Chapter 2
Statistical Models of Shape and Appearance

As was explained in the Introduction, our aim is to start with a set of example shapes (the *training set*), and learn from this the patterns of variability of the shape of the class of objects for which the training set can be considered a representative sample. We will first consider this problem in a rather abstract sense, and illustrate how the question of correspondence *between* different shapes is inextricably linked to the question of representing a set of shapes.

Mathematically, the usual approach is to construct a mapping from an example shape to a point in some shape space. This is the process of constructing a *representation* of shape. The idea is that every physical shape corresponds to a point in shape space, and conversely, each point in shape space corresponds to some physical shape. There are many ways of constructing such representations, but whichever method is chosen, what is then obtained is a mapping from our training set of shapes to a set of points in shape space.

Modelling can then be considered as the process of modelling the distribution of our training points in shape space. However, before we can begin to talk about the distribution of such points, we first need to define a notion of distance on shape space.

A definition of a distance on shape space then leads directly to the notion of correspondence between the physical shapes themselves. Consider two distinct physical shapes, and the two points in shape space which represent those two shapes. We can then imagine a continuous path between the two points in shape space, and, given that we have a definition of distance, the shortest such path between the two shapes. When we map this construction back to the space of physical shapes, what we obtain is a continuous sequence of physical shapes that interpolates between our two original shapes. If we now consider a single point on one shape, we can then follow it through this continuous sequence of shapes, and hence locate the physical point on the other shape to which this point corresponds. This is what is meant by a dense correspondence between shapes.

Let us now return to our physical shapes, and imagine translating and rotating a physical shape (that is, altering the *pose* of the shape). In many

R. Davies et al., *Statistical Models of Shape*,
DOI: 10.1007/978-1-84800-138-1_2,

cases, the pose of the physical shape is unimportant, and what we mean by *change* of shape is not such a transformation.

But there is another transformation we could consider. Suppose we imagine two *distinct* points in shape space, which give the same *physical* shape, but different correspondence when compared to some other (reference) shape (using the construction defined above). If such a construction is possible, we see that it is possible (at least in theory) to manipulate the correspondence between shapes (moving the point in shape space), whilst leaving the physical shape unaltered. Which then means that we have to answer the question as to what correspondence we should use for our analysis of shape variability (the *correspondence problem* – see also Sect. 3.1).

We hence see that the issue of shape correspondence naturally arises as soon as we consider the steps necessary to represent a set of shapes, and analyse their distribution. Some methods of shape representation do not allow correspondence to be manipulated independently of shape, and in these cases, the correspondence they generate can be considered as *implicit* (for example, the SPHARM method [76], or early M-Rep methods [137]). However, there are other methods of shape representation for which the correspondence is *explicit*, which allow correspondence to be manipulated independently of physical shape.

In the remainder of this book, we will restrict ourselves to such a shape representation, the shape representation which leads to the class of deformable models known as *Statistical Shape Models*[1] (SSMs). We will now describe this shape representation in detail, beginning with the finite-dimensional case.

2.1 Finite-Dimensional Representations of Shape

Let us consider first building a finite-dimensional representation of a single shape S. The most intuitive and simplest way to represent such a shape is a join-the-dots approach.

We take a set of n_P points which lie on the shape S, with positions:

$$\mathbf{x}^{(i)} \in S, \;\; i = 1, \ldots n_P. \tag{2.1}$$

The coordinates of each point position can be concatenated to give a single *shape vector* $\mathbf{x} = \{\mathbf{x}^{(i)}\}$. For example:

$$\mathbf{x} \doteq (x^{(1)}, y^{(1)}, z^{(1)}, x^{(2)}, y^{(2)}, z^{(2)}, \ldots, x^{(n_P)}, y^{(n_P)}, z^{(n_P)}), \;\; S \subset \mathbb{R}^3, \tag{2.2}$$

[1] Note that these were initially called *Point Distribution Models (PDMs)*. However, due to a clash with nomenclature in the statistics literature, they were later re-christened *Statistical Shape Models (SSMs)*. Both terms can be found in the literature.

where $(x^{(i)}, y^{(i)}, z^{(i)})$ are the Cartesian coordinates of the i^{th} point on the shape. For a shape S in $\mathbb{R}^d$, this gives a $d \times n_P$-dimensional representation. In most cases, Cartesian coordinates are sufficient, but in cases where parts of shapes can rotate, it may be useful to instead use angular coordinates [87].

The final representation of the shape is then generated from the shape vector by interpolation. For shapes in $\mathbb{R}^2$ (curves), this is a simple join-the-dots approach, using either straight-line segments (polygonal representation), or by spline interpolants if a smoother shape is preferred. For shapes in $\mathbb{R}^3$ (surfaces), interpolants can similarly also be linear (planes), or a higher-order spline interpolant.

What we have not considered so far is the connectivity of the points, and the topology of the shape. For the case of shapes in $\mathbb{R}^2$, the simplest case is where the shape has only one connected component, with the topology of either an open or closed line. The points are usually numbered so that they are connected consecutively – for the closed shapes, we must also form a loop by connecting the last point to the first. For more complicated multi-part shapes, the points which are members of each part, and the connectivity within each part have to be specified separately.

Similar considerations holds for shapes in $\mathbb{R}^3$. The simplest case is then single-part shapes in $\mathbb{R}^3$, with the topology of either open surfaces or spheres, with the points being part of a triangulated mesh.

Once we have a finite-dimensional representation of a single shape S, we can easily see how this can be extended to form a *common* representation of a *set* of shapes. To be specific, let us take a set of n_S shapes:

$$S_i : i = 1, \ldots n_S. \tag{2.3}$$

We suppose that each shape is then represented by a set of n_P points, such that the individual points are placed in *corresponding* positions across the set of shapes. This then gives us a set of initial shape vectors $\{\mathbf{x}_i : i = 1, \ldots n_S\}$ which form a representation of the whole set of shapes in a common shape space $\mathbb{R}^{dn_P}$.

2.1.1 Shape Alignment

In many cases, the size, placement, and orientation of an object is arbitrary, and has nothing to do with the actual variation of shape that we are interested in. In mathematical terms, there are degrees of freedom (scaling, translation, and rotation) associated with each shape example, which we wish to factor out of our shape analysis.

Consider a fixed shape $\mathbf{y}$, and a second moving shape $\mathbf{x}$, which we wish to align with the first by means of a similarity transformation. A general similarity transformation acting on $\mathbf{x}$ can be written as:

$$\mathbf{x} \mapsto s\mathbf{R}(\mathbf{x} - \mathbf{t}), \tag{2.4}$$

where $\mathbf{t}$ represents a translation in $\mathbb{R}^d$, $\mathbf{R}$ is a $dn_P \times dn_P$ representation of a rotation in $\mathbb{R}^d$, and $s \in \mathbb{R}^+$ is a scaling. Note that these elements of the representation of a similarity transformation are such that they act on the concatenated set of shape points in the shape vector. They are constructed from a representation that acts on single points in the obvious way, although the exact details depend on the way in which the coordinates of the shape points have been concatenated.

We wish to find the similarity transformation which brings the moving shape $\mathbf{x}$ as close as possible to the fixed shape $\mathbf{y}$. The simplest way to define proximity is just the magnitude of the Euclidean norm of the difference between the two shape vectors in $\mathbb{R}^{dn_P}$:

$$\mathcal{L} \doteq \|\mathbf{y} - s\mathbf{R}(\mathbf{x} - \mathbf{t})\|^2, \tag{2.5}$$

which is the square of the Procrustes distance between the shapes [78]. In terms of the positions of individual points, this expression can be rewritten as:

$$\mathcal{L} = \sum_{i=1}^{n_P} \|\mathbf{y}^{(i)} - s\mathbf{R}(\mathbf{x}^{(i)} - \mathbf{t})\|^2, \tag{2.6}$$

where $\mathbf{t}$ is now just a vector in $\mathbb{R}^d$, and $\mathbf{R}$ is a $d \times d$ rotation matrix.

If we define our origin so that it lies at the centre of mass of the fixed shape:

$$\frac{1}{n_P} \sum_{i=1}^{n_P} \mathbf{y}^{(i)} = \mathbf{0}, \tag{2.7}$$

with rotation defined about this origin, the optimal translation can then be calculated as:

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{t}}\right|_{s,\mathbf{R}} = 0 \implies \left.\frac{\partial}{\partial \mathbf{t}}\right|_{s,\mathbf{R}} \sum_{i=1}^{n_P} \|s\mathbf{R}(\mathbf{x}^{(i)} - \mathbf{t})\|^2 = \mathbf{0}, \tag{2.8}$$

$$\implies \left.\frac{\partial}{\partial \mathbf{t}}\right|_{s,\mathbf{R}} \sum_{i=1}^{n_P} \|(\mathbf{x}^{(i)} - \mathbf{t})\|^2 = \mathbf{0}, \tag{2.9}$$

$$\implies \boxed{\mathbf{t} = \frac{1}{n_P} \sum_{i=1}^{n_P} \mathbf{x}^{(i)}} = \frac{1}{n_P} \sum_{i=1}^{n_P} \left(\mathbf{x}^{(i)} - \mathbf{y}^{(i)}\right). \tag{2.10}$$

That is, the centroid/centre of mass of the original moving shape is translated so that it coincides with the centre of mass of the fixed shape.

Once the shapes have been centred, we can then calculate the combined scaling and rotation:

$$\frac{\partial \mathcal{L}}{\partial s\mathbf{R}} = 0 \implies \boxed{\sum_{i=1}^{n_P} y_\mu^{(i)} x_\beta^{(i)} = sR_{\mu\alpha} \sum_{j=1}^{n_P} x_\alpha^{(j)} x_\beta^{(j)},} \tag{2.11}$$

where $\mathbf{x}^{(i)} = \{x_\alpha^{(i)} : \alpha = 1, \dots d\}$ and $\mathbf{y}^{(i)} = \{y_\alpha^{(i)} : \alpha = 1, \dots d\}$ are the Cartesian components of the point positions. This can then be solved for the matrix $s\mathbf{R}$ (for example, see [102] for further details).

Rather than aligning just a pair of shapes, we wish to mutually align an entire set of shapes $\{S_i : i = 1, \dots n_S\}$, $S_i = \{\mathbf{x}_j^{(i)} : j = 1, \dots n_P\}$. We use a similar criterion to that considered above, either by considering the squared Procrustes distances between all pairs of shapes, or between all shapes and the mean shape. This is known as generalized Procrustes analysis. The translations are obtained as before, centering each shape on the origin. However, the general problem of finding the optimal rotations and scalings is not well-posed unless further constraints are placed on the mean [174], as will be explained below.

For statistical shape analysis and statistical shape models, a simple iterative approach is usually sufficient. After first centering all the shapes, a typical algorithm then proceeds [38] as Algorithm 2.1.

Algorithm 2.1 : Mutually Aligning a Set of Shapes.

Initialize:

- Choose one shape as the reference frame, call it $\mathbf{x_{ref}}$, and retain this.
- Normalize the scale so that $\|\mathbf{x_{ref}}\| = \mathbf{1}$.
- Set the initial estimate of the mean shape to be $\mathbf{x_{ref}}$.

Repeat:

- Perform pairwise alignment of all shapes to the current estimate of the mean shape.
- Recompute the mean of the set of shapes:

  $\bar{\mathbf{x}} \doteq \{\bar{\mathbf{x}}^{(i)} : i = 1, \dots n_P\}$, $\bar{\mathbf{x}} \doteq \dfrac{1}{n_S} \displaystyle\sum_{j=1}^{n_S} \mathbf{x_j}$.
- Align $\bar{\mathbf{x}}$ to the initial reference frame $\mathbf{x_{ref}}$.
- Normalize the mean so that $\|\bar{\mathbf{x}}\| = 1$.

Until convergence.

Note that it is necessary to retain the initial reference frame to remove the global degree of freedom corresponding to rotating all the shapes by the same amount. Setting $\|\bar{\mathbf{x}}\| = 1$ similarly removes the degree of freedom associated with scaling all the shapes by the same factor. The degrees of freedom associated with a uniform translation have already been removed by centering all the shapes before we began the rest of the alignment.

There remains the question of what transformations to allow during the iterative refinement. A common approach is to scale all shapes so that $\|\mathbf{x}_i\| = 1$, and allow only rotations during the alignment stage. This means that from the original shape space $\mathbb{R}^{dn_P}$, all shapes have been projected onto the surface of a hypersphere $\|\mathbf{x}\| = 1$. This means that the submanifold of $\mathbb{R}^{dn_P}$ on which the aligned shapes lie is curved, and if large shape changes occur, significant non-linearities can appear. This may be problematic when we come to the next stage of building a statistical model of the distribution of shapes. An alternative is to allow both scaling and rotation during alignment, but this can also introduce significant non-linearities. If this is a problem, the non-linearity can be removed by projecting the aligned shapes onto the tangent hyperplane to the hypersphere at the mean shape. That is:

$$\mathbf{x}_i \mapsto s_i \mathbf{x}_i, \;\; s_i \in \mathbb{R}^+ \text{ such that } (\bar{\mathbf{x}} - s_i \mathbf{x}_i) \cdot \bar{\mathbf{x}} = 0. \tag{2.12}$$

See [38] for further details and explicit examples.

2.1.2 Statistics of Shapes

To summarize our progress so far, we have mapped our initial shape vectors (2.2) in $\mathbb{R}^{dn_P}$ to a new set of mutually aligned shape vectors, by factoring out uninteresting degrees of freedom corresponding to pose (scale, orientation, and position). We now wish to analyse the statistics of this distribution of shape vectors. To do this, we first need to find a set of axes specific to the particular set of shapes. We have in some sense already started to perform this, since we have a mean shape $\bar{\mathbf{x}}$ that can be used as an origin.

To see that this is a necessary procedure, consider the extreme case where there is a shape point, $\mathbf{x}^{(i)}$ say, which does not change its position across the set of examples. Since this point does not vary, there is no value in retaining the axes corresponding to the coordinates of this point $\{x_\alpha^{(i)} : \alpha = 1, \dots d\}$. We wish instead to find a new set of axes in $\mathbb{R}^{dn_P}$ that span the subspace which contains the (aligned) shapes. One simple procedure for performing this task is Principal Component Analysis (PCA).

2.1.3 Principal Component Analysis

We start from our set of shape vectors $\{\mathbf{x}_i : i = 1, \dots n_S\}$ (we will assume from now on that we are only considering sets of shape vectors which have been aligned), with components relative to our original axes:

$$\mathbf{x}_i = \{x_{i\mu} : \mu = 1, \dots d \times n_P\}. \tag{2.13}$$

These are the components and axes defined by those in $\mathbb{R}^d$, the original space in which the input shapes reside.

We wish to find a new set of orthogonal axes in $\mathbb{R}^{dn_P}$ that better reflects the actual distribution of the set. The origin of this new set of axes will be set to the mean shape $\bar{\mathbf{x}}$. Let these new axes be described by a set of orthonormal vectors:

$$\{\mathbf{n}^{(a)}\} \text{ such that } \mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} = \delta_{ab}, \tag{2.14}$$

where δ_{ab} is the Kronecker delta.

We then have the following theorem:

Theorem 2.1. PCA.
The set of orthonormal directions $\{\mathbf{n}^{(a)}\}$ *that maximises the quantity:*

$$\mathcal{L} \doteq \sum_a \sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right)^2, \tag{2.15}$$

are given by the eigenvectors of the data covariance matrix $\mathbf{D}$ *for the shapes, where we define* $\mathbf{D}$ *of size* $dn_P \times dn_P$ *with components:*

$$\boxed{D_{\mu\nu} \doteq \sum_{i=1}^{n_S} (\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_i - \bar{\mathbf{x}})_\nu.} \tag{2.16}$$

Then the eigenvectors are defined by:

$$\mathbf{D}\mathbf{n}^{(a)} = \lambda_a \mathbf{n}^{(a)}, \;\; a = 1, \ldots n_S - 1. \tag{2.17}$$

Proof. Suppose we are extracting these vectors in some sequential manner, so that having found an acceptable subset $\{\mathbf{n}^{(a)} : a = 1, \ldots b-1\}$*, we now wish to make the optimum choice of the next vector* $\mathbf{n}^{(b)}$*. Optimality is then given by maximising:*

$$\mathcal{L} \doteq \sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(b)} \right)^2, \tag{2.18}$$

with respect to $\mathbf{n}^{(b)}$*, subject to the orthonormality constraints:*

$$\mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} = \delta_{ab}, \;\; a = 1, \ldots b. \tag{2.19}$$

Using Lagrange multipliers $\{c_{ba} : a = 1, \ldots b\}$*, the solution to this constrained optimisation problem corresponds to the stationary point of the function:*

$$\mathcal{L} = \sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(b)} \right)^2 + \sum_{a=1}^{b-1} c_{ba} \mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} + c_{bb} \left(\mathbf{n}^{(b)} \cdot \mathbf{n}^{(b)} - 1 \right). \quad (2.20)$$

$$\frac{\partial \mathcal{L}}{\partial c_{ba}} = 0 \implies \mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} = \delta_{ab}, \text{ which are the required constraints.} \quad (2.21)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}^{(b)}} = 0 \implies 2 \sum_{i=1}^{n_S} (\mathbf{x}_i - \bar{\mathbf{x}})_\nu (\mathbf{x}_i - \bar{\mathbf{x}})_\mu n_\mu^{(b)} + \sum_{a=1}^{b-1} c_{ba} n_\nu^{(a)} + 2 c_{bb} n_\nu^{(b)} = 0, \quad (2.22)$$

where we use the Einstein summation convention[2] *that the repeated index* μ *is summed from* $\mu = 1$ *to* dn_P*. Using the definition of the covariance matrix* $\mathbf{D}$ *(2.16), we can rewrite the condition as:*

$$2\mathbf{D}\mathbf{n}^{(b)} + \sum_{a=1}^{b-1} c_{ba} \mathbf{n}^{(a)} + 2 c_{bb} \mathbf{n}^{(b)} = \mathbf{0}. \quad (2.23)$$

For the case $b = 1$ *(the first direction we choose), this reduces to:*

$$\mathbf{D}\mathbf{n}^{(1)} + c_{11} \mathbf{n}^{(1)} = 0 \quad (2.24)$$

$$\implies \boxed{\mathbf{D}\mathbf{n}^{(1)} = \lambda_1 \mathbf{n}^{(1)}} \;\&\; \mathbf{n}^{(1)}\mathbf{D} = \lambda_1 \mathbf{n}^{(1)}, \; c_{11} \doteq \lambda_1. \quad (2.25)$$

That is, the vector $\mathbf{n}^{(1)}$ *is a left and right eigenvector of the (symmetric) shape covariance matrix* $\mathbf{D}$*, with eigenvalue* λ_1*. The condition for the second axis can then be written as:*

$$2\mathbf{D}\mathbf{n}^{(2)} + c_{21} \mathbf{n}^{(1)} + 2 c_{22} \mathbf{n}^{(2)} = \mathbf{0}. \quad (2.26)$$

Taking the dot product of this expression with $\mathbf{n}^{(1)}$*, we obtain:*

$$2\mathbf{n}^{(1)}\mathbf{D}\mathbf{n}^{(2)} + c_{21} = 0 \quad (2.27)$$

$$\implies 2\lambda_1 \mathbf{n}^{(1)} \cdot \mathbf{n}^{(2)} + c_{21} = 0 \implies c_{21} = 0. \quad (2.28)$$

$$\therefore \; \mathbf{D}\mathbf{n}^{(2)} + c_{22} \mathbf{n}^{(2)} = \mathbf{0} \implies \boxed{\mathbf{D}\mathbf{n}^{(2)} = -c_{22} \mathbf{n}^{(2)} \doteq \lambda_2 \mathbf{n}^{(2)}.} \quad (2.29)$$

It then follows by induction that the required set of axes $\{\mathbf{n}^{(a)}\}$ *are the orthonormal set of eigenvectors of the shape covariance matrix* $\mathbf{D}$*.* □

The sum of the squares of the components of the shape vectors along each of the PCA directions $\mathbf{n}^{(a)}$ is then given by:

[2] Note that, in general, indices that appear in brackets $.^{(a)}$ will *not* be summed over unless explicitly stated. See Glossary.

$$\sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right)^2 = n_\mu^{(a)} D_{\mu\nu} n_\nu^{(a)} = \lambda_a \geq 0. \tag{2.30}$$

This means that the set of axes can be ordered in terms of relative importance by sorting the eigenvalues in terms of decreasing size. Since there are n_S shapes, there are at most $n_S - 1$ non-zero eigenvalues. This means that for the case $n_S - 1 < dn_P$, we have performed dimensionality reduction by locating the directions with zero eigenvalue that are orthogonal to the subspace spanned by the data.

In practice, we retain not just the directions corresponding to non-zero eigenvalues, but instead that ordered set which encompasses a certain amount of the total variance of the data.

$$\text{Ordered set of eigenvalues:} \quad \lambda_1 \geq \lambda_2, \ldots \geq \lambda_{dn_P}, \tag{2.31}$$

$$\text{Total variance:} \quad \sum_{a=1}^{n_S - 1} \lambda_a, \tag{2.32}$$

$$\text{Variance up to } n_m: \quad \sum_{a=1}^{n_m} \lambda_a. \tag{2.33}$$

The number of modes n_m retained is then chosen to be the lowest value such that the variance up to n_m is some specified fraction of the total variance.

We can also transform coordinates to the system defined by the directions $\{\mathbf{n}^{(a)}\}$, with origin $\bar{\mathbf{x}}$. For each shape $\mathbf{x}_i$ this then defines a new vector of shape parameters $\mathbf{b}^{(i)} \in \mathbb{R}^{n_m}$ thus:

$$\mathbf{b}^{(i)} = \{b_a^{(i)} : a = 1, \ldots n_m\}, \;\; b_a^{(i)} \doteq \left(\mathbf{n}^{(a)} \cdot (\mathbf{x}_i - \bar{\mathbf{x}}) \right), \tag{2.34}$$

where the covariance in this frame is now given by the diagonal matrix:

$$\boxed{D_{ab} \doteq \sum_{i=1}^{n_S} (\mathbf{n}^{(a)} \cdot \mathbf{b}^{(i)})(\mathbf{n}^{(b)} \cdot \mathbf{b}^{(i)}) = \lambda_a \delta_{ab}.} \tag{2.35}$$

We define the matrix of eigenvectors:

$$\boxed{\mathbf{N}, \;\; N_{\mu a} \doteq n_\mu^{(a)},} \tag{2.36}$$

which is then of size $dn_P \times n_m$. We can then form an approximate reconstruction of the shape vector $\mathbf{x}_i$ from the corresponding parameter vector $\mathbf{b}^{(i)}$ thus:

$$\mathbf{x}_i \approx \bar{\mathbf{x}} + \mathbf{N}\mathbf{b}^{(i)}. \tag{2.37}$$

The reconstruction is only approximate, since we have only retained the first n_m eigenvectors, rather than all eigenvectors with non-zero eigenvalue.

The matrix $\mathbf{N}$ performs a mapping from the coordinate axes defined in (shape) parameter space to the original shape space. The mean shape simply performs a translation of the origin, since the origin of parameter space is taken to correspond to the mean shape. The corresponding backwards mapping, from shape space to parameter space, is performed by the matrix $\mathbf{N}^T$. For a general parameter vector $\mathbf{b} \in \mathbb{R}^{n_m}$ and shape vector $\mathbf{x} \in \mathbb{R}^{dn_P}$:

$$\boxed{\mathbf{b} \mapsto \bar{\mathbf{x}} + \mathbf{N}\mathbf{b}, \;\; \mathbf{x} \mapsto \mathbf{N}^T (\mathbf{x} - \bar{\mathbf{x}}) .} \tag{2.38}$$

Note however that the mappings are not the inverse of each other, even if all the variance is retained, since the dimensionality of parameter space is less than the dimensionality of shape space. For a shape vector $\mathbf{x}$ which is not part of the original training set, the action of $\mathbf{N}^T$ first projects the shape vector into the subspace spanned by the training set, then forms an (approximate) representation of this using the n_m available modes.

If we suppose that the parameter vectors for our original set of shapes are drawn from some probability distribution $p(\mathbf{b})$, then we can sample parameter vectors $\mathbf{b}$ from this distribution. We can then construct the corresponding shapes for each parameter vector $\mathbf{b}$ as above (2.38). This gives us an arbitrarily large set of generated shapes, sharing the same distribution as the original set. This is usually referred to as applying the SSM in a generative mode.

The remaining task is to learn this distribution $p(\mathbf{b})$, given our original set of shapes – in this context, we refer to this set as a *training set*.

2.2 Modelling Distributions of Sets of Shapes

For a simple unimodal distribution of shapes in shape space, PCA generates a coordinate system centred on the distribution, whose axes are aligned with the significant directions of the distribution, and represent modes of variation of that data. If the distribution is not simple, PCA will still enable us to discard dimensions which are orthogonal to the data, that is, perform dimensional reduction. The individual directions $\mathbf{n}^{(a)}$ will not however necessarily correspond to modes of variation of the data.

In the following sections, we consider various methods for studying and representing the distribution of the training data in shape space. We start with the simplest case of a single multivariate Gaussian, where the data is unimodal and the PCA axes do correspond to real modes of variation of the input data. For the case of multimodal or non-linear data distributions, we discuss two types of kernel methods, the classical method of kernel density estimation, and the more recent technique of kernel principal component analysis.

2.2.1 Gaussian Models

We will consider modelling the distribution of the data by a multivariate Gaussian. Having already applied PCA, we now model the parameter space containing the vectors $\{\mathbf{b}^{(i)}\}$ defined above (2.34).

We consider a multivariate Gaussian distribution centred on the origin in parameter space, that is, centred on the mean of the data in shape space.

Theorem 2.2. Maximum Likelihood Method.
Consider a centred Gaussian probability density function (pdf) of the form:

$$p(\mathbf{b}) \propto \left(\prod_{c=1}^{n_m} \frac{1}{\sigma_c}\right) \exp\left(-\frac{1}{2}\sum_{a=1}^{n_m}\left(\frac{\mathbf{b}\cdot\mathbf{m}^{(a)}}{\sigma_a}\right)^2\right), \tag{2.39}$$

where $\{\mathbf{m}^{(a)} : a = 1, \ldots n_m\}$ *are some orthonormal set of directions:*

$$\mathbf{m}^{(a)} \cdot \mathbf{m}^{(b)} = \delta_{ab}, \tag{2.40}$$

and $\{\sigma_a\}$ *are the set of width parameters. The fitted Gaussian which maximises the quantity:*

$$\prod_{i=1}^{n_S} p(\mathbf{b}^{(i)}), \tag{2.41}$$

is then given by $\{\mathbf{m}^{(a)}\}$ *equal to the eigenvectors of the covariance matrix of* $\{\mathbf{b}^{(i)}\}$*. If these eigenvectors have corresponding eigenvalues* $\{\lambda_a\}$*, then the optimum width parameters are:*

$$\sigma_a^2 = \frac{1}{n_S}\lambda_a. \tag{2.42}$$

Proof. We are required to maximise:

$$\prod_{i=1}^{n_S} p(\mathbf{b}^{(i)}). \tag{2.43}$$

Equivalently, we can maximise instead the logarithm of this:

$$\mathcal{L} = -n_S \sum_{c=1}^{n_m} \ln \sigma_c - \frac{1}{2}\sum_{i=1}^{n_S}\sum_{a=1}^{n_m}\left(\frac{\mathbf{b}^{(i)}\cdot\mathbf{m}^{(a)}}{\sigma_a}\right)^2 + \textit{(constant terms)}, \tag{2.44}$$

with the orthonormality constraints as above. For the case of the directions $\{\mathbf{m}^{(a)}\}$*, if we compare this to (2.20), we see that it is essentially the same optimisation problem as the one we encountered previously. Hence we can deduce that the directions* $\{\mathbf{m}_a\}$ *are just the eigenvectors of the covariance*

matrix of the $\{\mathbf{b}^{(i)}\}$*. And since this covariance matrix is diagonal in the PCA coordinate frame (2.35), we finally have that* $\mathbf{m}^{(a)} = \mathbf{n}^{(a)} \ \forall \ a = 1, \ldots n_m$*.*

For the parameters $\{\sigma_a\}$*, we then have to optimise:*

$$\mathcal{L} = -n_S \sum_{c=1}^{n_m} \ln \sigma_c - \frac{1}{2} \sum_{a=1}^{n_m} \frac{\lambda_a}{\sigma_a^2} + \textit{(constant terms)}, \tag{2.45}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \sigma_a} = -\frac{n_S}{\sigma_a} + \frac{\lambda_a}{\sigma_a^3}, \tag{2.46}$$

$$\therefore \ \frac{\partial \mathcal{L}}{\partial \sigma_a} = 0 \implies \boxed{\sigma_a^2 = \frac{1}{n_S}\lambda_a = \frac{1}{n_S} \sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right)^2}, \tag{2.47}$$

which is just the mean variance across the set of shapes in the direction $\mathbf{n}^{(a)}$*.* □

In many cases, where the shape variation is linear, a multivariate Gaussian density model is sufficient. A single Gaussian cannot however adequately represent cases where there is significant non-linear shape variation, such as that generated when parts of an object rotate, or where there are changes to the viewing angle in a two-dimensional representation of a three-dimensional object. The case of rotating parts of an object can be dealt with by using polar coordinates for these parts, rather than the Cartesian coordinates considered previously [87]. However, such techniques do not deal with the case where the probability distribution is actually multimodal, and in these cases, more general probability distribution modelling techniques must be used. In what follows, we consider kernel-based techniques, the first being classical kernel density estimation, and the second based on the technique of kernel principal component analysis.

2.2.2 Kernel Density Estimation

As before, we start from the set of n_S centred points $\{\mathbf{b}^{(i)}\}$ in shape space $\mathbb{R}^{n_m}$. Kernel density estimation [165] estimates a pdf from data points by essentially smearing out the effect of each data point, by means of a kernel K:

$$p(\mathbf{b}) = \frac{1}{n_S h^{n_m}} \sum_{i=1}^{n_S} K\left(\frac{\mathbf{b} - \mathbf{b}^{(i)}}{h} \right), \tag{2.48}$$

where h is a scaling parameter. In the trivial case where the kernel K is a Dirac δ-function, we obtain the empirical distribution of the data, a pdf $p(\mathbf{b})$ which is zero everywhere except at a data point. A non-trivial choice of kernel would be a multivariate Gaussian:

$$K(\mathbf{b}) \doteq \mathcal{N}(\mathbf{b}; \mathbf{0}, \mathbf{D}), \tag{2.49}$$

where the covariance $\mathbf{D}$ of the kernel can be chosen to match the covariance of the data $\{\mathbf{b}^{(i)}\}$.

A slightly more sophisticated approach is the sample smoothing estimator [15, 175]. Rather than a single global scale parameter h, there is now a local scale parameter, which reflects the local density about each data point, allowing wider kernels in areas where data points are sparse, and narrower kernels in more densely populated areas. Similarly, the kernel covariance can also vary locally [152].

Such kernel methods can give good estimates of the shape distribution. However, the large number of kernels can make them too computationally expensive in an application such as the Active Shape Model (ASM) (Sect. 2.4.1). Cootes et al. [35, 36] developed a method of approximating the full kernel density estimate using a smaller number of Gaussians within a Gaussian mixture model:

$$p_{mix}(\mathbf{b}) \doteq \sum_{i=1}^{n_{mix}} w_i \mathcal{N}(\mathbf{b}; \boldsymbol{\mu}_i, \mathbf{D}_i), \tag{2.50}$$

where n_{mix} is the number of Gaussians within the mixture model, w_i is the weight of the i^{th} Gaussian, with center $\boldsymbol{\mu}_i$ and covariance $\mathbf{D}_i$. The fitting of the parameters can be achieved using a modification [36] to the standard Expectation Maximisation (EM) algorithm method [117].

2.2.3 Kernel Principal Component Analysis

The previous method aims to fit a non-linear or multimodal shape distribution by constructing a parametric non-linear and multimodal distribution within the original shape space.

The Kernel Principal Component Analysis (KPCA) method takes a different approach. KPCA [156, 157] is a technique for non-linear feature extraction, closely related to methods applied in Support Vector Machines [194, 188] (SVMs). Rather than working within the original data space with non-linear and multimodal distributions, KPCA seeks to construct a non-linear mapping of input space $\mathcal{I}$ to a new feature space.

Let $\mathbf{b}$ represent a point in our input data space[3] $\mathcal{I} = \mathbb{R}^{n_m}$, which is mapped to a feature space $\mathcal{F}$:

$$\boldsymbol{\Phi} : \mathbb{R}^{n_m} \mapsto \mathcal{F}, \ \mathbb{R}^{n_m} \ni \mathbf{b} \mapsto \boldsymbol{\Phi}(\mathbf{b}) \in \mathcal{F}, \tag{2.51}$$

[3] Here, we start from the dimensionally reduced space $\mathbb{R}^{n_m}$ rather than the original shape space $\mathbb{R}^{dn_P}$ in order to also include the infinite-dimensional case $n_P \mapsto \infty$ that is considered in Sect. 2.3.

where $\mathcal{F}$ is typically of very high (even infinite) dimensionality. Rather than constructing the mapping $\mathbf{\Phi}$ explicitly, we instead employ the kernel trick, that dot products of mapped points, and hence the implicit mapping $\mathbf{\Phi}$, can be specified by giving the Mercer kernel function [122] $\mathcal{K}$, where:

$$\mathcal{K} : \mathcal{I} \times \mathcal{I} \ \mapsto \ \mathbb{R}, \tag{2.52}$$

$$\mathbf{\Phi}(\mathbf{b}) \cdot \mathbf{\Phi}(\mathbf{c}) \doteq \mathcal{K}(\mathbf{b}, \mathbf{c}) \equiv \mathcal{K}(\mathbf{c}, \mathbf{b}) \ \forall \ \mathbf{b}, \mathbf{c} \in \mathcal{I}. \tag{2.53}$$

If we recall the definition of PCA given earlier (Theorem 2.1), we see that computation of the PCA axes depends on our being able to compute dot products between data points and the PCA axis vectors. We hence deduce that since we can compute dot products in the feature space (2.53) by use of the kernel trick, we can then perform PCA in the feature space $\mathcal{F}$ without having to explicitly construct the kernel mapping $\mathbf{\Phi}$. And since the kernel mapping is a non-linear mapping, PCA in the feature space $\mathcal{F}$ then corresponds to a method of non-linear components analysis in the original data space $\mathcal{I}$.

Suppose we have n_S data points $\{\mathbf{b}^{(i)}\}$ in our data space $\mathcal{I}$. The non-linear KPCA components are then given by the following theorem.

Theorem 2.3. KPCA: Centred Components.
Suppose we have data points $\{\mathbf{b}^{(i)} : i = 1, \ldots n_S\}$ in a data space $\mathcal{I}$, and that there exists a mapping $\mathbf{\Phi}$ to a feature space $\mathcal{F}$, the mapping being defined by a Mercer kernel $\mathcal{K}$ as follows:

$$\mathbf{\Phi} : \mathbf{b} \ \mapsto \ \mathbf{\Phi}(\mathbf{b}) \ \forall \ \mathbf{b} \in \mathcal{I}, \ \ \mathbf{\Phi}(\mathbf{b}) \cdot \mathbf{\Phi}(\mathbf{c}) \doteq \mathcal{K}(\mathbf{b}, \mathbf{c}) \ \forall \ \mathbf{b}, \mathbf{c} \in \mathcal{I}. \tag{2.54}$$

We define the following:

$$\mathbf{\Phi}^{(i)} \doteq \mathbf{\Phi}(\mathbf{b}^{(i)}), \ \ i = 1, \ldots n_S \tag{2.55}$$

$$\widetilde{\mathbf{\Phi}}^{(i)} \doteq \mathbf{\Phi}^{(i)} - \frac{1}{n_S} \sum_{j=1}^{n_S} \mathbf{\Phi}^{(j)}. \tag{2.56}$$

$$\mathbf{K}_{ij} \doteq \mathbf{\Phi}^{(i)} \cdot \mathbf{\Phi}^{(j)} \equiv \mathcal{K}(\mathbf{b}^{(i)}, \mathbf{b}^{(j)}), \tag{2.57}$$

$$\widetilde{\mathbf{K}}_{ij} \doteq \widetilde{\mathbf{\Phi}}^{(i)} \cdot \widetilde{\mathbf{\Phi}}^{(j)}. \tag{2.58}$$

The centred KPCA components $\{\widetilde{\Phi}_\alpha^{(i)} : \alpha = 1, \ldots n_K, n_K \leq n_S - 1\}$ of a data point $\mathbf{b}^{(i)}$ are then extracted from the set of solutions of the eigenproblem:

$$\boxed{\lambda_\alpha n_i^{(\alpha)} = \widetilde{\mathbf{K}}_{ij} n_j^{(\alpha)}, \ \ \widetilde{\Phi}_\alpha^{(i)} = n_i^{(\alpha)}.} \tag{2.59}$$

Proof. As stated above, KPCA applied to the data points $\{\mathbf{b}^{(i)}\}$ is just ordinary linear PCA applied to the mapped data points $\{\mathbf{\Phi}^{(i)}\}$. Centering the mapped data points then gives the $\{\widetilde{\mathbf{\Phi}}^{(i)}\}$ as defined above.

In Theorem 2.1, linear PCA was defined by maximising the quantity given in (2.15). For KPCA, we define a set of orthogonal (but not necessarily orthonormal) direction vectors $\{\mathbf{n}^{(\alpha)} : \alpha = 1, \dots n_K\}$ *which lie in the subspace of* $\mathcal{F}$ *spanned by the* n_S *vectors* $\{\widetilde{\mathbf{\Phi}}^{(i)}\}$*, and maximise the analogous quantity:*

$$\mathcal{L} = \frac{1}{2n_S} \sum_{i=1}^{n_S} \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} \right)^2 - \frac{c_\alpha}{2} \left(\mathbf{n}^{(\alpha)} \cdot \mathbf{n}^{(\alpha)} - (a^{(\alpha)})^2 \right), \tag{2.60}$$

where $c_\alpha > 0$ *is a Lagrange multiplier for maximisation under the normalization constraint:*

$$\mathbf{n}^{(\alpha)} \cdot \mathbf{n}^{(\alpha)} \equiv \|\mathbf{n}^{(\alpha)}\|^2 = (a^{(\alpha)})^2. \tag{2.61}$$

We solve this problem by setting the first derivative of $\mathcal{L}$ *with respect to* $\mathbf{n}^{(\alpha)}$ *to zero as follows:*

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}^{(\alpha)}} = \frac{1}{n_S} \sum_{i=1}^{n_S} \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} \right) \widetilde{\mathbf{\Phi}}^{(i)} - c_\alpha \mathbf{n}^{(\alpha)}. \tag{2.62}$$

$$\therefore \quad \frac{\partial \mathcal{L}}{\partial \mathbf{n}^{(\alpha)}} = 0 \implies \frac{1}{n_S} \sum_{i=1}^{n_S} \widetilde{\mathbf{\Phi}}^{(i)} \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} \right) = c_\alpha \mathbf{n}^{(\alpha)}. \tag{2.63}$$

Taking the dot product with $\widetilde{\mathbf{\Phi}}^{(j)}$*:*

$$\implies \frac{1}{n_S} \sum_{i=1}^{n_S} \left(\widetilde{\mathbf{\Phi}}^{(j)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} \right) \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} \right) = c_\alpha \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(j)} \right), \tag{2.64}$$

$$\implies \sum_{i=1}^{n_S} \widetilde{\mathbf{K}}_{ji} \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} \right) = (n_S c_\alpha) \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(j)} \right). \tag{2.65}$$

The interpretation of $\left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(j)} \right)$ *is that it is the PCA component of* $\widetilde{\mathbf{\Phi}}^{(j)}$ *along the direction* $\mathbf{n}^{(\alpha)}$*, hence the* α^{th} *centred KPCA component of* $\mathbf{b}^{(j)}$*. If we define:*

$$n_j^{(\alpha)} \doteq \left(\mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(j)} \right), \tag{2.66}$$

then PCA in feature space reduces to the eigenproblem:

$$\boxed{\widetilde{\mathbf{K}}_{ji} n_i^{(\alpha)} = (n_S c_\alpha) n_j^{(\alpha)} = \lambda_\alpha n_j^{(\alpha)}.} \tag{2.67}$$

□

Note that as in linear PCA, we choose the define the index α so that the eigenvalues are ordered in decreasing order.

If we recall the definitions of the kernel matrices $\mathbf{K}$ (2.57) and $\widetilde{\mathbf{K}}$ (2.58), and rewrite $\widetilde{\mathbf{K}}$ in terms of $\mathbf{K}$, we have that:

$$\mathbf{K}_{ij} \doteq \mathcal{K}(\mathbf{b}^{(i)}, \mathbf{b}^{(j)}),$$
$$\widetilde{\mathbf{K}}_{ij} = \mathbf{K}_{ij} - \frac{1}{n_S}\sum_p \mathbf{K}_{pj} - \frac{1}{n_S}\sum_q \mathbf{K}_{iq} + \frac{1}{n_S^2}\sum_{p,q} \mathbf{K}_{pq}. \tag{2.68}$$

Looked at in this way, in terms of kernels involving the input data points, the significance of $\widetilde{\mathbf{K}}$ and its eigenvectors is obscured. It is only the identification between Mercer kernels and mappings that enables $\widetilde{\mathbf{K}}$ to be seen as just the covariance matrix for the mapped data points.[4]

We can also define non-centred KPCA components $\{\Phi_\alpha^{(i)}\}$, where:

$$\boxed{\Phi_\alpha^{(i)} \doteq \mathbf{n}^{(\alpha)} \cdot \mathbf{\Phi}^{(i)}.} \tag{2.69}$$

Theorem 2.4. KPCA: Non-centred Components.
With definitions as above, the non-centred KPCA components of a data point $\mathbf{b}^{(j)}$ *are given by:*

$$\boxed{\Phi_\alpha^{(j)} \doteq \mathbf{n}^{(\alpha)} \cdot \mathbf{\Phi}^{(j)} = \frac{1}{\lambda_\alpha} n_i^{(\alpha)} \mathbf{K}_{ij}.} \tag{2.70}$$

Proof. Remember that the $\{\widetilde{\mathbf{\Phi}}^{(i)}\}$ *are centred points, hence:*

$$\sum_{i=1}^{n_s} \widetilde{\mathbf{\Phi}}^{(i)} \equiv \mathbf{0} \;\Rightarrow\; \sum_{i=1}^{n_s} \mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(i)} = 0 \;\Rightarrow\; \sum_{i=1}^{n_s} n_i^{(\alpha)} = 0. \tag{2.71}$$

For all $\lambda_\alpha \neq 0$ *(2.59), the corresponding eigenvector* $\mathbf{n}^{(\alpha)}$ *lies in the space spanned by the set* $\{\widetilde{\mathbf{\Phi}}^{(i)}\}$*, hence* $\mathbf{n}^{(\alpha)}$ *can be expanded in this basis:*

$$\mathbf{n}^{(\alpha)} = \sum_{i=1}^{n_S} w_i^{(\alpha)} \widetilde{\mathbf{\Phi}}^{(i)}. \tag{2.72}$$
$$\therefore\; \mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}^{(j)} \doteq n_j^{(\alpha)} = w_i^{(\alpha)} \widetilde{\mathbf{K}}_{ij} \tag{2.73}$$
$$\Rightarrow\; \lambda_\alpha n_j^{(\alpha)} = \lambda_\alpha w_i^{(\alpha)} \widetilde{\mathbf{K}}_{ij}. \tag{2.74}$$

Comparison with the eigenvector equation gives that:

$$w_i^{(\alpha)} = \frac{1}{\lambda_\alpha} n_i^{(\alpha)} \;\Rightarrow\; \mathbf{n}^{(\alpha)} = \frac{1}{\lambda_\alpha} \sum_{i=1}^{n_S} n_i^{(\alpha)} \widetilde{\mathbf{\Phi}}^{(i)}. \tag{2.75}$$

Substituting $\mathbf{\Phi}^{(i)} - \frac{1}{n_S}\sum_{k=1}^{n_S} \mathbf{\Phi}^{(k)}$ *for* $\widetilde{\mathbf{\Phi}}^{(i)}$ *and using (2.71) then gives:*

[4] Note that this result, and the definition of a finite-dimensional covariance matrix $\widetilde{\mathbf{K}}_{ij}$ in a space $\mathcal{F}$ that is possibly infinite-dimensional is actually an application of a result for covariance matrices that is presented later in this chapter (Theorems 2.6 and 2.7).

$$\boxed{\mathbf{n}^{(\alpha)} = \frac{1}{\lambda_\alpha} \sum_{i=1}^{n_S} n_i^{(\alpha)} \widetilde{\mathbf{\Phi}}^{(i)} = \frac{1}{\lambda_\alpha} \sum_{i=1}^{n_S} n_i^{(\alpha)} \mathbf{\Phi}^{(i)}.} \tag{2.76}$$

Hence:

$$\boxed{\Phi_\alpha^{(j)} \doteq \mathbf{n}^{(\alpha)} \cdot \mathbf{\Phi}^{(j)} = \frac{1}{\lambda_\alpha} n_i^{(\alpha)} \mathbf{K}_{ij}.} \tag{2.77}$$

□

As well as the centred and non-centred components for data points, we can now also compute the centred and non-centred KPCA components for an arbitrary point $\mathbf{b}$ in the input space.

Theorem 2.5. KPCA: Components of a Test Point.
A general point $\mathbf{b}$ *in the input space maps to a point* $\mathbf{\Phi}(\mathbf{b})$ *in feature space, with non-centred KPCA components:*

$$\boxed{\Phi_\alpha(\mathbf{b}) \doteq \mathbf{n}^{(\alpha)} \cdot \mathbf{\Phi}(\mathbf{b}) = \frac{1}{\lambda_\alpha} \sum_{i=1}^{n_S} n_i^{(\alpha)} \mathcal{K}(\mathbf{b}^{(i)}, \mathbf{b}),} \tag{2.78}$$

and centred KPCA components:

$$\widetilde{\mathbf{\Phi}}(\mathbf{b}) \doteq \mathbf{\Phi}(\mathbf{b}) - \frac{1}{n_S} \sum_{j=1}^{n_S} \mathbf{\Phi}^{(j)},$$

$$\boxed{\widetilde{\Phi}_\alpha(\mathbf{b}) \doteq \mathbf{n}^{(\alpha)} \cdot \widetilde{\mathbf{\Phi}}(\mathbf{b}) = \Phi_\alpha(\mathbf{b}) - \frac{1}{\lambda_\alpha n_S} \sum_{i,j=1}^{n_S} n_i^{(\alpha)} \mathbf{K}_{ij}.} \tag{2.79}$$

Proof. This follows straightforwardly from the previous results, and is left as an exercise for the reader.

2.2.4 Using Principal Components to Constrain Shape

We now need to consider the ways that PCA and KPCA components are used in shape applications, and consider in detail the significant differences between them.

For shape spaces $\mathbb{R}^{n_m}$, it is obvious that PCA components can increase without limit, since the mapping from shape parameters to shapes (2.38) is

defined for any point in the parameter space. Hence we can always exclude test shapes $\mathbf{b}$ far from the training data by setting an *upper* bound on each of the PCA components, creating a bounding parallelepiped. This does not necessarily exclude all shapes which are unlike the training data for cases where the training set shape variation is non-linear or multimodal.

For the case of Gaussian distributions of shapes (2.39), when the PCA axes are appropriately scaled, the pdf becomes spherically symmetric:

$$p(\mathbf{b}) \propto \exp\left(-\frac{1}{2}\sum_{a=1}^{n_m}\left(\frac{\mathbf{b}\cdot\mathbf{m}^{(a)}}{\sigma_a}\right)^2\right),\ \ \mathbf{m}^{(a)}\cdot\mathbf{m}^{(b)} = \delta_{ab}, \tag{2.80}$$

$$b_a \doteq \mathbf{m}^{(a)}\cdot\mathbf{b},\ \ a = 1,\dots n_m. \tag{2.81}$$

$$\tilde{b}_a \doteq \frac{b_a}{\sigma_a}, \tag{2.82}$$

$$\therefore\ \ p(\tilde{\mathbf{b}}) \propto \exp\left(-\frac{1}{2}\sum_{a=1}^{n_m}\left(\tilde{\mathbf{b}}\cdot\mathbf{m}^{(a)}\right)^2\right) = \exp\left(-\frac{1}{2}\|\tilde{\mathbf{b}}\|^2\right), \tag{2.83}$$

where:

$$\|\tilde{\mathbf{b}}\|^2 \doteq \sum_{a=1}^{n_m}\left(\frac{b_a}{\sigma_a}\right)^2, \tag{2.84}$$

is the squared Mahalanobis distance [110] between the point $\mathbf{b}$ and the origin (mean shape). Hence surfaces of constant Mahalanobis distance correspond to probability isosurfaces, with a simple monotonic relationship between Mahalanobis distance and probability density, and we can create bounding ellipsoids by placing an upper bound on the Mahalanobis distance.

Our initial trivially obvious observation that PCA components can increase without limit is not however generally true for KPCA components.

Consider Theorem 2.5, (2.78):

$$\Phi_\alpha(\mathbf{b}) = \frac{1}{\lambda_\alpha}\sum_{i=1}^{n_S} n_i^{(\alpha)}\mathcal{K}(\mathbf{b}^{(i)},\mathbf{b}). \tag{2.85}$$

We see that the way the non-centred or centred components of a test point behave as the test point moves away from the data depends on the way the kernel function $\mathcal{K}(\mathbf{b}^{(i)},\mathbf{b})$ behaves. As was noted by Schölkopf et al. [156]:

$$\Phi_\alpha(\mathbf{b}) \doteq \mathbf{n}^{(\alpha)}\cdot\mathbf{\Phi}(\mathbf{b}) \le \|\mathbf{n}^{(\alpha)}\|\ \|\mathbf{\Phi}(\mathbf{b})\| = a^{(\alpha)}\left(\mathcal{K}(\mathbf{\Phi}(\mathbf{b}),\mathbf{\Phi}(\mathbf{b}))\right)^{\frac{1}{2}}, \tag{2.86}$$

where the normalization of the vector $\mathbf{n}^{(\alpha)}$ is as defined in (2.61).

In Table 2.1, we give examples of some commonly used kernels, and the range of allowed values of $\|\mathbf{\Phi}(\mathbf{b})\|$.

We see that for the polynomial kernels (which of course contain linear PCA as a limiting case when $m = 1$), the values of the KPCA components

Table 2.1 Examples of Mercer kernels.

Kernel Type	$\mathcal{K}(\mathbf{b},\mathbf{c})$	$\|\mathbf{\Phi}(\mathbf{b})\|$
Polynomial	$(\mathbf{b}\cdot\mathbf{c})^m$,	$0 \leq \|\mathbf{\Phi}(\mathbf{b})\|^2 < \infty.$
	$(\mathbf{b}\cdot\mathbf{c}+r)^m$	$r^m \leq \|\mathbf{\Phi}(\mathbf{b})\|^2 < \infty.$
Radial Basis Function (RBF)	$\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{b}-\mathbf{c}\|^2\right)$	$\|\mathbf{\Phi}(\mathbf{b})\|^2 \equiv 1.$
Sigmoid	$\tanh(\mathbf{b}\cdot\mathbf{c}+r)$	$0 \leq \|\mathbf{\Phi}(\mathbf{b})\|^2 \leq 1.$

are unlimited. However, for both the sigmoid and RBF kernels, $\|\mathbf{\Phi}(\mathbf{b})\|$ and hence the values of the components are strictly bounded.

The RBF kernel is particularly interesting. Note that the modulus of the mapped vector $\mathbf{\Phi}(\mathbf{b})$ in feature space is *identically* one. This means that the mapped input space $\Phi(\mathcal{I})$ is an embedded submanifold of feature space $\mathcal{F}$. If we consider the modulus of the difference vector between two mapped points, we have:

$$\|\mathbf{\Phi}(\mathbf{b}) - \mathbf{\Phi}(\mathbf{c})\|^2 \equiv 2\left(1 - \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{b}-\mathbf{c}\|^2\right)\right). \tag{2.87}$$

We hence see that as $\mathbf{b}$ moves away from $\mathbf{c}$ in input space:

$$\|\mathbf{\Phi}(\mathbf{b}) - \mathbf{\Phi}(\mathbf{c})\|^2 \to 2 \text{ as } \mathbf{\Phi}(\mathbf{b})\cdot\mathbf{\Phi}(\mathbf{c}) \to 0,$$

which is what we would expect for orthogonal points on a unit hypersphere.

Consider now the projection from this embedded submanifold to KPCA space, which is the space of KPCA components.[5] The explicit expression for the non-centred KPCA components (2.78) for the case of an RBF kernel is:

$$\Phi_\alpha(\mathbf{b}) = \frac{1}{\lambda_\alpha}\sum_{i=1}^{n_S} n_i^{(\alpha)} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{b}^{(i)}-\mathbf{b}\|^2\right), \quad \sum_{i=1}^{n_s} n_i^{(\alpha)} = 0. \tag{2.88}$$

The first trivially obvious point to note is that all the KPCA components tend to zero for any test point far from all the data. Let us now focus on the case where the kernel width σ is sufficiently small, and for a fixed value of α. Because of the summation constraint on $\{n_i^{(\alpha)}\}$, at least one of the elements $\{n_i^{(\alpha)}\}$ must be of opposite sign to the others. The set of $\{\Phi_\alpha^{(i)} = \Phi_\alpha(\mathbf{b}^{(i)})\}$ (the non-centred KPCA components of the data points), will hence take both negative and positive values across the data. We can hence conclude that for sufficiently small values of σ, the extrema of any KPCA component

[5] It should be noted that whilst we can always move from input space (the point $\mathbf{b}$) to KPCA space (the space of KPCA components) using (2.78), we cannot necessarily do the inverse. So, it is not necessarily the case that an arbitrary point in KPCA space (defined by a set of KPCA components) possesses a pre-image in input space (although various approximation methods do exist, see [156] Sect. 4.5 for further details).

will tend to lie in the vicinity of the data points, taking both negative and positive values, and that these values *bracket* the values obtained for points far from the data (that is, zero). Since this is an ordering property, it persists if we switch to centred KPCA components, since this just corresponds to a translation in KPCA space.

Consider a path in input space which starts at infinity, then approaches some part of the data, then moves out to infinity again. At first, *all* KPCA components will have vanishingly small modulus. As the point approaches the data, some component(s) acquire a value of larger modulus, which then shrinks again as we move away from all the data. In linear PCA, proximity to data was described by placing an *upper* bound to the modulus of each component, but such a procedure will not be generally valid for KPCA components.[6] This argument rests on the kernel width σ being in some sense small, but this behaviour persists for some finite range of σ (indeed, if it did not, RBF KPCA would be of no use as a *non-linear* feature extractor).

It hence suggests that an appropriate proximity-to-data measure for KPCA components would involve a sum over the moduli of non-centred KPCA components. This was the approach taken by Twining and Taylor [185] as follows.

We first define the normalization of the eigenvectors. In contrast to Mika et al. [124] and Schölkopf et al. [156], the eigenvectors are normalized (2.61) with respect to the data:

$$\sum_{i=1}^{n_S} \left(\widetilde{\Phi}_i^{(\alpha)}\right)^2 \equiv \sum_{i=1}^{n_S} \left(n_i^{(\alpha)}\right)^2 \doteq 1 \;\Rightarrow\; \|\mathbf{n}^{(\alpha)}\|^2 = \left(a^{(\alpha)}\right)^2 = \frac{1}{\lambda_\alpha} \;\forall\; \alpha, \quad (2.89)$$

which hence gives an orthogonal but not orthonormal basis $\{\mathbf{n}^{(\alpha)}\}$. We then introduce scaled non-centred components:

$$\Psi_\alpha(\mathbf{b}) \doteq \lambda_\alpha \Phi_\alpha(\mathbf{b}) = \sum_{i=1}^{n_S} n_i^{(\alpha)} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{b}^{(i)} - \mathbf{b}\|^2\right). \quad (2.90)$$

As to why scaled components are used rather than the original components, consider the following. For a general test point, we have the usual definition of non-centred components (2.78):

$$\Phi_\alpha \doteq \mathbf{n}^{(\alpha)} \cdot \mathbf{\Phi}(\mathbf{b}). \quad (2.91)$$

We can hence expand the vector $\mathbf{\Phi}(\mathbf{b})$ in terms of the eigenvectors thus:

[6] This point was not sufficiently appreciated by Romdhani et al. [148] when they considered shape spaces for faces. They defined their valid shape region by placing an *upper bound* on their KPCA components, by analogy with the case of linear PCA. In the limit of large σ, RBF KPCA does indeed approach linear PCA (see [185] Appendix A for details), but this behaviour and hence the analogy is not generally valid for RBF KPCA (nor for KPCA in general, as noted in the text).

$$\mathbf{\Phi}(\mathbf{b}) = \sum_{\alpha=1}^{n_K} \lambda_\alpha \Phi_\alpha(\mathbf{b}) \mathbf{n}^{(\alpha)} + \mathbf{V}(\mathbf{b}), \tag{2.92}$$

where n_K denotes that we have chosen only the n_K largest eigenvalues to include, and $\mathbf{V}(\mathbf{b})$ is some vector perpendicular to the set of eigenvectors so chosen. We can then deduce that since $\|\mathbf{\Phi}(\mathbf{b})\| \equiv 1$ for all test points, we have that:

$$\|\mathbf{\Phi}(\mathbf{b})\|^2 = \sum_{\alpha=1}^{n_K} \lambda_\alpha |\Phi_\alpha(\mathbf{b})|^2 + \|\mathbf{V}(\mathbf{b})\|^2 \equiv 1 \;\Rightarrow\; \sum_{\alpha=1}^{n_K} \lambda_\alpha |\Phi_\alpha(\mathbf{b})|^2 \leq 1. \tag{2.93}$$

This hence places a strict upper bound on the modulus of the components, with the larger the eigenvalue the *smaller* the maximum allowed value of $|\Phi_\alpha(\mathbf{b})|$. For the purposes of proximity-to-data estimation using sums of squares of components, this is not an ideal situation, since we feel intuitively that the modes with larger eigenvalues should make the largest contribution. Hence the use of the scaled components $\Psi_\alpha(\mathbf{b}) \doteq \lambda_\alpha \Phi_\alpha(\mathbf{b})$, since $|\Psi_\alpha(\mathbf{b})|^2 \leq \lambda_\alpha$.

A general proximity-to-data measure is then of the form:

$$f^{(m)}(\mathbf{b}) \propto \left(\sum_{\alpha=1}^{n_K} \Psi_\alpha(\mathbf{b}) \Psi_\alpha(\mathbf{b}) \right)^{\frac{m}{2}}, \quad m = 1, 2, \ldots \infty. \tag{2.94}$$

Of particular interest is the simplest case where $m = 2$, where the proximity-to-data measure can be exactly normalized, giving the pseudo-density:

$$\boxed{\widehat{p}(\mathbf{b}) = \frac{1}{A} \sum_{i,j=1}^{n_S} \sum_{\alpha=1}^{n_K} n_i^{(\alpha)} n_j^{(\alpha)} \mathcal{K}(\mathbf{b}, \mathbf{b}^{(i)}) \mathcal{K}(\mathbf{b}, \mathbf{b}^{(j)})} \tag{2.95}$$

$$\text{Normalization factor: } A = \sigma^{n_m} \pi^{\frac{n_m}{2}} \operatorname{Tr}(\mathbf{K}^{\frac{1}{2}} \mathbf{B})$$

$$\text{where } \mathbf{K}_{ij}^{\frac{1}{2}} \doteq \sqrt{\mathbf{K}_{ij}} \text{ and } \mathbf{B}_{ij} \doteq \sum_{\alpha=1}^{n_K} n_i^{(\alpha)} n_j^{(\alpha)}.$$

It has been shown, with the aid of exactly-solvable data distributions ([185] Appendix B), that this does indeed provide a smoothed estimate of the data density, with the kernel width σ acting as a smoothing parameter. On artificial noisy data, it was also shown to give quantitatively better estimates of the density than either naïve or adaptive kernel density methods such as those described in Sect. 2.2.2.

In [183], Twining and Taylor used a lower bound on the pseudo-density to define the class of allowed shapes for statistical shape models, and showed that this gave improved performance when compared to standard linear models.

From a theoretical point of view, the above density estimate is interesting since it differs from those described earlier (Sect. 2.2.2) in that it is a quadratic rather than a linear combination of kernel functions. The kernel width plays the same sort of rôle in both types of kernel methods. The KPCA pseudo-density possesses a further advantage, in that the number of KPCA components n_K included in the summation can be varied. This means that for noisy data, the higher modes (which tend to contain the noise) can be neglected in the density estimate by truncating the summation. As noted above, this has been shown to produce superior results on noisy input data to the standard kernel methods which do not possess such a feature.

However, the basic algorithm as summarized here is still computationally intensive, in that we have to calculate the kernel function between a test point and all data points. We note that several authors (e.g., [128, 166]) have already addressed the problem of optimising KPCA as applied to large data sets ($n_S > 3000$). The question of constructing reduced-set approximations to the exact KPCA results [155] has also been addressed, so we anticipate that considerable improvement on this basic algorithm is possible.

So far, we have considered training sets of shapes represented by a finite number of points, and shown how various methods of principal component analysis and density estimation can be used to describe the distribution of such shapes in shape space. However, in the real world, natural shapes we encounter are continuous. We hence proceed to discuss how the analysis can be extended to the case of continuous shapes.

2.3 Infinite-Dimensional Representations of Shape

In Sect. 2.1, we considered finite-dimensional representations of shape, and saw that both PCA and Gaussian modelling rests on the properties of the covariance matrix $\mathbf{D}$.

However, our input shapes are actually continuous, so we are interested in the limit where $n_P \to \infty$. The problem with our original covariance matrix $\mathbf{D}$ (2.16) is that it is of size $dn_P \times dn_P$, hence becomes infinitely large in this limit. However, the number of non-zero eigenvalues is at most $n_S - 1$. These eigenvalues can be calculated even in the limit by considering the following known result for covariance matrices [51].

Theorem 2.6. Equivalence of Eigenvectors: Finite-Dimensional Case.
Consider a set of shapes $\{\mathbf{x}_i : i = 1, \dots n_S\}$ *with shape covariance matrix* $\mathbf{D}$ *(2.16) with elements:*

$$\boxed{D_{\mu\nu} \doteq \sum_{i=1}^{n_S} (\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_i - \bar{\mathbf{x}})_\nu,} \tag{2.96}$$

and the corresponding eigenvectors and eigenvalues:

$$\mathbf{D}\mathbf{n}^{(a)} \doteq \lambda_a \mathbf{n}^{(a)}, \;\; \mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} = \delta_{ab}, \tag{2.97}$$

where there are at most $n_S - 1$ *non-zero eigenvalues.*

Also consider the matrix $\widetilde{\mathbf{D}}$ *with components:*

$$\boxed{\widetilde{D}_{ij} \doteq (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot (\mathbf{x}_j - \bar{\mathbf{x}}).} \tag{2.98}$$

There then exists a one-to-one mapping between the eigenvectors of $\mathbf{D}$ *with non-zero eigenvalue, and the eigenvectors of* $\widetilde{\mathbf{D}}$ *with non-zero eigenvalues, and the eigenvalues for corresponding eigenvectors are equal.*

Proof. We start from the eigenvector equation for $\mathbf{D}$*, and the definition of* $\mathbf{D}$ *(2.16)*[7]*:*

$$D_{\mu\nu} n_\nu^{(a)} = \lambda_a n_\mu^{(a)} \;\; \Rightarrow \;\; (\mathbf{x}_j - \bar{\mathbf{x}})_\mu (\mathbf{x}_j - \bar{\mathbf{x}})_\nu n_\nu^{(a)} = \lambda_a n_\mu^{(a)}, \;\; \lambda_a \neq 0. \tag{2.99}$$

Multiplying both sides by $(\mathbf{x}_i - \bar{\mathbf{x}})_\mu$*, and summing over* μ *gives:*

$$(\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_j - \bar{\mathbf{x}})_\mu (\mathbf{x}_j - \bar{\mathbf{x}})_\nu n_\nu^{(a)} = \lambda_a (\mathbf{x}_i - \bar{\mathbf{x}})_\mu n_\mu^{(a)}, \tag{2.100}$$

$$\Rightarrow \;\; (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot (\mathbf{x}_j - \bar{\mathbf{x}}) \left[(\mathbf{x}_j - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right] = \lambda_a \left[(\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right]. \tag{2.101}$$

$$\textit{Define: } \widetilde{\mathbf{n}}^{(a)} \doteq \{ n_i^{(a)} = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} : i = 1, \ldots n_S \}. \tag{2.102}$$

$$\therefore \;\; \widetilde{D}_{ij} \widetilde{n}_j^{(a)} = \lambda_a \widetilde{n}_i^{(a)}, \;\; \widetilde{\mathbf{D}} \widetilde{\mathbf{n}}^{(a)} = \lambda_a \widetilde{\mathbf{n}}^{(a)}. \tag{2.103}$$

So either $\widetilde{\mathbf{n}}^{(a)} = \mathbf{0}$*, or* $\widetilde{\mathbf{n}}^{(a)}$ *is an eigenvector of* $\widetilde{\mathbf{D}}$ *with eigenvalue* λ_a*. If* $\widetilde{\mathbf{n}}^{(a)} = \mathbf{0}$*, it then follows from the definition (2.102) that the corresponding* $\mathbf{n}^{(a)}$ *would be orthogonal to the data, with* $\lambda_a = 0$*, which contradicts our original condition that* $\lambda_a \neq 0$*.*

We can hence conclude that all eigenvectors of $\mathbf{D}$ *with non-zero eigenvalue have a corresponding eigenvector of* $\widetilde{\mathbf{D}}$ *with the same eigenvalue. What remains is the converse claim, that all eigenvectors of* $\widetilde{\mathbf{D}}$ *correspond to an eigenvector of* $\mathbf{D}$*.*

The proof follows in an exactly similar fashion. We start from the eigenvector equation:

$$\widetilde{\mathbf{D}} \widetilde{\mathbf{n}}^{(a)} = \lambda_a \widetilde{\mathbf{n}}^{(a)}, \tag{2.104}$$

where $\widetilde{\mathbf{n}}^{(a)}$ *is any eigenvector of* $\widetilde{\mathbf{D}}$ *with a non-zero eigenvalue,* $\lambda_a \neq 0$*.*

Inserting the definition of $\widetilde{\mathbf{D}}$ *(2.98):*

$$\widetilde{D}_{ij} \widetilde{n}_j^{(a)} = \lambda_a \widetilde{n}_i^{(a)} \;\; \Rightarrow \;\; (\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_j - \bar{\mathbf{x}})_\mu \widetilde{n}_j^{(a)} = \lambda_a \widetilde{n}_i^{(a)}. \tag{2.105}$$

[7] Note our summation convention (see Glossary) that repeated indices are summed over, except where indices appear in brackets $.^{(a)}$, which are only summed over if explicitly stated.

Multiplying by $(\mathbf{x}_i - \bar{\mathbf{x}})_\nu$ *and summing over* i*:*

$$(\mathbf{x}_i - \bar{\mathbf{x}})_\nu (\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_j - \bar{\mathbf{x}})_\mu \widetilde{n}_j^{(a)} = \lambda_a \widetilde{n}_i^{(a)} (\mathbf{x}_i - \bar{\mathbf{x}})_\nu \tag{2.106}$$

$$\Rightarrow \; (\mathbf{x}_i - \bar{\mathbf{x}})_\nu (\mathbf{x}_i - \bar{\mathbf{x}})_\mu \left[(\mathbf{x}_j - \bar{\mathbf{x}})_\mu \widetilde{n}_j^{(a)} \right] = \lambda_a \left[\widetilde{n}_i^{(a)} (\mathbf{x}_i - \bar{\mathbf{x}})_\nu \right]. \tag{2.107}$$

$$\textit{Define:} \quad \mathbf{n}^{(a)} = \{ n_\mu^{(a)} = \widetilde{n}_i^{(a)} (\mathbf{x}_i - \bar{\mathbf{x}})_\mu \}, \tag{2.108}$$

$$\therefore \quad D_{\nu\mu} n_\mu^{(a)} = \lambda_a n_\nu^{(a)}, \;\; \mathbf{D}\mathbf{n}^{(a)} = \lambda_a \mathbf{n}^{(a)}. \tag{2.109}$$

The solution $\mathbf{n}^{(a)} = 0$ *can be excluded: if we consider (2.105), this can be rewritten as:*

$$(\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_j - \bar{\mathbf{x}})_\mu \widetilde{n}_j^{(a)} = \lambda_a \widetilde{n}_i^{(a)} \;\; \Rightarrow \;\; (\mathbf{x}_i - \bar{\mathbf{x}})_\mu n_\mu^{(a)} = \lambda_a \widetilde{n}_i^{(a)}. \tag{2.110}$$

So if $\widetilde{\mathbf{n}}^{(a)} \neq \mathbf{0}$ *and* $\lambda_a \neq 0$*, it then follows that* $\mathbf{n}^{(a)} \neq 0$*.*

We hence see that all eigenvectors of $\widetilde{\mathbf{D}}$ *with non-zero eigenvalue correspond to eigenvectors of* $\mathbf{D}$ *with the same eigenvalue. And since the correspondence has now been established in both directions, it then follows that all the eigenvectors of* $\widetilde{\mathbf{D}}$ *and* $\mathbf{D}$ *with non-zero eigenvalues can be placed into a one-to-one correspondence, and the eigenvalues are the same.* □

Earlier (2.34), we defined shape parameter vectors $\mathbf{b}$, where:

$$\mathbf{b}^{(i)} = \{ b_a^{(i)} : a = 1, \ldots n_m \}, \;\; b_a^{(i)} \doteq \left(\mathbf{n}^{(a)} \cdot (\mathbf{x}_i - \bar{\mathbf{x}}) \right). \tag{2.111}$$

If we consider the eigenvectors $\{\widetilde{\mathbf{n}}^{(a)}\}$ defined above (2.102):

$$\widetilde{n}_i^{(a)} = \mathbf{n}^{(a)} \cdot (\mathbf{x}_i - \bar{\mathbf{x}}), \tag{2.112}$$

we see that:

$$\boxed{b_a^{(i)} \equiv \widetilde{n}_i^{(a)}.} \tag{2.113}$$

That is, it seems that we can obtain the parameter vectors for the training set of shapes directly from the eigenvectors of $\widetilde{\mathbf{D}}$. There is a final point that needs to be noted. The eigenvectors $\{\widetilde{\mathbf{n}}^{(a)}\}$ defined in (2.102) inherited their normalization from the *orthonormal* set of eigenvectors $\{\mathbf{n}^{(a)}\}$. It hence follows that:

$$\boxed{\|\widetilde{\mathbf{n}}^{(a)}\|^2 = \lambda_a.} \tag{2.114}$$

This means that we can indeed extract the needed eigenvalues and the shape parameter vectors directly from $\widetilde{\mathbf{D}}$.

As regards the shape generated from an arbitrary parameter vector $\mathbf{b}$, we have:

$$\mathbf{x} \doteq \bar{\mathbf{x}} + \mathbf{N}\mathbf{b}, \;\; \mathbf{N} = \{ N_{\mu a} = n_\mu^{(a)} \}. \tag{2.115}$$

This can then be re-written as:

$$\boxed{\mathbf{x} \doteq \bar{\mathbf{x}} + \sum_{i=1}^{n_S} (\mathbf{b}^{(i)} \cdot \mathbf{b})(\mathbf{x}_i - \bar{\mathbf{x}}).} \tag{2.116}$$

This means that a generated shape is to be considered as a linear combination of the training shapes, where the weight for each shape is related to the parameter vectors for the training shape and the generated shape.

2.3.1 Parameterised Representations of Shape

Let us consider for the moment the simple case of shapes in $\mathbb{R}^2$. The finite set of n_P points that describe such a shape can be indexed by an integer $\{\mathbf{x}^{(j)} : j = 1, \ldots n_P,\ \mathbf{x}^{(j)} \in S.\}$. For (single-part) shapes with the simplest topology (open or closed lines), we can include the connectivity information in with the indexing, so that $\mathbf{x}^{(j-1)}$ is joined to $\mathbf{x}^{(j)}$ and so on. The same indexing is applied across a set of shapes, so that the j^{th} point on any one shape corresponds with the j^{th} point on any other shape.

The subset of the integers $\{j : j = 1, \ldots n_P\}$ can then be considered as the parameter space for our shape representation, a parameter space consisting of just a set of discrete points. To construct an *infinite-dimensional* shape representation, we just need to consider a continuous parameter space.[8]

For our simple one-dimensional shapes, we have a continuous shape S_i which is then sampled at n_P points. S_i is then represented by a finite-dimensional shape vector:

$$\mathbf{x}_i \doteq \{\mathbf{x}_i^{(j)} : j = 1, \ldots n_P\}.$$

The associated discrete parameter space is:

$$\{1, 2, 3, \ldots n_P\} \equiv \left\{ \frac{1}{n_P}, \frac{2}{n_P}, \ldots, 1 \right\},$$

with the associated mapping:

$$\{1, 2, 3, \ldots n_P\} \ni j \mapsto \mathbf{x}_i^{(j)} \in S_i.$$

The mapping respects the topology of the shape, so that the ordering of the integers respects the ordering of points along the shape.

[8] Note that the reader should not confuse this usage of *parameter space* in terms of parameterised shape with the space of *shape* parameters $\mathbf{b}$ (2.34). *Parameter space*, when used in this latter context, should be understood as a convenient shorthand for *shape*-parameter space.

The continuous analog of this parameter space is just the real line between 0 and 1. For an arbitrary point on this line, with parameter value $x \in [0, 1]$, we then have the mapping to the shape:

$$[0, 1] \ni x \mapsto \mathbf{S}_i(x) \in S_i.$$

$\mathbf{S}_i(\cdot)$ is then the vector-valued *shape function* associated with shape S_i. The mapping between the real line and the shape has to respect the topology of the shape. This means that if we traverse the shape in one direction, the parameter value then either strictly decreases or strictly increases. The only exception is if the shape has the topology of a circle. This means that the point $x = 0$ in parameter space is connected to $x = 1$, so that the ends of the real line between 0 and 1 have now been joined to form a circle. In general, the mapping from the parameter space to the shape has to be continuous, and one-to-one, so that each parameter value corresponds to a single point on the shape, and each point on the shape has a single associated parameter value. In mathematical terms, such a mapping is a homeomorphism.[9]

In the general case, we have a vector-valued parameter $\mathbf{x} \in X$, where the topology of the parameter space X matches the topology of the shapes. So, for example, simple shapes in $\mathbb{R}^3$ might have the topology of a sphere, where the parameter space is then the appropriate topological primitive – that is, a sphere. For each shape in the training set, there is a continuous, one-to-one mapping $\boldsymbol{\mathfrak{X}}_i$ from the parameter space to the shape S_i.

$$\boxed{X \overset{\boldsymbol{\mathfrak{X}}_i}{\longmapsto} S_i, \ \mathbf{x} \overset{\boldsymbol{\mathfrak{X}}_i}{\longmapsto} \mathbf{S}_i(\mathbf{x}).} \tag{2.117}$$

The continuity of the mapping means that as the point $\mathbf{x}$ moves on some continuous path around the parameter space, the point $\mathbf{S}_i(\mathbf{x})$ similarly moves in a continuous path on the surface of the shape S_i, with no sudden jumps allowed.

In the finite-dimensional shape representation, shape correspondence is between points with the same index, so that $\mathbf{x}_i^{(j)} \sim \mathbf{x}_k^{(j)}, \ \forall \ i, k$, where $\cdot \sim \cdot$ denotes correspondence. In the continuous case, correspondence is defined analogously:

$$\boxed{\mathbf{S}_j(\mathbf{x}) \sim \mathbf{S}_k(\mathbf{x}),} \tag{2.118}$$

which gives a dense correspondence between any pair of shapes. It is the details of the set of mappings $\{\boldsymbol{\mathfrak{X}}_i\}$ which defines the dense correspondence across the set of shapes.

The covariance matrix $\widetilde{\mathbf{D}}$ for finite-dimensional shape representations is given by (2.98):

$$\widetilde{D}_{jk} \doteq (\mathbf{x}_j - \bar{\mathbf{x}}) \cdot (\mathbf{x}_k - \bar{\mathbf{x}}). \tag{2.119}$$

[9] Technically, a mapping which is one-to-one and continuous in both directions. If the mapping and its inverse is also constrained to be *differentiable* to some order, then this mapping is a *diffeomorphism*.

The continuous analog of the mean shape vector $\bar{\mathbf{x}}$ is the mean shape:

$$\bar{S}(\mathbf{x}) \doteq \frac{1}{n_S}\sum_{i=1}^{n_S}\mathbf{S}_i(\mathbf{x}). \tag{2.120}$$

To take the limit $n_P \to \infty$, we imagine sampling the shapes ever more densely. To make sure this limit is well-defined, we take the n_P points to be equally spaced on the mean shape. This then gives the infinite-dimensional limit of the covariance matrix:

$$\widetilde{D}_{jk} \doteq (\mathbf{x}_j - \bar{\mathbf{x}})\cdot(\mathbf{x}_k - \bar{\mathbf{x}}) \;\to\; \int (\mathbf{S}_j(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x}))\cdot(\mathbf{S}_k(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x}))dA(\mathbf{x}), \tag{2.121}$$

where $dA(\mathbf{x})$ is the length/area element on the mean shape at the point $\bar{\mathbf{S}}(\mathbf{x})$.

In order to have a smooth transition as regards eigenvalues, it is convenient to introduce the normalized covariance matrices, so that:

$$D_{\mu\nu} \doteq \frac{1}{n_P}\sum_{i=1}^{n_S}(\mathbf{x}_i - \bar{\mathbf{x}})_\mu(\mathbf{x}_i - \bar{\mathbf{x}})_\nu, \tag{2.122}$$

$$\widetilde{D}_{ij} \doteq \frac{1}{n_P}(\mathbf{x}_i - \bar{\mathbf{x}})\cdot(\mathbf{x}_j - \bar{\mathbf{x}}), \tag{2.123}$$

$$\Rightarrow \quad \boxed{\widetilde{D}_{ij} \doteq \frac{1}{A}\int (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x}))\cdot(\mathbf{S}_j(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x}))dA(\mathbf{x}),} \tag{2.124}$$

where A is the total surface area/length of the mean shape. In practice, such integrals can be evaluated by numerical integration techniques. We have chosen a common normalization for $\mathbf{D}$ and $\widetilde{\mathbf{D}}$ so that the equivalence of eigenvalues is maintained (Theorem 2.6). The connection between this common set of eigenvalues for the *normalized* covariance matrices and the variance along the PCA axes is now given by:

$$\boxed{\frac{1}{n_P}\sum_{i=1}^{n_S}\left((\mathbf{x}_i - \bar{\mathbf{x}})\cdot\mathbf{n}^{(a)}\right)^2 = \lambda_a.} \tag{2.125}$$

We would like to maintain the mapping from shape parameter vectors to shapes as just the generalization of (2.116):

$$\boxed{\mathbf{S}(\mathbf{x}) \doteq \bar{\mathbf{S}}(\mathbf{x}) + \sum_{i=1}^{n_S}(\mathbf{b}^{(i)}\cdot\mathbf{b})(\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})).} \tag{2.126}$$

However, to do this we need to look at the covariance matrix $\mathbf{D}$ and the meaning of the PCA directions $\{\mathbf{n}^{(a)}\}$ and the parameter vectors $\{\mathbf{b}^{(i)}\}$ in

the infinite-dimensional limit, and construct the infinite-dimensional analog of Theorem 2.6.

Theorem 2.7. Equivalence of Eigenvectors/Eigenfunctions: Infinite-Dimensional Case.
Consider the covariance matrix (2.124):

$$\widetilde{\mathbf{D}} \doteq \{\widetilde{D}_{ij} : i, j = 1, \dots n_S\}, \tag{2.127}$$

$$\widetilde{D}_{ij} \doteq \frac{1}{A} \int (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) \cdot (\mathbf{S}_j(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) dA(\mathbf{x}), \tag{2.128}$$

with eigenvectors and (non-zero) eigenvalues:

$$\widetilde{\mathbf{D}}\widetilde{\mathbf{n}}^{(a)} = \lambda_a \widetilde{\mathbf{n}}^{(a)}, \;\; \lambda_a \neq 0. \tag{2.129}$$

Consider also the matrix-valued shape covariance function $\mathbf{D}(\mathbf{y}, \mathbf{x})$ *with elements:*

$$\boxed{D_{\nu\mu}(\mathbf{y}, \mathbf{x}) \doteq \frac{1}{A}(S_{i\nu}(\mathbf{y}) - \bar{S}_\nu(\mathbf{y}))(S_{i\mu}(\mathbf{x}) - \bar{S}_\mu(\mathbf{x}))} \tag{2.130}$$

and the general integral eigenproblem:

$$\int D_{\nu\mu}(\mathbf{y}, \mathbf{x}) n_\mu^{(a)}(\mathbf{x}) dA(\mathbf{x}) = \lambda_a n_\nu^{(a)}(\mathbf{y}) \tag{2.131}$$

$$\Rightarrow \quad \boxed{\int \mathbf{D}(\mathbf{y}, \mathbf{x}) \mathbf{n}^{(a)}(\mathbf{x}) dA(\mathbf{x}) = \lambda_a \mathbf{n}^{(a)}(\mathbf{y}), \;\; \lambda_a \neq 0.} \tag{2.132}$$

There then exists a one-to-one mapping between the eigenvectors of $\widetilde{\mathbf{D}}$ *with non-zero eigenvalue, and the eigenfunctions of* $\mathbf{D}(\mathbf{y}, \mathbf{x})$ *with non-zero eigenvalues, and the eigenvalues for corresponding eigenvectors/functions are equal.*

Proof. For parameterised shapes $\{S_i\}$ *in* $\mathbb{R}^d$*, we define:*

$$\widetilde{\mathbf{S}}_i(\mathbf{x}) \doteq \mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x}), \;\; \widetilde{\mathbf{S}}_i(\mathbf{x}) \doteq \{\widetilde{S}_{i\mu}(\mathbf{x}) : \mu = 1, \dots d\}. \tag{2.133}$$

Then the eigenvector equation for $\widetilde{\mathbf{D}}$ *can be written as:*

$$\frac{1}{A} \int \widetilde{S}_{i\mu}(\mathbf{x}) \widetilde{S}_{j\mu}(\mathbf{x}) \widetilde{n}_j^{(a)} dA(\mathbf{x}) = \lambda_a \widetilde{n}_i^{(a)}. \tag{2.134}$$

If we multiply by $\widetilde{S}_{i\nu}(\mathbf{y})$ *and sum over* i*, we obtain:*

$$\frac{1}{A} \int \widetilde{S}_{i\mu}(\mathbf{x}) \widetilde{S}_{i\nu}(\mathbf{y}) \left(\widetilde{S}_{j\mu}(\mathbf{x}) \widetilde{n}_j^{(a)} \right) = \lambda_a \left(\widetilde{S}_{i\nu}(\mathbf{x}) \widetilde{n}_i^{(a)} \right). \tag{2.135}$$

If we define the vector-valued function:

$$\mathbf{n}^{(a)}(\mathbf{x}) = \{n_\mu^{(a)}(\mathbf{x}) : \mu = 1, \dots d\},\ \ n_\mu^{(a)}(\mathbf{x}) \doteq \left(\widetilde{S}_{i\mu}(\mathbf{x})\widetilde{n}_i^{(a)}\right), \tag{2.136}$$

and using the definition of the matrix-valued covariance function:

$$\mathbf{D}(\mathbf{y}, \mathbf{x}) \doteq \{D_{\nu\mu}(\mathbf{y}, \mathbf{x}) : \nu, \mu = 1, \dots d\},\ \ D_{\nu\mu}(\mathbf{y}, \mathbf{x}) \doteq \frac{1}{A}\widetilde{S}_{i\nu}(\mathbf{y})\widetilde{S}_{i\mu}(\mathbf{x}), \tag{2.137}$$

then the eigenvector equation can be rewritten as:

$$\int D_{\nu\mu}(\mathbf{y}, \mathbf{x}) n_\mu^{(a)}(\mathbf{x}) dA(\mathbf{x}) = \lambda_a n_\nu^{(a)}(\mathbf{y}) \tag{2.138}$$

$$\Rightarrow \quad \int \mathbf{D}(\mathbf{y}, \mathbf{x})\mathbf{n}^{(a)}(\mathbf{x}) dA(\mathbf{x}) = \lambda_a \mathbf{n}^{(a)}(\mathbf{y}). \tag{2.139}$$

Hence $\mathbf{n}^{(a)}(\mathbf{y})$ *is a solution of the required eigenproblem, with matching eigenvalue* λ_a*.*

Similarly, if we start from a solution to this eigenproblem, and take the dot product of both sides with $\widetilde{S}_i(\mathbf{y})$ *and integrate over* $dA(\mathbf{y})$*, we obtain:*

$$\frac{1}{A}\int \widetilde{S}_{j\nu}(\mathbf{y})\widetilde{S}_{j\mu}(\mathbf{x}) n_\mu^{(a)}(\mathbf{x})\widetilde{S}_{i\nu}(\mathbf{y}) dA(\mathbf{x}) dA(\mathbf{y}) = \lambda_a \int n_\nu^{(a)}(\mathbf{y})\widetilde{S}_{i\nu}(\mathbf{y}) dA(\mathbf{y}). \tag{2.140}$$

If we define:

$$\widetilde{\mathbf{n}}^{(a)} \doteq \{\widetilde{n}_i^{(a)} : i = 1, \dots n_S\},\ \ \widetilde{n}_i^{(a)} \doteq \int n_\nu^{(a)}(\mathbf{y})\widetilde{S}_{i\nu}(\mathbf{y}) dA(\mathbf{y}), \tag{2.141}$$

then we have:

$$\frac{1}{A}\int \widetilde{S}_{j\nu}(\mathbf{y})\widetilde{S}_{i\nu}(\mathbf{y})\widetilde{n}_j^{(a)} dA(\mathbf{y}) = \lambda_a \widetilde{n}_i^{(a)}, \tag{2.142}$$

$$\Rightarrow \quad \widetilde{D}_{ij}\widetilde{n}_j^{(a)} = \lambda_a \widetilde{n}_i^{(a)} \ \Rightarrow\ \widetilde{\mathbf{D}}\widetilde{\mathbf{n}}^{(a)} = \lambda_a \widetilde{\mathbf{n}}^{(a)}, \tag{2.143}$$

which gives a solution to the other eigenproblem. We hence have a one-to-one mapping between the solutions of the two eigenproblems with non-zero eigenvalues. □

We hence have shown the equivalence of the two eigenproblems in the infinite-dimensional case, just as Theorem 2.6 showed their equivalence in the finite-dimensional case.

The integral eigenproblem (2.130) is just the infinite-dimensional analog of the finite-dimensional eigenvector problem for the normalized covariance matrix $\mathbf{D}$ (2.122). The eigenfunctions $\{\mathbf{n}^{(a)}(\mathbf{x})\}$ are then the infinite-dimensional

analog of the eigenvectors $\{\mathbf{n}^{(a)}\}$ that we introduced when we considered PCA ((2.17) and Theorem 2.1). As in the finite-dimensional case, these eigenfunctions are both left and right eigenfunctions (since $\mathbf{D}(\mathbf{y}, \mathbf{x})$ is symmetric), and it then follows that eigenfunctions belonging to different eigenvalues are orthogonal, where we define the equivalent of the dot product between these vector-valued eigenfunctions as follows:

$$\int \mathbf{n}^{(a)}(\mathbf{x}) \cdot \mathbf{n}^{(b)}(\mathbf{x}) dA(\mathbf{x}). \tag{2.144}$$

We can hence define an orthonormal set of eigenfunctions, so that:

$$\boxed{\int \mathbf{n}^{(a)}(\mathbf{x}) \cdot \mathbf{n}^{(b)}(\mathbf{x}) dA(\mathbf{x}) \doteq \delta_{ab}.} \tag{2.145}$$

Following the finite-dimensional case (2.34), we introduce parameter vectors $\{\mathbf{b}^{(i)} : i = 1, \dots n_m\}$, which are defined as:

$$\boxed{\mathbf{b}^{(i)} \doteq \{b_a^{(i)} : a = 1, \dots n_S\},\ b_a^{(i)} \doteq \int \mathbf{n}^{(a)}(\mathbf{x}) \cdot \widetilde{\mathbf{S}}_i(\mathbf{x}) dA(\mathbf{x}).} \tag{2.146}$$

From (2.141), we see that as before (2.113):

$$\boxed{\widetilde{n}_i^{(a)} = b_a^{(i)}.} \tag{2.147}$$

The size of the vectors $\widetilde{\mathbf{n}}^{(a)}$ is however slightly different:

$$\boxed{\|\widetilde{\mathbf{n}}^{(a)}\|^2 = A\lambda_a.} \tag{2.148}$$

Putting all this together, it means that rather than trying to solve the integral eigenproblem, we can instead solve for the eigenvalues and eigenvectors of the covariance matrix $\widetilde{\mathbf{D}}$ (2.124). The integral over the mean shape in the covariance matrix can be solved using a numerical approximation. We then apply the above normalization to the vectors $\widetilde{\mathbf{n}}^{(a)}$, and hence obtain the shape parameter vectors $\mathbf{b}^{(i)}$.

In the finite-dimensional case, the shapes vectors could be expanded in terms of the PCA eigenvector basis, the coefficients being the shape parameter vectors. In the infinite-dimensional case, the shape functions can be expanded in terms of the eigenfunctions. If we define:

$$\widetilde{\mathbf{S}}_i(\mathbf{x}) \approx \sum_{a=1}^{n_m} c_{ia} \mathbf{n}^{(a)}(\mathbf{x}), \tag{2.149}$$

then by taking the dot product with $\mathbf{n}^{(a)}(\mathbf{x})$ we find that:

$$c_{ia} = \int \widetilde{\mathbf{S}}_i(\mathbf{x}) \cdot \mathbf{n}^{(a)}(\mathbf{x}) dA(\mathbf{x}) = b_a^{(i)}, \tag{2.150}$$

$$\widetilde{\mathbf{S}}_i(\mathbf{x}) \approx \sum_{a=1}^{n_m} b_a^{(i)} \mathbf{n}^{(a)}(\mathbf{x}) \;\Rightarrow\; \mathbf{S}_i(\mathbf{x}) \approx \bar{\mathbf{S}}(\mathbf{x}) + \sum_{a=1}^{n_m} b_a^{(i)} \mathbf{n}^{(a)}(\mathbf{x}). \tag{2.151}$$

This shape representation is only approximate since we are not necessarily using all the eigenvectors, but only the first n_m (note that, as before, we assume the eigenvalues are arranged in decreasing order).

For a general shape generated by a parameter vector:

$$\mathbf{b} \doteq \{b_a : a = 1, \ldots n_m\},$$

we have that:

$$\mathbf{S}(\mathbf{x}) \doteq \bar{\mathbf{S}}(\mathbf{x}) + \sum_{a=1}^{n_m} b_a \mathbf{n}^{(a)}(\mathbf{x}). \tag{2.152}$$

As in (2.116), this can also be rewritten as follows:

$$\text{From (2.136:)} \quad \mathbf{n}^{(a)}(\mathbf{x}) = \{n_\mu^{(a)}(\mathbf{x}) : \mu = 1, \ldots d\}, \tag{2.153}$$

$$n_\mu^{(a)}(\mathbf{x}) \doteq \left(\widetilde{S}_{i\mu}(\mathbf{x}) \widetilde{n}_i^{(a)}\right) = \sum_{i=1}^{n_S} \left(\widetilde{S}_{i\mu}(\mathbf{x}) b_a^{(i)}\right). \tag{2.154}$$

$$\therefore \quad \mathbf{S}(\mathbf{x}) \doteq \bar{\mathbf{S}}(\mathbf{x}) + \sum_{a=1}^{n_m} b_a \mathbf{n}^{(a)}(\mathbf{x}) \tag{2.155}$$

$$= \bar{\mathbf{S}}(\mathbf{x}) + \sum_{a=1}^{n_m} b_a \sum_{i=1}^{n_S} \left(\widetilde{\mathbf{S}}_i(\mathbf{x}) b_a^{(i)}\right) \tag{2.156}$$

$$\Rightarrow \quad \boxed{\mathbf{S}(\mathbf{x}) = \bar{\mathbf{S}}(\mathbf{x}) + \sum_{i=1}^{n_S} \left(\mathbf{b} \cdot \mathbf{b}^{(i)}\right) \widetilde{\mathbf{S}}_i(\mathbf{x}).} \tag{2.157}$$

To summarize, in the infinite-dimensional case, we have the infinite-dimensional parameterised shapes $\{\mathbf{S}_i(\mathbf{x})\}$, and we can perform PCA on these as is given in Algorithm 2.2.

The point to note is that here we have used PCA to perform a radical dimensional reduction, taking us from the space of infinite-dimensional shapes, to the finite-dimensional space of shape parameter vectors. The use of the alternative covariance matrix means that the only infinite-dimensional objects we need to consider are the input shapes themselves, since all further objects are finite-dimensional. The only approximation required is in the initial calculation of the covariance matrix, where we have to use numerical methods to perform the area integral. The previous statements about the link between the shape parameter vectors and variance in the various PCA directions still hold, given the definition we have already used for vector dot products in the

Algorithm 2.2 : PCA for Infinite-Dimensional Shapes.

- Construct the *finite-dimensional* covariance matrix $\widetilde{\mathbf{D}}$ (2.124) by performing numerical integration over the mean shape.
- Solve for the finite-dimensional eigenvectors $\{\widetilde{\mathbf{n}}^{(\mathbf{a})}\}$ and eigenvalues $\{\lambda_a\}$ of $\widetilde{\mathbf{D}}$.
- Normalize the eigenvectors so that $\|\widetilde{\mathbf{n}}^{(\mathbf{a})}\|^{\mathbf{2}} = \mathbf{A}\lambda_{\mathbf{a}}$.
- Construct the shape parameter vectors, where $\widetilde{n}_i^{(a)} = b_a^{(i)}$.
- We then can generate shapes from a shape vector $\mathbf{b}$:

$$\mathbf{S}(\mathbf{x}) = \bar{\mathbf{S}}(\mathbf{x}) + \sum_{\mathbf{i=1}}^{\mathbf{n_S}} \left(\mathbf{b} \cdot \mathbf{b}^{(\mathbf{i})}\right) \widetilde{\mathbf{S}}_{\mathbf{i}}(\mathbf{x}). \tag{2.158}$$

infinite-dimensional space (2.144). Specifically, the mean variance per shape in the direction $\mathbf{n}^{(a)}(\mathbf{x})$ is given by:

$$\widetilde{\mathbf{S}}_i \cdot \mathbf{n}^{(a)} \doteq \int \widetilde{\mathbf{S}}_i(\mathbf{x}) \cdot \mathbf{n}^{(a)}(\mathbf{x}) dA(\mathbf{x}) = b_a^{(i)}. \tag{2.159}$$

$$\therefore \quad \sum_{i=1}^{n_S} \left(\widetilde{\mathbf{S}}_i \cdot \mathbf{n}^{(a)}\right)^2 = \sum_{i=1}^{n_S} \left(b_a^{(i)}\right)^2 = \sum_{i=1}^{n_S} \left(\widetilde{n}_i^{(a)}\right)^2 = \|\widetilde{\mathbf{n}}^{(a)}\|^2 = A\lambda_a.$$

$$\Rightarrow \quad \boxed{\frac{1}{A}\sum_{i=1}^{n_S} \left((\mathbf{S}_i - \bar{\mathbf{S}}) \cdot \mathbf{n}^{(a)}\right)^2 = \lambda_a.} \tag{2.160}$$

Previously (2.125), we had that λ_a represented the summed variance for all shapes about the mean shape in the direction $\mathbf{n}^{(a)}$, normalized by the number of shape points n_P. Here we see that we have the corresponding expression, but normalized to give the variance per unit area of the mean shape.

Since we have now projected our original data from the infinite-dimensional space of shapes to the finite-dimensional space of shape parameter vectors, the modelling of the distribution of parameter vectors proceeds as before.

The notation used for finite and infinite-dimensional shape representations, and the details of the PCA eigenproblems are summarized in Table 2.2.

2.4 Applications of Shape Models

In the previous sections (Sects. 2.1 and 2.3), we have shown how statistical shape models can be built from training sets of shapes, and how principal component and density estimation techniques can be applied to characterize the distribution of shapes. If all we wish to do is analyse the distribution of

Table 2.2 Summary of the notation and conventions used in the text for finite and infinite-dimensional shape representations, covariance matrices, and PCA eigenproblems. Continues on next page.

Finite Dimensional		**Infinite Dimensional**
Shapes and Shape Representation		
$\{S_i \subset \mathbb{R}^d : i = 1, \dots n_S\}$ $\{1, 2, \dots n_P\} \overset{\mathbf{x}_i}{\longmapsto} S_i,\ j \overset{\mathbf{x}_i}{\longmapsto} \mathbf{x}_i^{(j)} \in S_i$ $S_i \to \mathbf{x}_i \doteq \{\mathbf{x}_i^{(j)} : j = 1, \dots n_P\}$ $\bar{\mathbf{x}} \doteq \frac{1}{n_S} \sum_{i=1}^{n_S} \mathbf{x}_i$		$\{S_i \subset \mathbb{R}^d : i = 1, \dots n_S\}$ $X \overset{\mathbf{x}_i}{\longmapsto} S_i,\ \mathbf{x} \overset{\mathbf{x}_i}{\longmapsto} \mathbf{S}_i(\mathbf{x})$ $S_i \to \mathbf{S}_i \doteq \{\mathbf{S}_i(\mathbf{x}) : \mathbf{x} \in X\}$ $\bar{\mathbf{S}}(\mathbf{x}) \doteq \frac{1}{n_S} \sum_{i=1}^{n_S} \mathbf{S}_i(\mathbf{x})$ $\widetilde{\mathbf{S}}_i(\mathbf{x}) \doteq \mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})$
Covariance Matrices		
$\mathbf{D}$, $D_{\mu\nu} \doteq (\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_i - \bar{\mathbf{x}})_\mu$ $\widetilde{\mathbf{D}}$, $\widetilde{D}_{ij} \doteq (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot (\mathbf{x}_j - \bar{\mathbf{x}})$	$\mathbf{D}$, $D_{\mu\nu} \doteq \frac{1}{n_P} (\mathbf{x}_i - \bar{\mathbf{x}})_\mu (\mathbf{x}_i - \bar{\mathbf{x}})_\mu$ $\widetilde{\mathbf{D}}$, $\widetilde{D}_{ij} \doteq \frac{1}{n_P} (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot (\mathbf{x}_j - \bar{\mathbf{x}})$	$\mathbf{D}(\mathbf{y}, \mathbf{x})$, $D_{\mu\nu}(\mathbf{x}, \mathbf{y}) \doteq \frac{1}{A} (S_{i\mu}(\mathbf{x}) - \bar{S}_\mu(\mathbf{x}))(S_{i\nu}(\mathbf{y}) - \bar{S}_\nu(\mathbf{y}))$ $\widetilde{\mathbf{D}}$, $\widetilde{D}_{ij} \doteq \frac{1}{A} \int (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) \cdot (\mathbf{S}_j(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) dA(\mathbf{x})$
Eigenproblems		
$\mathbf{D}\mathbf{n}^{(a)} = \lambda_a \mathbf{n}^{(a)}$		$\int \mathbf{D}(\mathbf{y}, \mathbf{x}) \mathbf{n}^{(a)}(\mathbf{x}) dA(\mathbf{x}) = \lambda_a \mathbf{n}^{(a)}(\mathbf{y})$
$\widetilde{\mathbf{D}} \widetilde{\mathbf{n}}^{(a)} = \lambda_a \widetilde{\mathbf{n}}^{(a)}$		

Table 2.2 Summary of the notation and conventions used in the text for finite and infinite-dimensional shape representations, covariance matrices, and PCA eigenproblems. Continued from previous page.

Finite Dimensional		**Infinite Dimensional**
Eigenvectors and Eigenvalues		
$\mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} = \delta_{ab}$		$\int \mathbf{n}^{(a)}(\mathbf{x}) \cdot \mathbf{n}^{(b)}(\mathbf{x}) dA(\mathbf{x}) = \delta_{ab}$
$\lambda_a = \sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right)^2$	$\lambda_a = \frac{1}{n_P} \sum_{i=1}^{n_S} \left((\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{n}^{(a)} \right)^2$	$\lambda_a = \frac{1}{A} \sum_{i=1}^{n_S} \left((\mathbf{S}_i - \bar{\mathbf{S}}) \cdot \mathbf{n}^{(a)} \right)^2$ $(\mathbf{S}_i - \bar{\mathbf{S}}) \cdot \mathbf{n}^{(a)} \doteq \int (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) \cdot \mathbf{n}^{(a)}(\mathbf{x}) dA(\mathbf{x})$
$\widetilde{\mathbf{n}}^{(a)} \doteq \{\widetilde{n}_i^{(a)} : i = 1, \dots n_S\}$		
$\widetilde{n}_i^{(a)} \doteq \mathbf{n}^{(a)} \cdot (\mathbf{x}_i - \bar{\mathbf{x}})$		$\widetilde{n}_i^{(a)} \doteq \int \mathbf{n}^{(a)} \cdot (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) dA(\mathbf{x})$
$\|\widetilde{\mathbf{n}}^{(a)}\|^2 \doteq \lambda_a$	$\|\widetilde{\mathbf{n}}^{(a)}\|^2 \doteq n_P \lambda_a$	$\|\widetilde{\mathbf{n}}^{(a)}\|^2 \doteq A\lambda_a$
Parameter Vectors		
$\mathbf{b}^{(i)} = \{b_a^{(i)} : a = 1, \dots n_m\}$		
$b_a^{(i)} \doteq \mathbf{n}^{(a)} \cdot (\mathbf{x}_i - \bar{\mathbf{x}})$ $\mathbf{x}_i \approx \bar{\mathbf{x}} + \sum_{a=1}^{n_S} b_a^{(i)} \mathbf{n}^{(a)}, \;\; \mathbf{x} \doteq \bar{\mathbf{x}} + \sum_{a=1}^{n_S} b_a \mathbf{n}^{(a)} = \bar{\mathbf{x}} + \sum_{i=1}^{n_S} \left(\mathbf{b}^{(i)} \cdot \mathbf{b} \right) (\mathbf{x}_i - \bar{\mathbf{x}})$		$b_a^{(i)} \doteq \int \mathbf{n}^{(a)}(\mathbf{x}) \cdot (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x})) dA(\mathbf{x})$ $\mathbf{S}(\mathbf{x}) = \bar{\mathbf{S}}(\mathbf{x}) + \sum_{i=1}^{n_S} \left(\mathbf{b}^{(i)} \cdot \mathbf{b} \right) (\mathbf{S}_i(\mathbf{x}) - \bar{\mathbf{S}}(\mathbf{x}))$

shapes across the training set, this is often sufficient. For example, we can use the shape of the estimated density or information from principal components to classify subsets of shapes within the training set. Principal component analysis can also tell us about the major and minor modes of shape variation seen across the training set, and provide an intuitive picture of the way the shapes vary.

If we are given an unseen shape, one which was not included in our training set, we can use the same techniques to analyse this new shape. For example, we can decide whether it is like or unlike those seen previously, to what category of shape it belongs, or describe in what way it varies from what we have already seen. We can hence describe this new shape within a wider context of learnt information about this type of shape.

This however presumes that we already have our unseen shape. In many computer vision or medical imaging applications that study shape, the shapes themselves are obtained from images. The most intuitive, and the simplest, way of extracting the shape from the image is to use manual annotation. However, when there are a large number of examples to process, this can become extremely time-consuming. For the case of images in three dimensions, such as those encountered in medical imaging, this annotation can become very difficult.

There are many basic methods for automatically segmenting images [169]. These typically use information such as the colour/greyscale values and texture in the image to identify regions of the image, and information about edge structures in the image to try to delineate the boundaries of such regions or sets of regions that constitute the shape of the imaged object. This can work well provided the shapes are relatively simple, or have good texture/colour cues, or where we do not know what shapes we expect to see in an image. However, for cases where we are looking for a particular known object, the most promising approaches are those which adopt a learning framework. Such systems proceed in much the same way that a human annotator would proceed. The trainee human annotator or computer system is first presented with a set of training examples which have been previously annotated by some expert. Based on what has been learnt from these examples as to the shape which is required, and how it varies, the trainee system or human then annotates examples, possibly with some continuing feedback from an expert to correct mis-annotation.

Such a system can be constructed using a statistical shape model to encode the learnt information about shape. Two algorithms which use such a system for automatic image annotation are the Active Shape Model (ASM) [29, 39] and Active Appearance Model (AAM) [26, 25, 27].

2.4.1 Active Shape Models

Suppose we have an image that contains an example of an object we are trying to annotate with a set of shape points. In general terms, there are several components that help us differentiate a successful from an unsuccessful annotation.

First, we have the global constraint that the set of points should describe a shape which is, according to what we have learnt about shape, a valid example of that object. Secondly, we also have the local constraint that the shape points should lie on edges or structures in the image that look like the locations where such points have been placed in the annotated training examples.

Given an initial guess as to the shape, these two constraints can be used in tandem to home in on the correct position of the shape. Essentially, for each current shape point, we search in the neighbourhood of that point to see if there is a candidate position which better fits the expected appearance of the image. Given such a set of candidate positions, we then apply the global constraint, by replacing the candidate shape by a shape which is as close as possible to the candidate shape (hence fits the local constraints), yet is a valid shape as far as the global constraint is concerned. The process is then iterated until it converges on a final answer. This is the basic Active Shape Model search algorithm.

The global constraint is applied by quantifying the training information about shape and shape variation in terms of a statistical shape model. For a candidate shape, the positions of the candidate points are encoded in terms of a shape vector $\mathbf{x}$ as described previously (2.2). This shape is then Procrustes aligned (Sect. 2.1.1) with the mean shape $\bar{\mathbf{x}}$ from the SSM, to remove unimportant details such as the precise scale and orientation of the object. We then project this shape vector into the subspace of shape space described by the SSM, and evaluate its proximity to the training set of shapes. The first stage typically means extracting the PCA components of the candidate shape as in (2.38), which gives us an approximation representation of the candidate shape in terms of a set of shape parameters $\mathbf{b}$.

We then have to evaluate the proximity of the point $\mathbf{b}$ to the training set of shapes. For PCA components, we can constrain each component individually, forcing the shape to lie within a bounding parallelepiped as described in Sect. 2.2.4. Alternatively, for cases where a density estimate is available (e.g., as in Sects. 2.2.1–2.2.3), we can restrict the minimum allowed value of $p(\mathbf{b})$. For points that do not initially satisfy the constraint, we can evolve the point $\mathbf{b}$ through gradient ascent of $p(\mathbf{b})$ until the constraint is satisfied. For Gaussian density models, this process is considerably simplified, given the monotonic relationship of Mahalanobis distance and the probability density as described in Sect. 2.2.4, and it is sufficient to move the point $\mathbf{b}$ inwards along the ray connecting $\mathbf{b}$ to the origin of parameter space until the constraint is satisfied. It should be noted that setting the appropriate limits is

important. Setting them too high at the beginning of the search can overly constrain the shape, and not allow enough freedom in moving through the search space to locate the optimum fitted shape, or allow only solutions which are very close to the training shapes. Whereas too loose a constraint can allow the search process to get stuck in local minima, fitting to shapes far from optimum.

The local part of the ASM search is built on learning about the local image appearance in the neighbourhood of the shape points. For each example of a specific shape point on each training example, the normal to the shape at that point is constructed. The image intensity values are then sampled along this normal to form an image profile. This set of image profiles from each example is then combined into a statistical profile model in the same general manner as for shape models. When searching for a new candidate position for a shape point on an unseen image during search, profiles are sampled in the vicinity, and the profile that best fits the profile model for that point is selected as the new candidate position.

There is an extensive and still growing research literature as regards Active Shape Models, with various variations on the basic ASM described above (e.g., see [40, 33, 39, 28, 34], and the reader should consult the appropriate literature for full details (see [32, 37] for reviews).

2.4.2 Active Appearance Models

The ASM search performs extremely well on some types of data. However, the model uses only limited image information to locate the shape. In the Active Appearance Model (AAM) [26, 25, 27], the training process incorporates information about the expected appearance within the entire interior of the annotated shape, rather than just samples of image information in the neighbourhood of the shape points.

For the annotated training images, the shape part of the model is constructed as before. We obviously cannot just combine the raw image information from the interiors of all the training shapes, but have first to convert this information into a common frame of reference. This is done using the shape information, since this tells us the varying positions of corresponding shape points across the whole set of training examples. If we interpolate the interior of each shape, based on the shape points, this then gives us, by interpolation, a correspondence between the interiors of each shape across the whole set. We then map each training shape to the mean shape, and resample the image information from each shape into the frame of the mean shape. This gives us a set of shape-free texture examples, one from each training image. The pixel-value information for each shape-free texture example is then concatenated into a vector, with the entire training set then giving us a set of data points in a shape-free texture space. The distribution of points in

shape-free texture space can then be analysed and modelled using the same techniques as those used for modelling shape spaces. We then have both a statistical model of shape, and a statistical model of texture (essentially a type of shape-free eigenface model [179]). Using these statistical models in generative mode, we can then create shape-free variations of texture and modes of variation of texture. Using the reverse of the initial mapping from texture on a shape to texture on the mean shape, we can also vary shape whilst leaving texture unchanged.

For many imaging examples, there is a significant correlation between shape and texture. One obvious example is two-dimensional images of faces. It is obvious that, for a fixed illumination, as the pose of the subject changes, the texture (i.e., the positions of highlights and shadows) changes, and is correlated with the changes in shape. Even without change of pose, the shape of the face changes under changes of expression, and the texture changes in a correlated fashion.

These correlations can be modelled by concatenating the shape vector and the texture vector of each training example into a single vector. A statistical model of appearance is then built in the usual manner in this combined space of shape and texture, and generates modes of variation that capture the correlations noted above.

The search algorithm for the AAM is slightly more complicated than for the ASM (and we refer readers to the literature for the details [26, 25, 27]). However, the basic rationale is the same as for the ASM, where the learnt information about permissable levels of variation is incorporated into and constrains the search process.

The statistical appearance model, like the statistical shape model, can also be applied in a generative mode. By sampling the space of parameters according to the modelled pdf, we can generate an arbitrarily large number of artificial examples of the modelled object.

For the case of faces, these artificially generated examples can be almost photo-realistic. Analysis of the space of the model can separate out the subspaces corresponding to varying lighting, pose, identity, and expression [44, 42]. This means that given an image of an unseen individual, we can generate examples of this same individual, but apply different expressions. If information about the gender of subjects across the training set is available, it is also possible to manipulate the perceived gender, making a face appear more masculine or more feminine according to what has been learnt from the training set about the way faces tend to differ with gender [43]. Similarly, it is also possible to simulate ageing [104] (see Chap. 1).

The power and flexibility of the ASM/AAM approach has led to their usage in a large (and still growing) number of applications in computer vision and medical imaging. Both approaches require an initial statistical shape model for their implementation, and the quality of this initial model is a prime determining factor in ensuring the quality of the final system. Hence

establishing a suitable correspondence is a key step in the model-building process, and one that we address in greater detail in the next chapter.