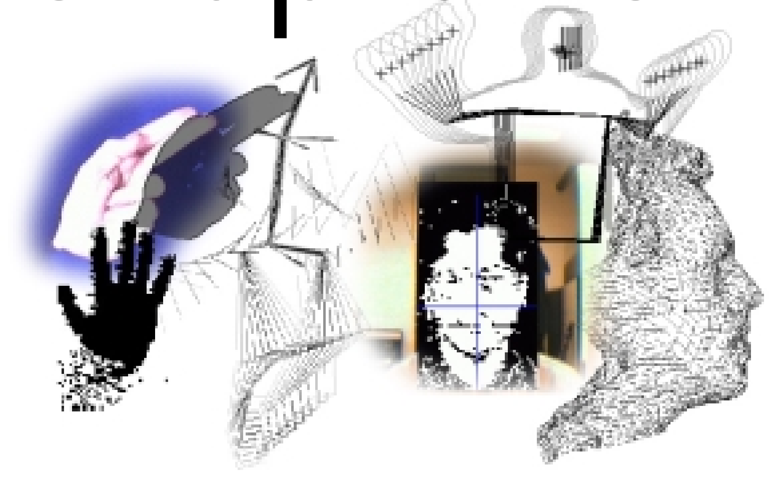
Chapter 3



3 Linear Point Distribution Models

3.1 Introduction

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

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

3.2 Constructing a Point Distribution Model

3.2.1 Overview

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

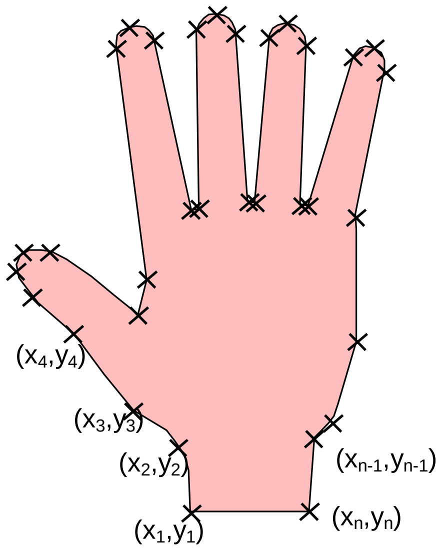


Figure 3.2.1 - 2D Contour of a hand

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

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

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

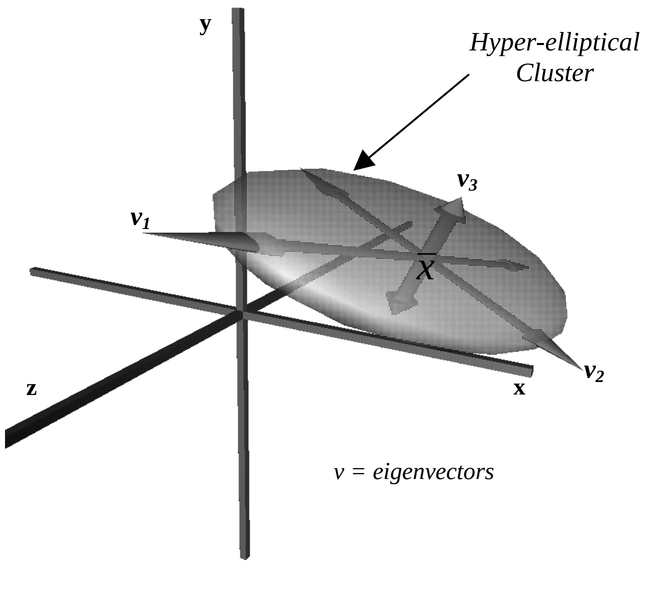


Figure 3.2.2 - Hyper-elipsoid in n Dimensional Space

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

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

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

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

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

3.2.2 Obtaining Training Examples

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

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

Other approaches to the automatic and semiautomatic generation of training examples are the use of snakes [Kass 88] to segment simple deformable objects from image sequences. In a temporal image sequence the pose of a converged snake can be used as an initial estimate for the next frame reducing the susceptibility of snakes to their initial location. Cootes et al have also proposed using the PDM itself to locate new objects by bootstrapping the procedure and using a partially 'learnt' PDM to constrain segmentation of future models.

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

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

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

3.2.3 Landmark Point Assignment

In order to perform statistical analysis on a training set the procedure assumes a single cluster is formed in shape space by the training set. This assumption works on the principle that common points along the contour boundary do not change between examples. Similar shapes therefore produce similar vectors which occupy a tight cluster in shape space. However, in order for this assumption to hold true, consistent points along the contour must be located.

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

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

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

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

3.2.4 Training Set Alignment

Cootes *et al* suggested aligning training examples by calculating the scaling, translation and rotation of each model to minimise the sum of the squares of distances between equivalent points for all examples. This exhaustive process although suitable for simple 2D contours of low dimensionality does not provide a suitable approach for more complex high dimensional objects.

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

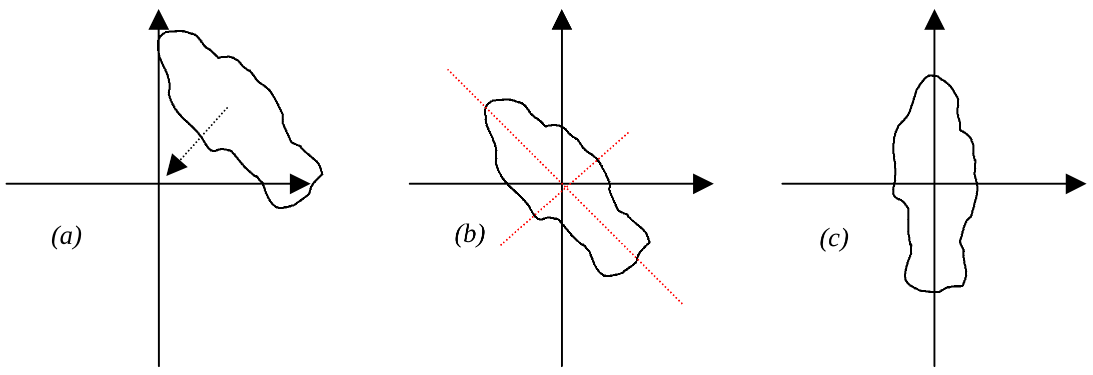


Figure 3.2.3 - Aligning the training set

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

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

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

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

3.2.5 Learning Shape Space

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

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

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

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

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

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

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

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

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

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

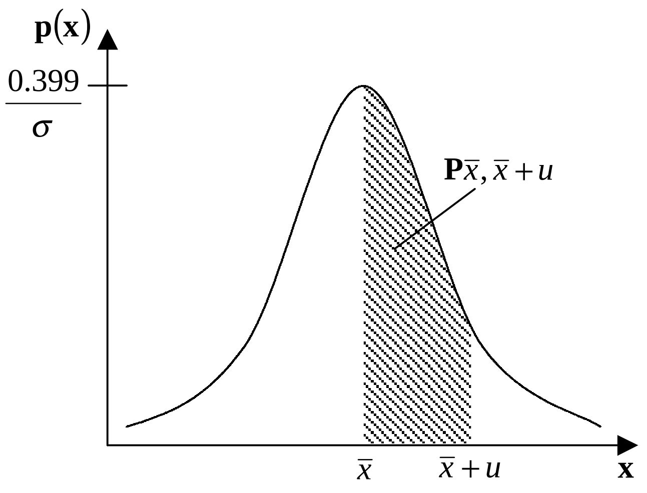


Table 3.2-1 - The area probability under a normalised Gaussian distribution

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

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

is rewritten as

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

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

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

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

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

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

then

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

premultiplying by **D yields**

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

and therefore

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

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

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

with corresponding eigenvalues $\lambda_i = \gamma_i$.

3.2.6 Human Head Example

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

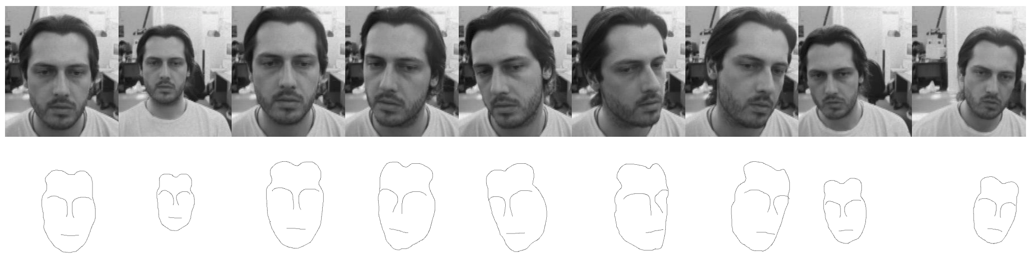


Figure 3.2.4 - Training Examples for 2D Head PDM

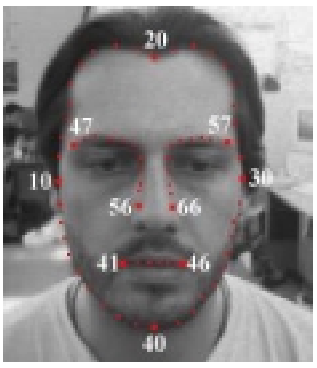


Figure 3.2.5 - Landmark points of the 2D Head PDM

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

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

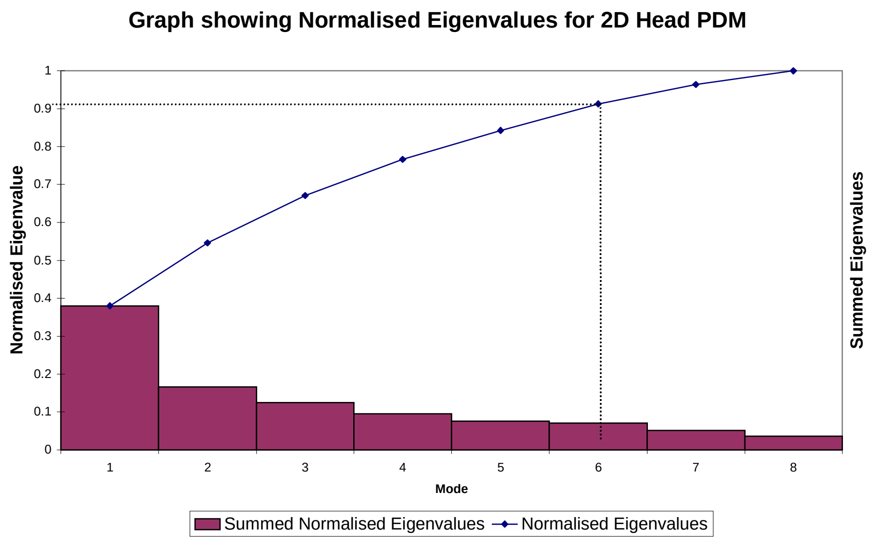


Figure 3.2.6 - Graph showing Normalised Eigenvalues for the 2D Head PDM

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

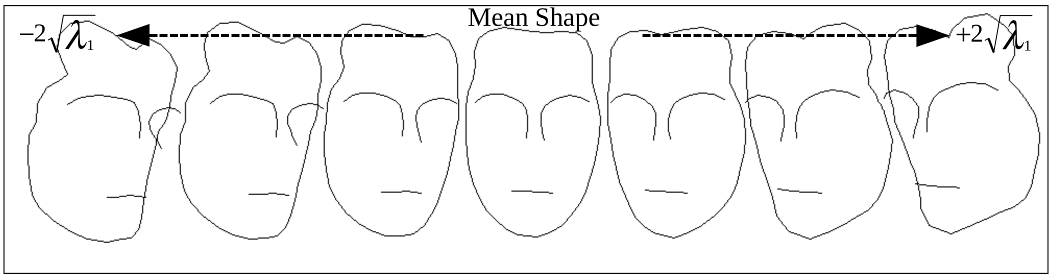


Figure 3.2.7 - Primary mode of the 2D Head PDM

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

3.3 Active Shape Models

3.3.1 Overview

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

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

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

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

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

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

The ASM tracking algorithm can be summarised as

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

Each of these steps will now be considered in turn.

3.3.2 ASM Initialisation

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

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

3.3.3 Feature Detection

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

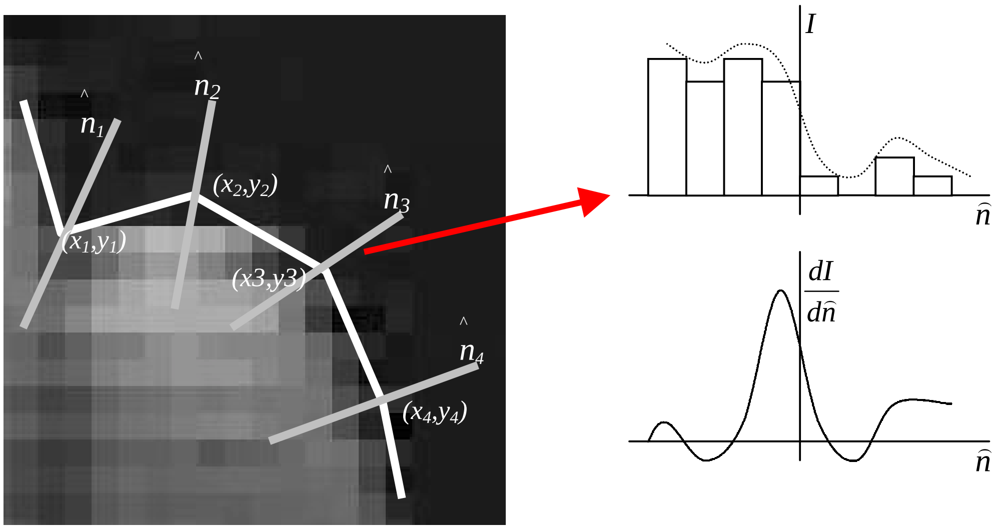


Figure 3.3.1 - Local edge detection along boundary normals

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

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

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

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

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

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

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

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

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

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

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

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

3.3.4 Iterative Refinement

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

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

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

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

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

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

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

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

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

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

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

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

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

Equation 3.3-4
$$d\mathbf{x} = M((s(1+ds))^{-1}, -(\theta + d\theta))[M(s,\theta)[\mathbf{x}] + d\mathbf{X} - d\mathbf{X}_c] - \mathbf{x}$$

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

From Equation 3.2-4
$$\mathbf{x} + d\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{P}(\mathbf{b} + d\mathbf{b})$$
therefore
$$d\mathbf{b} = \mathbf{P}^{-1}d\mathbf{x}$$
or $d\mathbf{b} = \mathbf{P}^T d\mathbf{x}$ since $\mathbf{P}^T \equiv \mathbf{P}^{-1}$, as the columns of **P** are mutually orthogonal and of unit length [Cootes 95].

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

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

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

3.4 Reconstructive Ability

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

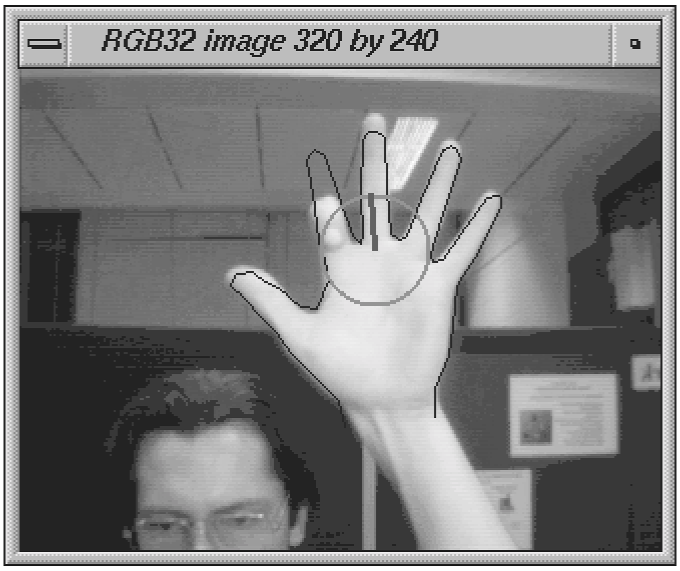


Figure 3.4.1 - Constrained PDM tracking hand

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

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

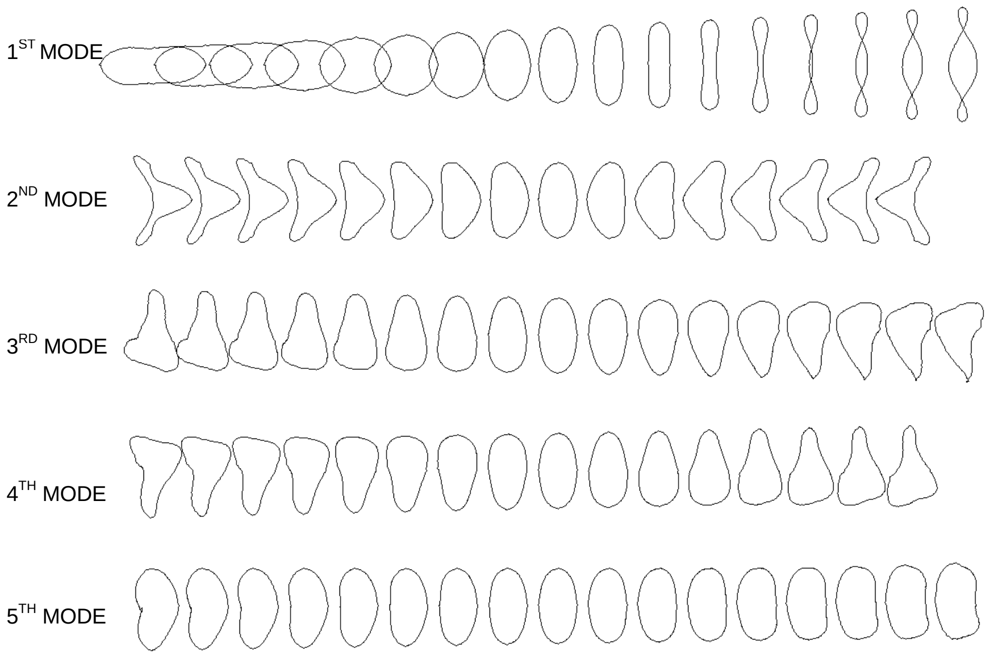


Figure 3.4.2 - First Five Modes of variation of the leaf PDM

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

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

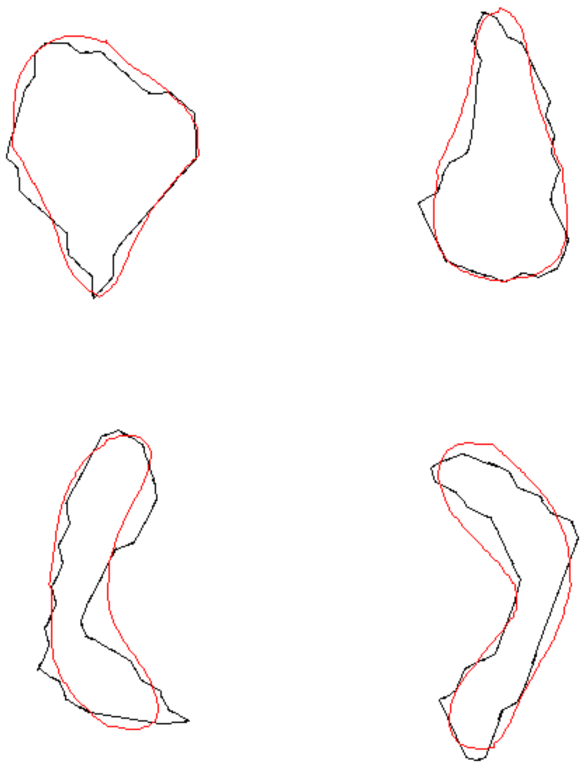


Figure 3.4.3 - Training examples and the reconstructed shape using 9 modes of variation

Although this smoothing is a lossy compression technique, the information that is discarded is of little use. This is due to small leaf samples where their extraction resulted in blobs of the order of tens of pixels rather than hundreds. The resulting boundary is heavily ‘step-like’ due to the pixelisation of the shape. During re-sampling, bilinear interpolation results in the boundary being smoothed into unrepresentative shapes which are indicative of the modality used, and not the actual leaf sample. By using the minimum number of modes to reconstruct the shape, the errors introduced into the shape by the image size are discarded and a better estimation of shape provided. Figure 3.4.4 shows a small leaf sample, with the interpolated/resampled boundary and the resulting smoothing which comes from PDM reconstruction. It should be noted that the smoothed boundary produced by the PDM goes some way to reconstructing the information lost during acquisition. This is due to the statistical nature of the PDM and its knowledge of what a leaf '*should look like*'.

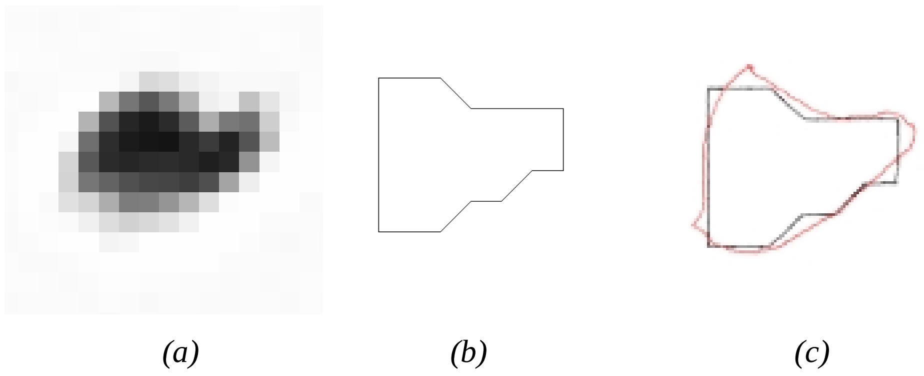


Figure 3.4.4 - Training examples and the reconstructed shape using 9 modes
(a) Original Image of leaf (b) resampled boundary of leaf (c) reconstructed boundary of leaf

3.5 Conclusions

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