters 6 and 7, ideas from probabilistic pairwise matching (Chapter 5) were used to

develop class models for unordered point sets and contours. Similarly, the work on part-based pairwise matching in Section 5.3 could be used to develop a model for part-based shapes. Part-based models of unordered point sets could be particularly useful in some chemoinformatics problems where shape variation is dominated by spatially localized chemical groups undergoing significant movement relative to the rest of the molecule – *i.e.* when the variation contained in the data set cannot be adequately described using continuous global transformations. Part-based models of contours could be used for recognition of articulated objects in computer vision.

Combining contour and point data

Shapes are often composed of points and lines. In computer vision, we may have access to an object's boundary as well as the position and values of various features – *e.g.* the boundary of a face and the position and color of the eyes. Combining our work on unordered and ordered point sets would allow us to utilize both types of information in a single model.

We note that the above extensions could be achieved in a straightforward and principled fashion due to the consistent use of generative probabilistic models in Chapters 5-7.

Bibliography

Abd-Almageed, W., & Smith, C. (2002). Hidden Markov models for silhouette classification. *Proceedings of the 5th Biannual World Automation Congress 2002.*

Adamek, T., & O'Connor, N. (2003). Efficient contour-basd shape representaion and matching. *5th ACM SIGMM International Workshop on Multimedia Information Retrieval* (pp. 138–143).

Adamek, T., & O'Connor, N. (2004). A mutliscale representation method for nonrigid shapes with a single closed contour. *IEEE Transactions on Circuits and Systems for Video Technology, 14*, 742–753.

Arica, N., & Yarman-Vural, F. (1999). A new HMM topology for shape recognition. *Nonlinear Signal and Image Processing (NSIP)* (pp. 756–760).

Attalla, E., & Siy, P. (2005). Robust shape simlilarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. *Pattern Recognition, 38*, 2229–2241.

Belongie, S., & Malik, J. (2000). Matching with shape contexts. *IEEE Workshop on Content-based Access of Image and Video Libraries* (pp. 20–26).

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 24*, 509–522.

Bennemoun, M. (1994). A contour-based part segmentation algorithm. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

Besl, P., & McKay, N. (1992). A method for the registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 14*, 239–256.

Bicego, M., & Murino, V. (2001). 2d shape recognition by hidden markov models. *The 11th International Conference on Image Analysis and Processing (ICIAP).*

Bicego, M., & Murino, V. (2004). Investigating hidden Markov models' capabilities in 2d shape classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 26*, 281–286.

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

Bishop, C., Svensen, M., & Williams, C. (1998). GTM: The generative topographic mapping. *Neural Computation, 10*, 215–234.

Bookstein, F. L. (1996). Landmark methods for forms without landmarks: morphometrics of group differences in outline shape. *Medical Image Analysis, 1*, 225–243.

Cates, J., Meyer, M., Fletcher, P., & Whitaker, R. (2006). Entropy-based particle systems for shape correspondence. *The 9th International Society and Conference Series on Medical Image Computing and Computer-Assisted Intervention (MICCAI).*

Chui, H., & Rangarajan, A. (2000a). A feature registration framework using mixture models. *Mathematical Methods in Biomedical Image Analysis (MMBIA)* (pp. 190–197).

Chui, H., & Rangarajan, A. (2000b). A new algorithm for non-rigid point matching. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Coats, E. (1998). The comfa steroids as a benchmark data set for development of 3d qsar methods. *Perspectives in Drug Discovery and Design, 12*, 199–213.

Cohen, S. D., & Guibas, L. J. (1999). The earth mover's distance under transformation sets. *The 6th IEEE International Conference on Compter Vision (ICCV)* (pp. 1076–1083).

Cootes, T., & Taylor, C. (1999). A mixture model for representing shape. *Image and Vision Computing, 17*, 567–574.

Cootes, T., Taylor, C., Cooper, D., & Graham, J. (1992). Training models of shape variation from sets of examples. *British Machine Vision Conference (BMVC).* Springer-Verlag.

Costa, L. F., & Cesar, R. M. (2001). *Shape analysis and classification, theory and practice*. CRC Press.

Cox, T., & Cox, M. (2000). *Multidimensional scaling*. Chapman & Hall.

Davies, R. (2002). *Learning shape: Optimal models for analysing shape variability*. Doctoral dissertation, University of Manchester.

Davies, R. H., Twining, C. J., Cootes, T. F., Waterton, J. C., & Taylor, C. J. (2002). A minimum description length approach to statistical shape modeling. *IEEE Transactions on Medical Imaging, 21*, 525–537.

Del Bimbo, A., & Pala, P. (1999). Shape indexing by multi-scale representation. *Image and Vision Computing, 17*, 245–261.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological) (JRSSB), 39*, 1–38.

Dryden, I., Hirst, J., & Melville, J. (2007). Statistical analysis of unlabelled point sets: comparing molecules in chemoinformatics. *Biometrics, 63*, 237–251.

Dryden, I. L., & Mardia, K. V. (1998). *Statistical shape analysis*. Wiley.

Ericsson, A., & Astrom, K. (2003a). An affine invariant deformable shape representation for general curves. *International Conference in Computer Vision (ICCV)*.

Ericsson, A., & Astrom, K. (2003b). Minimizing the description length using steepest descent. *The 14th British Machine Video Conference (BMVC)*.

Ericsson, A., & Karlsson, J. (2006). Benchmarking of algorithms for automatic correspondence of shapes. *British Machine and Video Conference (BMVC)*.

Ghosh, A., & Petkov, N. (2005). Robustness of shape descriptors to incomplete contour representaions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 27*, 1793–1804.

Grigorescu, C., & Petkov, N. (2003). Distance sets for shape filters and shape recognition. *IEEE Transactions on Image Processing, 12*, 1274–1286.

Hladuvka, J., & Buhler, K. (2005). MDL spline models: Gradient and polynomial reparameterisations. *The 16th British Machine Video Conference (BMVC).*

Huttenlocher, D., Klanderman, D., & Rucklige, A. (1993). Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 15*, 850–863.

Jalba, A. C., Wilkinson, M. H. F., & Roerdink, J. B. T. M. (2006). Shape representation and recognition through morphological curvature scale spaces. *IEEE Transactions on Image Processing, 15*, 331–341.

Kent, J. T., Mardia, K. V., & Taylor, C. C. (2004). Matching problems for unlabelled configurations. *Leeds Annual Statistical Research Workshops (LASR).*

Keogh, E., Wei, L., Xi, X., Lee, S., & Vlachos, M. (2006). LB-Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. *32nd International Conference on Very Large Data Bases.*

Kim, D. H., Yun, I. D., & Lee, S. U. (2005). A new shape decomposition scheme for graph-based representation. *Pattern Recognition, 38*, 673–689.

Kotcheff, A. C. W., & Taylor, C. J. (1998). Automatic constuction of eigenshape models by direct optimization. *Medical Image Analysis, 2*, 303–314.

Krishnapuram, B., Bishop, B., & Szummer, M. (2004). Generative Bayesian models for shape recognition. *The Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR-9).*

Latecki, J., & Lakamper, R. (2002). Application of planar shape comparison to object retrieval in image databases. *Pattern Recognition, 35*, 15–29.

Latecki, L. J., Lakämper, R., & Eckhardt, U. (2000). Shape descriptors for non-rigid shapes with a single closed contour. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 424–429).

Ling, H., & Jacobs, D. (2005). Using the inner-distance for classification of ariculated shapes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*, 91–110.

Luo, B., & Hancock, E. (2002). Iterative Procrustes alignment with the EM algorithm. *Image and Vision Computing, 20*, 377–396.

Luo, B., & Hancock, E. (2003). A unified framework for alignment and correspondence. *Computer Vision and Image Understanding, 92*.

Mardia, K. V., Kent, J. T., & Bibby, J. M. (1979). *Multivariate analysis*. Academic Press.

McNeill, G., & Vijayakumar, S. (2005). 2D shape classification and retrieval. *The 19th International Joint Conferences on Artificial Intelligence (IJCAI)* (p. 1483).

McNeill, G., & Vijayakumar, S. (2006a). Hierarchical Procrustes matching for shape retrival. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

McNeill, G., & Vijayakumar, S. (2006b). Part-based probabilistic point matching. *The 18th International Conference on Pattern Recognition (ICPR)* (pp. 382–386).

McNeill, G., & Vijayakumar, S. (2006c). Part-based probabilistic point matching using equivalence constraints. *Advances in Neural Information Processing Systems (NIPS)*.

McNeill, G., & Vijayakumar, S. (2006d). A probabilistic approach to robust shape matching. *International Conference on Image Processing (ICIP)*.

McNeill, G., & Vijayakumar, S. (2007). Linear and nonlinear generative probabilistic class models for shape contours. *The 24th International Conference on Machine Learning (ICML)*.

Meng, X., & Rubin, D. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika, 80*, 267–278.

Milios, E., & Petrakis, E. G. M. (2000). Shape retrieval based on dynamic programming. *IEEE Transactions on Image Processing, 1*, 141–147.

Mokhtarian, F., Abbasi, S., & Kittler, J. (1997). Efficient and robust retrieval by shape content through curvature scale space. In *Image databases and multi-media search*, 51–58. World Scientific.

Mokhtarian, F., & Bober, M. (2003). *Curvature scale space representation: Theory, applications and mpeg-7 standardization.* Kluwer Academic.

Ramsay, J., & Silverman, B. (2005). *Functional data analysis.* Springer.

Rangarajan, A., Chui, H., & Bookstein, F. L. (1997). The softassign Procrustes matching algorithm. *Information Procesing in Medical Imaging* (pp. 29–42). Springer.

Revow, M., Williams, C., & Hinton, G. (1996). Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 18*, 592–606.

Sebastian, T. B., Klein, P. N., & Kimia, B. B. (2003). On aligning curves. *IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI), 25*, 116–125.

Sebastian, T. B., Klein, P. N., & Kimia, B. B. (2004). Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 26*, 550–571.

Shental, N., Bar-Hillel, A., Hertz, T., & Weinshall, D. (2004). Computing Gaussian mixture models with EM using equivalence constraints. In *Advances in neural information processing systems (nips).*

Siddiqi, K., & Kimia, B. B. (1995). Parts of visual form: Computational aspects. *Transactions on Pattern Analysis and Machine Intelligence (PAMI), 17*, 239–251.

Super, B. (2003). Improving object recognition accuracy and speed through non-uniform sampling. *SPIE Conference on Intelligent Robots and Computer Vision XXI: Algorithms, Techniques, and Active Vision* (p. 228).

Super, B. J. (2004a). Fast correspondence-based system for shape-retrieval. *Pattern Recognition Letters, 25*, 217–225.

Super, B. J. (2004b). Learning chance probability functions for shape retrieval or classification. *IEEE Workshop on Learning in Computer Vision and Pattern Recognition (CVPRW).*

Taylor, C., Hill, A., & Brett, A. (2000). A framework for automatic landmark identification using a new method of nonrigid correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 22*, 241–251.

Taylor, C., Mardia, K., & Kent, J. (2003). Matching unlabelled configurations using the EM algorithm. *Leeds Annual Statistical Research Workshops (LASR).*

Thodberg, H. (2003). Minimum description length shape and appearance models. *Information Processing in Medical Imaging (IPMI).*

Thodberg, H., & Olafsdottir, H. (2003). Adding curvature to minimum description length shape models. *The 14th British Machine Video Conference (BMVC).*

Tipping, M., & Bishop, C. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society.*

Titsias, M. (2005). *Unsupervised learning of multiple objects in images.* Doctoral dissertation, University of Edinburgh.

Tu, Z., & Yuille, A. (2004). Shape matching and recognition using generative models and informative features. *The 8th European Conference on Computer Vision (ECCV).*

Twining, C., & Taylor, C. (2001). Kernel principal component analysis and the construction of non-linear active shape models. *British Machine Vision Conference (BMVC).*

Wagener, M., Sadowski, J., & Gasteiger, J. (1995). Autocorrelation of molecular surface properties for modeling corticosteroid binding globulin and cytosolic ah receptor activity by neural networks. *Journal of the American Chemical Society, 117*, 7769 – 7775.

Wang, F., Vemuri, B., Rangarajan, A., Schmalfuss, I., & Eisenschenck, S. (2006). Simultaneous registration of multiple point-sets and atlas construction. *The 9th European Conference on Computer Vision (ECCV).*

Wang, S., Kubota, T., & Richardson, T. (2004). Shape correspondence through landmark sliding. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Williams, C. (1994). *Combining deformable models and neural networks for handprinted digit recognition.* Doctoral dissertation, University of Toronto.

Zhang, J., Zhang, X., Krim, H., & Walter, G. G. (2003). Object representation and recognition in shape spaces. *Pattern Recognition, 36*, 1143–1154.