A Bivariate Finite Element Smoothing Spline Applied to Image Registration

by

Tim Ramsay

A thesis submitted to the Department of Mathematics and Statistics in conformity with the requirements for the degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

December 1999

Acknowledgements

I would like to thank Dr. Agnes Herzberg for convincing me to study statistics. Were it not for her influence, I might never have become interested in the subject. I would like to thank my father, Dr. James Ramsay, not only for playing an important role in my coming to be here in this world but also for sparking my interest in the exciting field of functional data analysis. I would also like to thank both of my supervisors, Drs. Lorne Campbell and James McLellan for having enough faith in me to support my research in a field with which neither was familiar. The funding for this research came, at least in part, from the Natural Sciences and Engineering Research Council.

I am very grateful to Dr. J. Terry Smith for allowing me to work in the Queen's Statistical Consultation Service. I undoubtedly learned more statistics working there with Terry than I did in all of my statistics courses combined.

I need to thank Premier Harris for his continuing dedication to the support of higher education in this fine province.

Lastly, I would like to thank Loan Nguyen whose love and fine cooking have sustained me throughout the entire course of working for this degree and Abscissa for giving me a reason to live.

Abstract

The branch of statistics known as *functional data analysis* seeks to analyze data which presents itself in the form of functions. A unique characteristic of functional data is that it displays two very different types of variation. Variation in range is variation in the value of a function over different realizations of that function. Variation in domain is variation in the point in the domain of the function at which a particular value, or event, occurs.

Functional registration, a technique for separating range variation from domain variation, involves continuously deforming the domain of each function so as to make certain known events, called *landmarks*, occur at the same points in the domain of each observation. The observed functions of the deformed domains are called *registered functions* and can be thought of as containing all of the observed range variation without any of the domain variation. The domain deformation functions used to register the observations are called *registration functions* and can be thought of as containing all of the observed domain variation without any of the range variation.

In order to estimate registration functions for a set of functional observations containing landmarks, it is necessary to estimate smooth continuous functions from discrete observations at arbitrarily spaced points in the domain. In one dimension, this estimation can be performed using various *spline functions* and good functional registration techniques based on spline functions are available. The existing techniques for arbitrarily scattered points in the plane, on the other hand, are less convenient than spline techniques for estimating functions of $\mathbb{R}$.

This thesis presents a method for constructing finite-element approximations of a class of bivariate smoothing splines analogous to one-dimensional L-splines. These functions are applied to a sample bivariate registration problem and the quality of the registration is analyzed through a variety of simulations. It is concluded that the functional estimates are an effective tool for bivariate registration. The methods presented can be generalized to estimate functions of 3 variables.

Contents

List of Figures

List of Tables

Chapter 1

Introduction: Functional Data

Functional data analysis, a relatively new field of statistics, deals with the statistical analysis of continuous, observable functions. It has been successfully applied to a wide variety of functions in areas as diverse as biology, biomechanics, medicine, meteorology, and engineering. Examples of such functions include the height of a growing organism, the movement of the knee while walking, brain structure as observed in medical images, temperature records, and pressure in an internal combustion engine. All of these examples are functions either of time or of space which can both be repeatedly observed and be expected to vary in a variety of ways from observation to observation. It is this inherent variability which makes such observations a suitable area for the application of statistical methods.

Before functional data can be statistically analyzed, two forms of preprocessing must first take place. The first, and perhaps most obvious, step is that of estimating continuous functions from the observations. Even though many functions are continuous and can be observed at any point in their domain, one can, in general, only measure them discretely. The location of an individual's knee, for example, will be recorded as a series of position values recorded at consecutive time points. Similarly, a digital image of a brain is a discrete set of intensity values sampled at the locations of the pixels. Before such data can be analyzed in the form of continuous functions, therefore, it is necessary to estimate continuous functions from discrete observations.

If the estimate passes through the data points, the points are said to be interpolated by the estimate.

In addition to discreteness, another aspect of functional data is that it is usually intrinsically noisy. *Smoothing* is the term typically used to describe the process of removing as much of this noise as possible. The term 'smoothing' arises from the fact that a smooth function, when observed discretely in the presence of noise, will often appear to be rather rough. When one has an observation of a function which is assumed to be more or less smooth, refining a rough estimate to make it smoother corresponds to an attempt to remove noise. This smoothing can be done as a second step, after the initial estimate has been computed, or else it can be incorporated into the estimation process itself. In either case, smoothing is often an essential part of estimating a continuous function from a discrete sample.

Registration, a process whereby the *domain* of a sample of functional observations is transformed, is often a necessary precursor to obtaining a reasonable estimate of the mean function. The need for registration arises from the fact that the variation present in a given function is often of two very different types, namely variation in the range of the function and variation in the domain. Consider, as an example, measurements of the heights of the left knee of an individual taken as he or she walks. One can easily imagine that the function representing the height will be cyclic, rising to a maximum height as the leg is lifted and the falling to a minimum as the foot flattens on the ground. The maximum height attained will certainly vary between individuals and can even be expected to exhibit some variation in one given individual. Variation in the height at a given point in the cycle is variation in the range. The time taken to complete a cycle, however, will also vary. For example, depending on the speed of walking, the point 50 milliseconds after the left foot is lifted off the ground will not always correspond to the same point in the overall gait cycle. This type of variation is called variation in domain, as the variation occurs not in the value measured, but in the domain value at which it is observed.

Variation in domain complicates the process of estimating a mean function from a sample of observations. If one were to line up a set of knee height curves, beginning at a minimum and ending at the following minimum, the variation in timing would result in the curves ending at different times. This problem can be fairly easily solved by linearly rescaling each curve by a suitably chosen scaling factor, but one would then find that the maxima of the different curves did not occur at the same time. Registration is a process whereby the arguments of functional data are transformed, often non-linearly, in a smooth manner so that various identifiable features in the function being studied occur at common arguments in each of the observations. I have been able to find only one example in the literature of the methods of functional data analysis being applied to image data [31]. This lack of bivariate functional data analyses is due, at least in part, to the lack of generally applicable bivariate smoothing and registration techniques. This thesis presents a method which can be used to register bivariate data.

Different types of functional data can have different mathematical structures. For example, while the height of an organism is a single-valued function of one variable, namely time, the position of the knee in space is a vector-valued function of one variable and the structure of the brain is a function of two or three spatial coordinates. In the language of functional analysis, a function of several variables such as a function of the plane ($\mathbb{R}^2$) or of space ($\mathbb{R}^3$) is called a *multivariate function*, and a function with more than one value such as a parametrized curve is called a *vector-valued function*. In the language of statistics, on the other hand, data such as the position of a sensor in space in which a single observation corresponds to several values is called *multivariate data*. These different conventions of language can lead to a certain amount of confusion when speaking of functional data since expressions such as "multivariate data" or "an observation of a multivariate function" might be interpreted in more than one way.

To avoid such confusion, I propose to use the terms *vector-valued* for functions having more than one value and *multivariable* for functions depending on more than one variable. Therefore image data, which is the focus of this thesis, will be referred to as "bivariable" data. This indicates that the observed function depends on two variables, namely the x and y coordinates of the Cartesian plane. Multivariable data can also be vector-valued. For example, the RGB format image associates a triple of numbers to each pixel. This triple contains one number each to denote the intensity in the colours red, green, and blue. A black-and-white image, on the other hand, consists of one number for each pixel. That one number denotes a grey-scale value. Thus, an RGB image is a sample of a vector-valued bivariable function while a black-and-white image is a sample of a single-valued bivariable function. The position of a sensor in space, observed as a function of time, is an example of a vector-valued univariable function.

Univariable functions can always be represented as curves. A vector-valued univariable function can be displayed as a single parametrized curve in space or, equivalently, as several functions of the same independent variable. The position in space of an object over time, for example, would have the form $f(t) = [x(t), y(t), z(t)]$. While a variety of standard methods already exist in the literature for studying curve data, in many cases no comparable methods exist for the study of multivariable functional data. This thesis presents a method based on finite element analysis for registering bivariable functional data. Even though this method has only been fully developed for bivariable data, it is easily generalizable to three and higher dimensions.

Chapter 2 presents methods based on spline functions which can be used to estimate, smooth, and register univariable functional data. This chapter also shows why the functional registration problem in two dimensions is fundamentally different than that in one dimension. Chapter 3 contains a survey of existing methods of bivariable image registration and also shows why these methods might not be appropriate for functional data analysis. Chapter 4 defines a finite-element approximation to a generalized L-spline smoothing function and shows how this approximation can be used to register bivariable functional data. This chapter contains the original theoretical work in this thesis. The finite-element approximation is a modification of an existing method for approximating the solution to the biharmonic equation.

Chapter 5 explains how to compute the finite element approximation in an efficient manner. Chapter 6 contains a sample bivariable registration problem and uses the finite-element L-spline to register a sample of images of human faces. Chapter 7 presents the results of some computational experiments that suggest that the registration algorithm based on the finite-element L-spline is both accurate and robust. Finally, chapter 8 both summarizes the conclusions of the thesis and suggests some directions for further research. Chapters 4, 6, and 7 contain most of the original contributions of the thesis, although some of the suggestions for implementation in chapter 5 are based on personal experience.

Chapter 2

Univariable Methods of Function Registration and the Bivariable Problem

2.1 Curve Data and Functional Estimation

The functional data considered in the literature to date has almost invariably been in the form of curves with the independent variable often representing time. To simplify the following discussion of univariable functional data analysis, all data will be assumed to be functions of time. The reader should bear in mind, however, that the concepts presented are generally applicable to a univariable function of any continuous argument. Position, pressure, voltage, or solute concentration are examples of possible alternative independent variables. For a more comprehensive discussion of the methods of functional data analysis, the reader is referred to the book *Functional Data Analysis* by Ramsay and Silverman [44].

An observation of a univariable function is presented to the analyst as a time series: a sequence of pairs (t_i, x_i), for $i = 1, \ldots, n$, where x_i is the observed value of the function at the distinct time t_i. It is assumed that the pairs are ordered so that $t_1 < t_2 < \cdots < t_n$, but not that the time values are necessarily equally spaced. Univariable functional data can also represent a curve in the plane or in space, parametrized by time, in which case the observed values in the time series would have an x, y, and possibly a z component and would be of the form (t_i, x_i, y_i) or (t_i, x_i, y_i, z_i). This type of data can also be thought of as two or more different curves with the same argument. Measurements of the height of a tobacco plant on different days over the growing season or of the air temperature in Central Park at hourly intervals over a 24-hour period could be examples of single-valued functional data. The location of a radio-tagged caribou on successive days over a period of migration could be an example of functional data in the form of a parametrized curve in the plane.

Although one might well analyze such data using the classical methods of time-series analysis, especially if the data were sampled regularly in time, such an approach would ignore the fact that the data consists of measurements of a process which varies continuously and smoothly over time. These features of the underlying process, namely continuity and smoothness, can be used in two ways to extract much more information from the data than is available in time series analysis. Firstly, the knowledge that the underlying process is smooth and continuous is a form of prior knowledge about the data which can be used effectively both to impute its behaviour over the intervals between the times of observation and to remove some of the error in observation. Secondly, smoothness implies the existence of derivatives which can be used to examine the mechanical or physical properties of the process which produced the observations.

Before applying the ideas of functional analysis to data, it is first necessary to estimate continuous functions from the discrete observations. How best to do this estimation is a non-trivial problem, and depends greatly on one's prior beliefs about the functions being observed. Moreover, once the continuous functions have been estimated it is often necessary to modify these estimates before acceptable mean functions can be estimated and variation can be examined. The purpose of this process of modification, called registration, is to align the different observations of the same function so that various identifiable features in the function, called landmarks, occur at the same times across all of the observations. Like the first step of estimating functions from discrete data, the step of function registration also requires the estimation of continuous functions. Both of these steps can be executed using the spline functions presented in the following two sections.

2.2 Cubic Splines and Spline Smoothing

Given an ordered set of n points $\{(t_i, x_i)\}_{i=1}^{n}$, there are many different ways in which one might estimate a function $f : \mathbb{R} \to \mathbb{R}$ such that $f(t_i) = x_i$. In fact, any set of n or more linearly independent functions can be combined to form a linear combination which exactly interpolates the observations. One well-known way in which to obtain a smooth interpolating function is to use an interpolating polynomial. Interpolating polynomials have the advantage of being continuously differentiable as well as being easy to evaluate but also have the distinct disadvantage of tending to be rather wiggly. This wiggliness arises from the fact that, in general, a polynomial of degree $n-1$ is needed to interpolate n points and such a polynomial can have as many as $n-2$ critical points, each of which may correspond to a local maximum or minimum. This type of behaviour may provide an unsatisfactory estimate of a function which is expected to vary smoothly. For example, if one were to measure the air temperature in Central Park at hourly intervals, one would certainly not be happy with an estimate of the temperature function which had eleven local maxima and eleven local minima.

One alternative to interpolating polynomials of high degree is to estimate the function using piecewise polynomials in which the polynomial pieces are of lower degree. This approach retains the ease of evaluation and differentiation of polynomials while avoiding the wiggliness. Piecewise polynomial functions which are constructed so as to satisfy certain continuity constraints are known as **spline functions** and are widely used in the estimation of smooth univariable functions. A comprehensive reference on the theory and construction of spline functions is the book by Schumaker

[46]. One useful spline function is the cubic spline, defined as follows. Note that the derivative operator D, which will be used extensively throughout this thesis, is defined by

$$Df = f' \quad \forall f : \mathbb{R} \to \mathbb{R}.$$

Therefore the notation $D^m f$ means the mth derivative of the function f.

Definition 1 (Cubic Splines) *Let $t_0 < t_1 < \cdots < t_n < t_{n+1}$ be real numbers. For each $0 \leq k \leq n$, let f_k be a cubic polynomial defined on the interval $[t_k, t_{k+1}]$ such that the following condition holds for each $0 \leq k \leq (n-1)$ and $i = 0, 1, 2$:*

$$D^i f_k(t_{k+1}) = D^i f_{k+1}(t_{k+1}).$$

Then the collection f_k is called a **cubic spline** *on the interval $[t_0, t_{n+1}]$, the numbers $t_0, t_1, \ldots, t_{n+1}$ are called the* **knots**, *and the numbers $t_1, \ldots, t_n$ are called the* **interior knots** *of the cubic spline.*

In other words, a cubic spline is a piecewise cubic function which is continuous and has continuous first and second derivatives. It should be noted here that cubic splines have been defined differently by different authors and that some definitions (see, for example, [18]) describe larger classes of functions than that defined here.

A particular type of cubic spline, the natural cubic spline, can be defined as follows.

Definition 2 (Natural Cubic Spline) *Let $a < b$ be real numbers. A cubic spline f defined on the interval $[a, b]$ is a natural cubic spline if*

$$D^2 f(a) = D^3 f(a) = D^2 f(b) = D^3 f(b) = 0.$$

The conditions on the derivatives imply that the natural cubic spline on $[a, b]$ can be extended to a function on the real line which is linear outside of the interval $[a, b]$.

If $a \leq t_1 < t_2 < \cdots < t_n \leq b$ and $x_1, \cdots, x_n$ are all real numbers, then the natural cubic spline interpolant of the points $(t_1, x_1), \ldots, (t_n, x_n)$ on $[a, b]$ is simply the natural cubic spline on $[a, b]$ which passes through each of the (t_i, x_i). In practice, the numbers a and b are arbitrarily chosen to define the estimation interval. It can be shown ([28]) that the natural cubic spline interpolant defined on $[a, b]$ is the function which minimizes

$$\int_a^b (D^2 f(t))^2 dt. \tag{2.1}$$

among all interpolants which are differentiable and possess absolutely continuous derivatives on (a, b).

The curvature κ of the graph of a function $f(t)$ in the plane is defined (see [14, p.25]) as

$$\kappa(t) = [1 + Df(t)^2]^{-3/2} |D^2 f(t)|.$$

One way of interpreting the characterization in (2.1) is that the natural cubic spline interpolant is the interpolating curve which is the closest to a line. This is because for the non-negative expression (2.1) to equal zero, the curve $f(t)$ must have second derivative identically zero and hence must be linear on $[a, b]$. In fact, since $D^2 f(t)$ is a multiple of the curvature of f at t, the variational expression (2.1) is a measure of the total curvature of f on $[a, b]$. Therefore the natural cubic spline on $[t_1, t_n]$ which interpolates $(t_1, x_1), \ldots, (t_n, x_n)$ while minimizing (2.1) can be considered to be the "least curved" of all functions passing through the interpolated points. For this reason, given the assumption that a discrete time series is a record of a smooth function, the natural cubic spline interpolant can be thought of as an optimal estimate of that function. The linearity of the natural cubic spline outside of $[a, b]$ can be thought of as arising from the fact that the data contain no information on curvature outside of that interval.

With observed data, one can usually assume that there is some error in the measurement process by which the data was obtained. In this case, one might well prefer a smoother estimate of the observed function which comes 'close' to the observed data points to a rougher (more wiggly) estimate which interpolates them exactly.

The sum of squared prediction errors,

$$\sum_{i=1}^{n}[f(t_i) - x_i]^2, \tag{2.2}$$

is a good measure of how close an estimate $f(t)$ comes to exactly interpolating the data. Therefore the best estimate to a function observed in the presence of measurement error should, in some way, simultaneously minimize the two criteria (2.1) and (2.2). One way of specifying this requirement mathematically is to choose a positive real number λ and then define the optimal estimate $\hat{f}$ to be the function which minimizes the compound criterion

$$\sum_{i=1}^{n}[f(t_i) - x_i]^2 + \lambda \int_a^b (D^2 f(t))^2 dt. \tag{2.3}$$

It can be shown that the function which minimizes (2.3), called a **cubic smoothing spline**, is a natural cubic spline with interior knots at $t_1, \ldots, t_n$ ([28, p.107]). The constant λ, called the smoothing parameter, controls the degree of trade-off between the two optimality criteria. At one extreme, as λ tends to infinity, the estimate tends to the regression line fit to the data. The regression line is the smoothest possible estimate but is unlikely to fit the data very well. As λ tends to zero, on the other hand, the estimate will tend to the natural cubic spline interpolant. Therefore a greater degree of confidence in the accuracy of the data would lead one to choose a smaller value of λ while, a belief that the data is rather noisy would suggest a larger value of λ.

One method for choosing the smoothing parameter, called **cross validation**, attempts to choose the parameter which leads to the lowest mean squared prediction error for the data. The idea behind this method is that if one point in the observed time series were left out and the function were estimated from the remaining data, a good estimate should predict the deleted observation fairly accurately. Suppose, then, that a smoothing parameter λ has been chosen and that the point (t_i, x_i) has been left out of the observed time series. Define $f_i(\lambda, t)$ to be the cubic smoothing spline computed from the remaining data. If the prediction error is defined as $e_i(\lambda) =$

$f_i(\lambda, t_i) - x_i$, then the observed mean square prediction error associated with the given choice of λ is given by the expression

$$\mathrm{MSPE}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} e_i(\lambda)^2. \tag{2.4}$$

The cross-validation smoothing parameter is the value of λ for which $\mathrm{MSPE}(\lambda)$ is a minimum. Although the function $\mathrm{MSPE}(\lambda)$ is unknown, its minimum can be estimated fairly accurately using an iterative technique.

2.3 L-Spline Smoothing

The second term in expression (2.3), which penalizes the curvature of the estimate f by forcing its squared second derivative to be small, is often called a *roughness penalty*. Another way of describing the role of the roughness penalty is to say that the expression inside the integral defines the class of perfectly smooth functions. If smoothness is defined by relatively little curvature, then the only functions which are perfectly smooth are lines. The set of linear functions can be mathematically defined as the set of functions $f(t)$ which satisfy the differential equation $D^2 f = 0$ or, equivalently, as the null-space of the operator D^2. Viewed in this manner, with the roughness penalty seen as a linear differential operator which defines smoothness, the natural cubic spline interpolant can be considered to be a special case of a more general class of smoothing splines. This class has been called the class of L-spline smoothing functions ([46]) and is defined as follows.

Definition 3 (L-spline smoothing functions) *Let $(t_1, x_1), \ldots, (t_n, x_n)$ be n observed points in $\mathbb{R}^2$ with the t_i in nondecreasing order. Let a and b be real numbers with $a < t_1$ and $t_n < b$. Let L be a linear differential operator of order m and let λ be a positive real number. Then the* **L-spline smoothing function** *defined by λ and the (t_i, x_i) on the interval $[a, b]$ is the unique function f which minimizes the* *expression*

$$\sum_{i=1}^{n}[f(t_i) - x_i]^2 + \lambda \int_a^b [Lf(t)]^2 dt. \tag{2.5}$$

In this definition, the operator L is chosen to embody any prior beliefs about the function being estimated and λ controls the degree with which the estimate is smoothed towards the null-space of L. For example, if one had reason to expect the observed function to be sinusoidal in nature, one could penalize the linear differential operator $L = D^2 + \omega^2 I$, where I is the identity operator and ω is a parameter to be estimated. Perhaps the most common use of L-splines is as a means of obtaining function estimates with smooth derivatives. This can be achieved through penalizing the operator $L = D^m$, with m greater than two.

Suppose, for example, that one has a discrete observation of a function $f(t)$ and that one is interested in studying $D^d f(t)$, the dth derivative of $f(t)$. A natural strategy would be to construct an estimate of f and then to study the dth derivative of the estimate. The cubic smoothing spline estimate is unsuitable to this approach because its piecewise cubic nature implies that its derivatives will not be smooth. By penalizing $L = D^{d+2}$ in (2.5), however, one can minimize the integral of the squared second derivative of $D^d f(t)$ and can therefore obtain an estimate $f(t)$ whose dth derivative is smooth in the sense of a cubic smoothing spline.

The L-spline smoothing problem, the problem of finding the function which minimizes (2.5), is solved explicitly using the Green's function associated with the linear differential operator L. It can be shown [23] that for the problem defined in (2.5), there exists a bivariable function G_L such that

$$\int_a^b G_L(t, w) Lf(w) dw = f(t) \tag{2.6}$$

for all functions f such that

$$f(a) = Df(a) = \cdots = D^{m-1} f(a) = 0. \tag{2.7}$$

Recall here that m is the order of L, or the order of the highest derivative occurring in the definition of L. In other words, the Green's function inverts the action of L.

Of course, L does not have a unique inverse, since if $Lf = g$ and h is in the null-space of L, then $L(f + h) = g$ as well. However, given any measurable function g, there is a unique function f which satisfies the boundary conditions in (2.7) and also satisfies $Lf = g$. What the equation in (2.6) says is that that function f is given by the expression

$$f(x) = \int_a^b G_L(x, w)g(w)dw.$$

The Green's function can be constructed from another bivariable function, the so-called *reproducing kernel* associated with the linear differential operator L and the boundary conditions (2.7). The reproducing kernel is the bivariable function K_L which satisfies

$$\int_a^b L \circ K_L(t, s)Lf(s)ds = f(t)$$

for all functions f which satisfy the boundary conditions in (2.7). The importance of the with no deformation reproducing kernel lies in the following theorem. Recall that the values $t_1, \ldots, t_n$ are the n distinct values at which the data were observed.

Theorem 2.1 (Optimal basis theorem) *Let L be defined as in the definition of the L-spline smoothing function and let $u_1, \ldots, u_m$ be a basis for the null-space of L. Then the function f which minimizes (2.5) can be uniquely expressed as*

$$f(x) = \sum_{i=1}^{m} c_i u_i(x) + \sum_{j=1}^{n} d_j K_L(t_j, x).$$

A discussion of the mathematical properties of L-splines can be found in [46, ch.10], and the potential use of L-splines in statistics, together with an efficient ($\mathcal{O}(n)$) algorithm for computing them, is presented in [23]. This article also discusses applications of L-spline smoothing to statistics, along with several examples. The algorithm for computing an L-spline involves computing a basis for the null-space of the operator L and computing the reproducing kernel, both of which can be done numerically.

2.4 Functional Registration

2.4.1 Landmarks

In this thesis, a univariable **landmark** is a feature or event in an observable function which can be localized in time (recall that we are assuming that the independent variable represents time). The event itself is a feature of the process being observed and the landmark is a localizable feature of the function which is measured. Since they must be unambiguously identifiable, landmarks usually correspond to maxima, minima, or zero crossings in the function or in one of its derivatives. As was outlined in the introduction, registration is the process whereby the time components in a sample of functional observations are transformed in such a way as to make landmarks match up across the different observations.

This process is well illustrated by the hypothetical example of the vertical position of the knee as a function of time while walking normally. Such data would clearly be cyclic and one might very well be interested in finding the average, or mean, cycle. To make the example as simple as possible, imagine that the data is a record of the vertical position of the left knee for one individual taken over 20 consecutive gait cycles and, furthermore, that this has already been transformed into a continuous function.

The first task is to define what is meant by one gait cycle. Since the time taken to complete a cycle can be expected to vary somewhat from cycle to cycle, it is not enough simply to divide the total time into 20 equal intervals. One can state, however, that during one gait cycle the left leg must rise and fall once and, therefore, that each cycle should be characterized by exactly one local maximum in the data. This local maximum is a landmark which can easily be located and can therefore be used to define the beginning of each cycle. Thus the sequence can be broken up into 20 observations of individual cycles, each of which begins and ends at a maximum value.

Now each of the observations can be rescaled to a common length by multiplying the time coordinate by some scaling constant. In addition to the local maximum, one can now observe that the middle part of each observation is marked by a section where the value of the observation is fairly constant. This corresponds to the part of the gait cycle where the right leg is in motion and the left foot is planted, motionless, on the ground. Assuming that one has a smooth estimate of the second derivative of each function, one should then notice a clear local minimum at the beginning of this period of motionlessness, corresponding to the sharp deceleration when the left foot strikes the ground. Due to the natural variation in one's gait, however, the landmarks defined by these peaks in deceleration will not occur at the same locations in each observation. In order to line them up, it is necessary to compute a registration function for each observation.

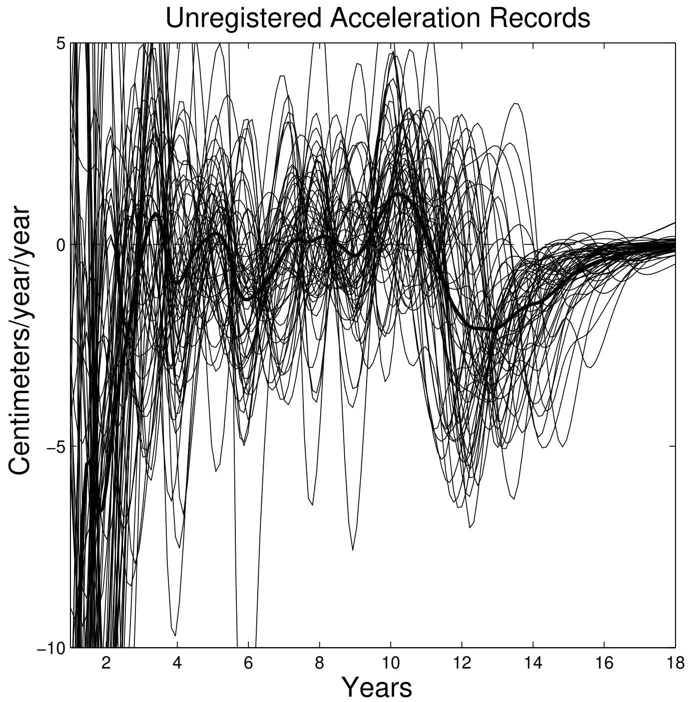

Figure 2.1: Female height measurements from the Berkeley growth study: estimated acceleration curves.

Figures 2.1, 2.2 and 2.3 illustrate some unregistered and registered functional data, along with the effect of registration on the mean. The data is the height of

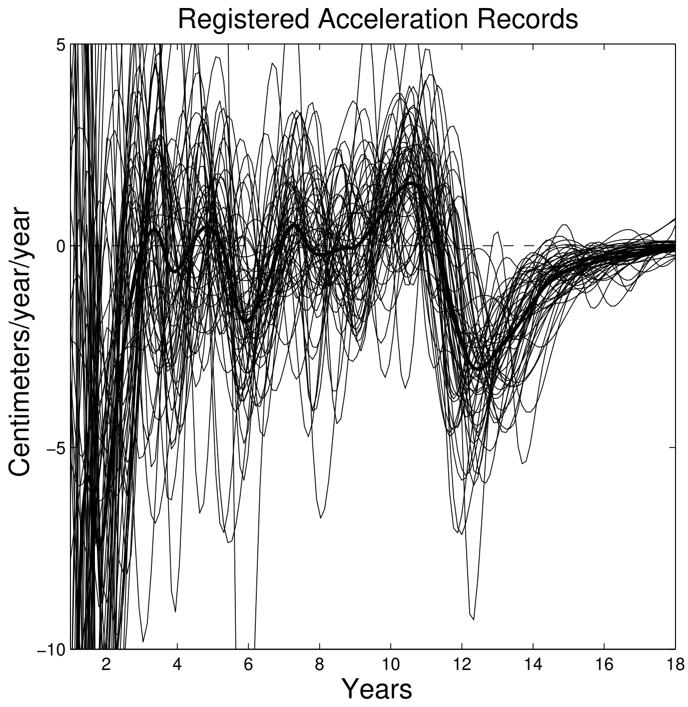

Figure 2.2: Female height measurements from the Berkeley growth study: registered acceleration curves.

54 female children, measured 31 times between the ages of one and eighteen. The figures themselves were generously provided by Dr. J.O. Ramsay. All 54 estimated acceleration curves are displayed in figure 2.1. The registered acceleration curves are displayed in figure 2.2. Figure 2.3 shows the means of both the registered and unregistered curves.

The height of a female child is typically characterized by a large deceleration in early childhood (following the rapid growth as an infant), by three childhood growth spurts, and by a very marked growth spurt occurring at puberty. Growth spurts are visible in the acceleration curve as periods of positive acceleration followed by negative acceleration. The presence of the pubertal growth spurt has been well known for as long as there have been children, but the childhood growth spurts were discovered through functional data analysis.

One can see some important differences between the means of the registered and

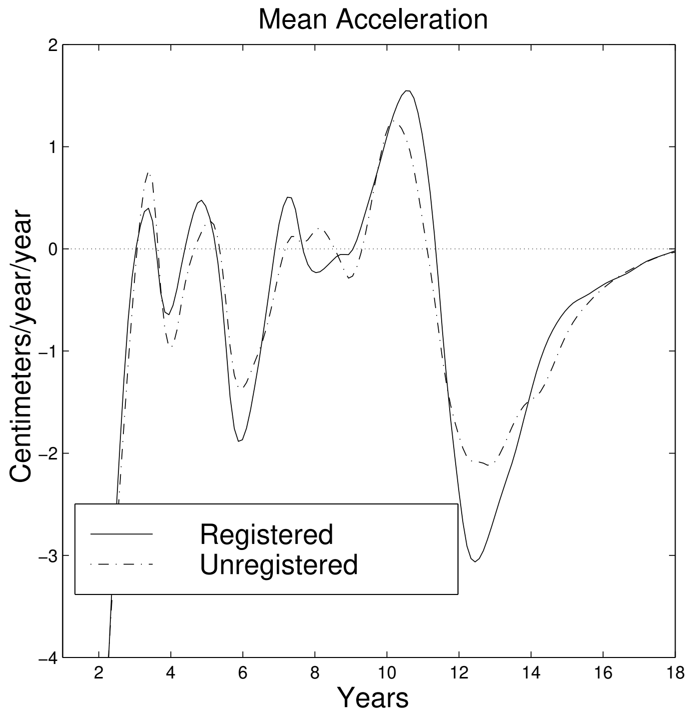

Figure 2.3: Female height acceleration: registered and unregistered mean curves.

unregistered data in figure 2.3. For example, the magnitude of the pubertal growth spurt and of two of the three childhood growth spurts are too small in the unregistered mean. Another difference can be seen in the part of the curve which rises from the period of greatest deceleration towards zero, following the pubertal growth spurt; this part of the curve is rather rough in the unregistered mean curve.

2.4.2 Registration Functions

If one were to estimate the mean gait function by simply averaging all of the (properly rescaled) observations, the fact that the deceleration landmarks occur at different times would result in the average second derivative function having a local minimum which was not as low, on average, as those observed in the individual observations. To avoid such a clearly unsatisfactory estimated mean function, it is necessary to warp each function so that the landmarks all occur at the same location. This warping, or registration, is typically done using a **registration function** computed separately for each observation.

Suppose that the average time at which the peak deceleration occurs is t_{min} and that each (rescaled) observation is defined on the interval $[0, T]$. Suppose further that in the i^{th} observed function, $f_i(t)$, the peak occurs at time t_i. Then a registration function for the i^{th} observed function is a smooth, strictly monotonic function $h_i(t)$ which maps the interval $[0, T]$ onto itself and satisfies $h_i(t_{min}) = t_i$. This registration function is used to yield the registered i^{th} function $f_i[h_i(t)]$ which, while still defined on the interval $[0, T]$, now has the property that the deceleration landmark occurs at t_{min}. The condition of monotonicity is necessary to ensure that the order in which different events occur in time is not reversed. The condition of monotonicity also guarantees invertibility of the registration function.

A more general mathematical definition of registration functions must assume an arbitrary number of landmarks and must map each of those landmarks to a specified target. While it is not strictly necessary, it is usually required that a registration function be smooth. Smoothness is certainly desirable if one wishes to work with derivatives of the registered functions.

Definition 4 (Registration function) *Let f_1 and f_2 be two realizations of a function defined on the interval $[0, T]$. Let $\{l_1, \ldots, l_m\}$ be a set of landmarks such that l_i has been observed to occur at time t_{i1} on f_1 and at time t_{i2} on f_2, for $1 \leq i \leq m$. A* registration function *for registering f_1 onto f_2 is a strictly monotonic function $h(t)$ which satisfies*

1. *$h(0) = 0$,*

2. *$h(T) = T$,*

3. *$h(t_{i2}) = t_{i1}$ for $1 \leq i \leq m$.*

The function $f_1 \circ h$ is thought of as f_1 after being registered onto f_2. Note that the landmark l_j occurs at t_{j2} in $f_1 \circ h$ because

$$(f_1 \circ h)(t_{j2}) = f_1(h(t_{j2})) = f_1(t_{j1}).$$

In the notation of this definition, one way to produce a registration function is by using a cubic spline to interpolate the $m + 2$ points $(0, 0)$, (T, T), and (t_{i2}, t_{i1}). Although it is not guaranteed that such a construction will yield a monotonic function, in practice it almost always does since the observed pairs of landmarks observations will themselves be monotonically increasing. Ramsay [43] has also used L-splines and a linear differential operator derived from the condition of monotonicity to formulate a more complicated algorithm which is guaranteed to produce monotone estimates.

2.5 Bivariable Splines and Function Estimation

In the univariable setting, spline functions have been used to estimate continuous functions from discrete data and also to produce registration functions [44]. In the bivariable setting, however, the estimation of functions is made more difficult by the fact that bivariable spline functions are less versatile than univariable splines. In addition, the problem of landmark registration is made more difficult by the fact that landmarks can be one-dimensional. This differs from the case of landmarks on curves which, being points, are zero-dimensional. Thus, the increased complexity of the function estimation and registration problems in two dimensions is due both to the nature of the available bivariable estimating functions and to the nature of the bivariable data itself.

One method for estimating functions from discrete bivariable data is the bicubic spline. This generates functions of the form $f(x, y) = f_1(x) \cdot f_2(y)$, where $f_1(x)$ and $f_2(y)$ are each univariable cubic splines. In other words, a bicubic spline is the product of two univariable splines. This approach, which can be used to create a bivariable functional estimator from any univariable estimator, has two distinct disadvantages from the point of view of functional data analysis. Firstly, it requires that the data be given on a regular grid and, secondly, it does not allow one to exert any control over the degree of curvature in the cross-product terms. The text *Curve and Surface Fitting with Splines*, by Dierkcx [19], is a useful reference on the fitting of tensor-product splines.

Bivariable data which comes in the form of an image is necessarily present on a regular pixel grid but there are many interesting sources of data which are not of this form. Geological and meteorological data, especially, tend to be collected at locations which are selected for convenience of measurement. This type of data will be given at scattered, rather than regular, locations and hence will be unsuitable to treatment by tensor product methods. In addition, even if the original data is given on a regular grid, the landmarks used to calculate registration functions will be unlikely to occur on a regular grid. In certain situations, therefore, tensor product methods of function estimation may be unsuitable for registering bivariable data.

Kernel smoothing is another method for estimating bivariable functions from discrete samples. In kernel smoothing, the value of a function at a point is estimated as a linear combination of all of the observed values, weighted by their closeness to the point at which the function is being estimated. For a set of n observations $\{f(x_1, y_1), \ldots, f(x_n, y_n)\}$ of a function f, a kernel smoothing estimate $\widehat{f}$ of f takes the form

$$\widehat{f}(x, y) = \sum_{i=1}^{n} f(x_i, y_i) K_h((x_i, y_i) - (x, y)),$$

where the function K_h is called the kernel. The kernel takes on larger values when (x_i, y_i) is close to (x, y) and smaller values, often zero, when (x_i, y_i) is further away from (x, y).

Typically, a bivariable kernel function K is chosen such that $K(u, v)$ is either zero or else decays rapidly for $||(u, v)|| > 1$ and K_h is defined by

$$K_h(u, v) = K\left(\frac{(u, v)}{h}\right).$$

In this case the parameter h is called the bandwidth and is used to control how many observations contribute to the value of $\widehat{f}(x, y)$. Smaller values of h lead to the value

$\widehat{f}(x, y)$ depending on observations in a smaller neighbourhood of (x, y) and hence to a less smooth estimate of $\widehat{f}$. The monograph *Kernel Smoothing*, by Wand and Jones [49], contains a detailed description of the theory involved in both univariable and bivariable kernel smoothing.

While kernel smoothing can often perform quite well in the interior of an area in which the data occurs, it becomes unstable near the boundary of such an area and this instability worsens as the bandwidth becomes larger [44]. Since excessively small bandwidths will often result in undersmoothing, the instability of kernel smoothing near the boundary can make this method unsuitable for estimating bivariable functions from functional data.

The thin-plate spline is another popular method for constructing bivariable functions from discrete data and has been used fairly extensively in the field of medical image analysis for bivariable registration and estimation. A partial list of references to the use of the thin-plate spline in medical image registration can be found in [29]. One way in which to characterize a thin-plate spline interpolating surface is as the function which minimizes the bending energy of an infinitely thin steel plate of infinite size which has been deformed so as to pass through a chosen set of points. This bending energy is proportional to the expression

$$\int_{\mathbb{R}^2} (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2)dxdy. \tag{2.8}$$

The notation f_{xx}, f_{xy}, and f_{yy} in this expression refers to the three second partial derivatives of f. This notation will be used again later in the thesis.

Given n observed points $\{z(x_1, y_1), \ldots, z(x_n, y_n)\}$, the thin-plate spline interpolant is the function f passing through the observed points which minimizes the expression in (2.8).

The original reasoning behind defining the cubic spline to be an optimal estimator for smooth functions arose from a tool used by draftsmen for drawing smooth curves passing through prescribed points. This tool, called a spline, was a long, flexible strip of wood which, when bent to pass through a set of fixed points, would elastically conform to a smooth curve. This curve would take on the shape which minimized the bending energy of the wooden spline. Since that bending energy happened to be proportional to the square of the second derivative, the cubic spline interpolant was shown to be mathematically equivalent to the mechanical interpolant which was already in use by draftsmen the world over. Because physical characterization of the thin-plate spline is analogous to that of the natural cubic spline, the thin-plate spline is often presented as a two-dimensional analogue to the cubic spline and is used in similar contexts.

In the same manner as the cubic spline interpolant, the thin-plate spline interpolant can be generalized to define the thin-plate smoothing spline as the unique function f which minimizes, for some positive parameter λ, the expression

$$\sum_{i=1}^{n} \left(f(x_i, y_i) - z(x_i, y_i)\right)^2 + \lambda \int_{\mathbb{R}^2} (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2)\, dx dy. \tag{2.9}$$

As in the case of the cubic spline smoothing function, the first term in this expression measures fidelity to the data and will be equal to zero if f interpolates the data exactly. The second term, on the other hand, measures smoothness and will be equal to zero if f is a plane, corresponding to a thin steel sheet with no deformation. The parameter λ controls the trade-off between the desire for fidelity to the data and the desire for smoothness, with larger values producing smoother, but less faithful, estimates.

Besides the intuitively attractive form of its analogy to the cubic spline, the primary reason for the popularity of the thin-plate spline for function estimation is the fact that there exists an efficient algorithm for its computation which is defined for any arbitrary configuration of data [47]. It can be shown [21] that the general form of the thin-plate spline is

$$f(x, y) = \alpha_0 + \alpha_1 x + \alpha_2 y + \sum_{i=1}^{t} \beta_i \Phi(\|(x, y) - (x_i, y_i)\|),$$

where the univariable function Φ is defined by $\Phi(r) = r^2 \ln(r^2)$. Although the function Φ is not defined at the origin, it can be extended by continuity to have value $\Phi(0) = 0$.

Thus, every thin-plate spline is the direct sum of a linear function and a finite number of translations of the radially symmetric function Φ. The function Φ is an example of a radial basis function and thin-plate spline smoothing is a special case of a method known as radial basis function smoothing. Other examples of radial basis functions are the (reciprocal) multiquadric functions $\Phi(r) = (r^2 + c^2)^{\pm 1/2}$ and the Gaussian functions $\Phi(r) = e^{-cr^2}$, both of which depend on on the parameter c ([48]). When such a method is used to estimate a function from a set of t data points, the resulting estimate is the direct sum of a plane and t radial basis functions, one centered at each data point.

As an example of bivariable data which might need to be registered, consider a frontal image of a human face. To simplify the problem, assume that the head has been completely shaved. Although the size, shape, and fine structure of the face varies considerably across individuals, as is evidenced by the fact that we are readily able to recognize different people from their faces alone, the overall features are constant and one can easily identify a large number of landmarks. Although some of these landmarks may be single points, as is the case in univariable data, many of the landmarks are one-dimensional curves. For example, if one wanted to register a sample of faces with the goal of obtaining a recognizable mean face, one would certainly want to warp the entire boundary of each face to the same location. Similarly, if the line where the lips meet was not registered properly along its full length, the mean face might not appear to have two separate lips. This presence of one-dimensional landmarks is a fundamental difference between univariable and bivariable functional registration.

A serious disadvantage of the original algorithm for computing the thin-plate spline is that its estimates tend to become ill-conditioned in the presence of data points which are too close together. This weakness creates serious problems in estimating registration functions from one-dimensional landmarks. The reason for this is that to estimate a smooth function possessing a prescribed value along a curve, it is necessary to interpolate or smooth a fairly dense set of points along the entire curve. These points are necessarily close together and hence any attempt to use the thin-plate spline to estimate a function from one-dimensional landmarks may require inverting the action of an ill-conditioned, or almost singular, matrix. Such an operation can be either computationally infeasible or numerically unstable.

A discussion of the ill-conditioning problem can be found in [47]. The same paper presents an efficient pre-conditioning technique which leads to a well-conditioned algorithm which can smooth large numbers of closely located points. Using this modified algorithm, one should be able to use the thin-plate spline to register one-dimensional landmarks. To the best of my knowledge, however, no attempt to do so has appeared in the literature.

Chapter 3

Image Registration Methodology: a Literature Review

3.1 Different Goals, Different Methods

The problem of image registration occurs in fields as diverse as those of computer vision, remote sensing, and medical imaging. Many medical image registration methods are designed to register trivariable functions (three-dimensional images, especially of the brain) but these can generally also be adapted to bivariable data. In any field, the image registration problem can be stated as that of smoothly deforming one image, called the *source* image so that it looks like, or matches, another image, called the *target* image. Although this might seem to imply that there are a large number of different solutions already existing for the registration of images, the solutions are often suitable only to very specific registration contexts.

In the field of computer vision, for example, the problem of image registration arises in the context of image recognition. The problem there is to match an observed image to a pre-existing template. The computer will have a number of templates of images which it recognizes and will try to decide which of its templates is 'closest' to the observed image. In this context, one image is 'close to' another if it can be warped to match that image. The warps are usually constructed as discrete maps of one pixel grid to another, however, and make no assumptions about the smoothness of the underlying function. A functional registration function must be a smooth map between two continuous subsets of $\mathbb{R}^2$ rather than between two functions on discrete grids.

In general, methods of image registration can be divided into landmark methods and global methods. Unlike landmark registration, where the domain deformation attempts to match up certain corresponding structures in as smooth a manner as possible, global registration attempts to match up the entire image in as smooth a manner as possible. That is, instead of measuring the quality of the match by the difference in location of certain specified landmarks, the match is measured by some measure of closeness which involves the entire domain of the function. This measure is often as simple as minimizing the squared or absolute difference in intensity, but can also be more complex. For an exhaustive list of the different closeness measures that have been used in the literature, see the article "A survey of medical image registration", by Maintz and Viergever [35] and the references therein. Two of the more common methods are maximization of the pixel-wise cross-correlation of the range and maximization of cross-correlation in the Fourier domain. It should be noted here that the term "global registration" has sometimes (for example, in [3]) been used to mean affine registration, or registration which results only from translation, rotation, and scaling. Two recent surveys of image registration methodology can be found in [29] and [35].

From the point of view of visually matching up two images so that they look as similar as possible, a global registration approach will usually do a better job than landmark registration. Global registration also has the advantage of not requiring the selection and identification of landmarks, a process which is both time-consuming and a potential source of error. Because the user is not required to pre-process the data in this way, global registration methods are often referred to as *automatic registration*. For these two reasons, namely an optimal visual match and the absence of a landmark selection step, global registration is more popular than landmark registration.

From the point of view of separating variation in range from variation in domain, however, global registration can often be inferior to landmark registration. The point of picking out landmarks is that one chooses points or curves in each image which one knows, a priori, correspond to each other. Often, if a source image is properly registered to a target image, the range value of a point in one image will be similar to the range value in the other but variation in range means that the two range values will not generally be identical. However, if the variation in domain has been removed, then the location of each identifiable feature common to the general object being depicted should be identical in each image. Matching up the range of each image, however, in no way guarantees that features themselves will match. In general, global registration can be expected to perform best in situations in which little range variation is expected.

As an extreme example of how matching up range values (global registration) could go wrong, one can imagine registering images of two eyes: one with a pale blue iris and the other with a relatively dark brown iris. In registering the pale eye to the darker one, it is quite possible that the range values corresponding to the dark brown iris might be closer to the the range values of the pupil, rather than to the pale blue iris, of the other eye. In such a case, many global registration methods would attempt to enlarge the pupil of the pale eye to cover the entire iris of the brown eye.

As another example, imagine registering two faces, each of which has a differently shaped birthmark on the right cheek. From the point of view of functional data analysis, these birthmarks should be considered to be random range variation rather than corresponding features which need to be registered. A global registration algorithm, however, might well incorrectly attempt to match up the birthmarks. In cases where range variation can be confounded with spatial, or domain, variation, good landmark registration can be expected to yield better results than global registration.

3.2 Affine Transformation Models

Satellite images of the earth and images obtained from other remote sensing data are often registered by applying affine transformations to the axes defining the domain. The transformation typically takes the form of translating, rotating, and stretching or compressing the image in such a way as to minimize some appropriate measure of image closeness. This type of registration is very effective at removing the effect of taking pictures of two-dimensional surfaces from different distances or angles and from locations which are moving with respect to the subject of the picture. Although affine registration is most frequently used in the processing of remote sensing imagery, it can be of value for the same reasons in the registration of any image data. The sensors used in obtaining the data may not always be placed in exactly the same location relative to the subject and this will result in differences in the images which correspond to affine transformations.

Affine transformations alone, however, are insufficient to model the variation present in images which would typically be the subject of functional data analysis, as that variation tends to be highly non-linear. It is not hard to see, for example, that the shape differences between the face of Don Knotts and that of Marilyn Monroe are due to more than just an affine transformation of space. For this reason, the affine transformation method of registration is useful mainly as a preprocessing tool in the registration of functional data. By aligning two images as closely as possible using a linear change of coordinates, one will not usually register them properly but one can ensure that the further non-linear distortion necessary for proper registration will be as small as possible.

Nonetheless, probably because it is the easiest method of registration available, affine methods have been and continue to be used extensively in the field of medical imaging, both for landmark and global methods. Affine registration methods are used to optimally align a set of landmarks in [6], [5], [39], [41], [30], [34], [37], and [33]; [50] and [24] use affine methods to maximize the cross-correlation between the registered and target images; [32] uses affine registration to maximize the mutual information of the two images. An interesting theoretical analysis of the properties of populations of shapes after modding out all changes of location, rotation, and size can be found in the book *Statistical Shape Analysis* by Dryden and Mardia [20].

3.3 Thin-Plate Spline Warps

In the field of medical imaging, at least, thin-plate spline registration is currently the most commonly used method for non-linear landmark registration. As was mentioned in section 2.5, the thin-plate spline function can be used to interpolate or to smooth a set of observations of a real-valued function f of the plane while minimizing bending energy corresponding to a thin metal plate of infinite dimension. If the observations are of the form of triples (x_i, y_i, z_i), for $1 \leq i \leq n$, with z_i being the observed value at the point (x_i, y_i) in the domain, then the thin-plate spline smoothing estimate $\hat{f}$ of the function $z(x, y)$ depends on a positive smoothing parameter λ and is the function which minimizes

$$\sum_{i=1}^{n} \left(f(x_i, y_i) - z_i\right)^2 + \lambda \int_{\mathbb{R}^2} (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2)\,dx\,dy.$$

The general representation of a bivariable registration function is as a vector-valued, bivariable deformation function of the form

$$\mathbf{h}(x, y) = [h_1(x, y), h_2(x, y)].$$

The domain Ω is deformed by the mapping $(x, y) \to (x, y) - \mathbf{h}(x, y)$. The problem of estimating the vector-valued registration function $\mathbf{h}$ can then reduced to that of estimating the real-valued bivariable functions h_1 and h_2, the deformation functions associated with the first and second coordinates of the plane. Bookstein ([9]) used thin-plate splines to register a sample of images of the human brain from 14 landmark points. More recent references to the use of the thin-plate spline for image registration include [7], [1], and [27].

The original algorithm for computing thin-plate spline functions tends to be poorly conditioned in the presence of landmarks which are closely spaced and which correspond to similar function values [47]. Unfortunately, this is exactly the type of situation which arises when registering curves embedded in bivariable data. A result of this ill-conditioning problem is that the original algorithm for computing thin-plate spline smoothing functions is ill-equipped to register one-dimensional, or curve, landmarks.

This problem has been greatly reduced, if not eliminated, by an efficient preconditioning technique proposed by Sibson and Stone in [47] and it is now possible to register large numbers of closely spaced points with thin-plate splines. In order to perfectly register an entire curve, one must register an infinite number of closely spaced points along the entire curve. In practice, registering a reasonably close mesh of points along the curve will produce acceptable results.

The result of not registering obvious one-dimensional landmarks is readily visible in Bookstein's example ([9]). Although the improvement of the registered mean over the mean of the unregistered images is marked, the boundary of the skull in the registered mean remains very fuzzy and ill-defined. This problem could be avoided by treating the boundary of the skull as a one-dimensional landmark and registering a reasonably dense set of points on the boundaries.

Another computational problem which arises when using the thin-plate spline to register large numbers of points is the fact that, in order to evaluate the deformation function $\mathbf{h}$ at an arbitrary point (x, y), it is necessary to evaluate the radial basis function

$$\Phi_i(x,y) = \|(x,y) - (x_i, y_i)\|^2 \ln\left(\|(x,y) - (x_i, y_i)\|^2\right)$$

for each data point (x_i, y_i) used to define the warp. If the deformation must be evaluated at a large number of points throughout the domain Ω, as is often the case, this can add up to a large computational load. Powell ([42]) has devised a method based on Laurent series for approximating the value of a thin-plate spline at a large number of points more quickly. For a situation in which the data points from which a thin-plate smoothing spline is being computed are the lattice points of a regular rectangular grid and one only needs to evaluate the spline on that grid, Buckley presents a very fast algorithm for computing the solution in [12].

3.4 Elastic Deformation Models

In elastic registration, the deformation of the image due to registration is modelled as an elastic sheet (or body, in the three-dimensional case) subject both to external forces, which drive the deformation, and internal forces, or strains, which impose smoothness. In other words, the image to be deformed is thought of as lying on an elastic rubber sheet which is being stretched by an external force in order to match up with the target image. The elastic nature of the sheet ensures that the deformation is smooth. If the deformation at a point $\mathbf{x}$ is described by the vector-valued function $\mathbf{u}(\mathbf{x})$, then the behaviour of such an elastic body is described by the Navier equation

$$\mu \nabla^2 \mathbf{u} + (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}) + \mathbf{f}(\mathbf{u}) = \mathbf{0}, \tag{3.1}$$

where $\mathbf{f}(\mathbf{u})$ describes the applied body forces which drive the registration. This vector equation describes the behaviour of both coordinate deformations simultaneously.

The quantities λ and μ are Lamé's elasticity constants and relate the applied forces per area (stresses) to the resulting deformations (strains). For the purpose of computing registration functions, λ is usually set to zero and μ determines the degree of smoothness of the resulting function $\mathbf{u}(\mathbf{x})$. Larger values of μ correspond to a stiffer elastic medium with stronger strain forces and result in smaller deformations but a smoother deformation function $\mathbf{u}$. Therefore, μ is somewhat analogous to the smoothing parameter in spline smoothing. An early example of the use of the elastic model in registration is [2], while a more recent one is [3]. The registration function itself is typically approximated by a finite element function.

The force function $\mathbf{f}(\mathbf{u})$ is designed to locally deform the image in such a way as to reduce some measure of distance between the source and target images. The purpose of $\mathbf{f}(\mathbf{u})$ is to bring similar regions into correspondence. In the work by Bajscy et. al. ([2], [4], [3]), the force function is related to the gradient of the function $S(\mathbf{u})$ which estimates the cross-correlation between the deformed source image and the target image. The force function then attempts to deform the source image in such a way as to maximize the correlation between the deformed and target images. Although other types of force functions could be used, I am not aware of any others occurring in the literature.

A global registration approach, elastic body registration attempts to optimize the overall match of the deformed image to the target without considering individual landmarks, a feature which makes it attractive in situations where landmarks are not present or are difficult to identify. In the presence of identifiable landmarks, however, this overall matching strategy is unlikely to match up the landmarks as well as would a landmark-based registration method.

Besides its inability to distinguish between domain and range variation, another disadvantage of the elastic registration process is that it tends to suppress large localized deformations. This effect is due to the stress/strain formulation of elastic solids whereby stresses increase in direct proportion to the size of the deformation. The net result is that highly localized deformations tend to be oversmoothed ([17]).

3.5 Viscous Fluid Models

Developed more recently by Christensen [16], viscous fluid registration models the registration function as a thick, flowing fluid. Although this physically based approach is similar in philosophy to the elastic deformation model, it has the advantage of easily accommodating the large localized deformations which tend to be suppressed by the elastic body model. Both elastic and viscous fluid methods of registration are most commonly applied to the registration of brain images and it seems intuitively more sensible to model the brain as a viscous fluid than an elastic solid. One important use of global registration in brain imaging is to register different images of the same brain to each other. Given the fluid nature of the brain, the shape of an individual's brain tends to change unpredictably over time and hence successive images of the brain, even if the position of the head is identical, will not match up. This type of movement on the part of the brain is likely to be better modelled by a viscous fluid than by an elastic body.

The viscous fluid model assumes that internal forces which resist deformation will relax over time. Thus the deformation is seen as a flow which occurs over time, rather than as an elastic stretching in which the resulting stresses are time-invariant. The model itself is governed by the Navier-Stokes fluid equation for $\mathbf{v}(\mathbf{x}, t)$, the velocity field associated with the deformation $\mathbf{u}(\mathbf{x}, t)$, where t represents time. The Navier-Stokes fluid equation is

$$\mu\nabla^2\mathbf{v} + (\lambda + \mu)\nabla(\nabla \cdot \mathbf{v}) + \mathbf{f}(\mathbf{u}) = \mathbf{0}. \tag{3.2}$$

Again, $\mathbf{f}(\mathbf{u})$ is the force function which drives the deformation flow. The velocity $\mathbf{v} = [v_1(\mathbf{x}, t), v_2(\mathbf{x}, t)]$ of an element of mass passing through $\mathbf{x}$ at time t is related to the deformation field $\mathbf{u}(\mathbf{x}, t)$ by the relation

$$\mathbf{v} = \frac{d\mathbf{u}}{dt} = \frac{\partial \mathbf{u}}{\partial t} + v_1\frac{\partial \mathbf{u}}{\partial x_1} + v_2\frac{\partial \mathbf{u}}{\partial x_2}$$

It is interesting to note that, except for the forcing function $\mathbf{f}(\mathbf{u})$, the viscous fluid equation (3.2) is nothing more than the elastic equation (3.1) applied to the velocity field rather than to the displacement field.

The velocity and displacement solutions $\mathbf{u}(\mathbf{x}, t)$ and $\mathbf{v}(\mathbf{x}, t)$ are calculated simultaneously as finite element functions at progressive increments of time t. The solutions are calculated on some closed region Ω enclosing the data with the boundary conditions that the deformation, and hence the velocity, always be zero on the boundary. This results in Ω being mapped to itself with the boundary unchanged. At each successive increment t, the force function $\mathbf{f}(\mathbf{u})$ evaluated at time $t-1$ is used to calculate the new deformation field at time t.

The force driving the deformation is based on a Bayesian formulation of the difference between the deformed and target images and this force is defined everywhere on the image being deformed. The Bayesian prior likelihood is defined as the integral over the domain Ω of the squared intensity difference between the deformed and target images and is a function both of the deformation field $\mathbf{u}$ and of time. The body force $\mathbf{f}$ is then the variation of this function taken with respect to the displacement. The flow is carried out at successive time intervals until the driving force or, equivalently, the difference between the registered and target image, falls below some threshold value.

As a global approach to image matching, this method shares with elastic registration the disadvantage of confounding variation in range with variation in domain. Another disadvantage is that fluid registration is prone to misregistering images by allowing small parts of the deforming image to grow outside of the target image. Although it often appears to register images very well, it can produce spectacular failures with small parts of the source image growing to cover large areas. Ironically, while a major failing of elastic registration is its inability to model large localized deformations, fluid registration reproduces localized deformations so well that it can result in misregistration. One method of reducing the probability of these misregistrations is to constrain the deformation by requiring that it exactly match up certain previously identified landmarks. This raises the possibility that perhaps, in the future, global and landmark methods of registration might be combined in order to incorporate the advantages of both.

Chapter 4

A Finite Element Bivariable L-Spline Function

4.1 A Bivariable Generalization of the L-Spline

Recall that the (univariable) L-spline smoothing function is constructed from a linear differential operator L chosen to measure smoothness and from a positive smoothing parameter λ. Given a set of observations (x_i, y_i), for $1 \leq i \leq n$, the L-spline smoothing function is then defined to be the unique real-valued function f which minimizes the quantity

$$\sum_{i=1}^{n}[f(x_i) - y_i]^2 + \lambda \int [Lf]^2 dx, \tag{4.1}$$

where the integral is evaluated over some interval which contains all of the x_i.

It is natural to ask if the L-spline can be generalized to multivariable functions. This chapter will examine the bivariable case. Except for the fact that the arguments of the solution function are now vector valued, the description of the bivariable L-spline smoothing problem is virtually identical to that in one dimension. Suppose that one has observed t values $\{z_1, \ldots, z_t\}$ of an unknown function at t different argument points $\{\mathbf{p}_1, \ldots, \mathbf{p}_t\}$ in the real plane. Then one would like to estimate a smooth function f of $\mathbb{R}^2$, where smoothness is defined by minimizing the square of some linear partial differential operator L applied to f, which comes close to the values z_i at each of the arguments $\mathbf{p}_i$. Rather than being a curve, the solution $\hat{f}$ will now be a surface which must be estimated over some region $\Omega \in \mathbb{R}^2$ containing all of the points $\mathbf{p}_i$. This function can be defined, in a manner similar to that in (4.1), by substituting $\mathbf{p}_i$ for x_i and z_i for y_i in the sum of squared error term: the solution is the function $\hat{f}$ which minimizes the quantity

$$\sum_{i=1}^{t} (f(\mathbf{p}_i) - z_i)^2 + \lambda \int_{\Omega} (Lf)^2 d\Omega. \tag{4.2}$$

There are two important differences between the univariable and the bivariable L-spline problems. Although there is a standard method for computing the Green's function associated with a linear differential operator, there is no general method for defining the Green's function of a linear partial differential operator. For this reason the univariable algorithm for computing an L-spline cannot be generalized to penalizing arbitrary linear partial differential operators in two or more dimensions. Nonetheless, if one were to find a linear partial differential operator whose Green's function was known and whose null-space was finite dimensional, the univariable algorithm for solving the L-spline smoothing problem could be adapted to more than one dimension.

The second difference between the univariable and bivariable smoothing problems is that there is often no obvious unique choice of axes for parametrizing bivariable functional data. In the univariable case, the argument usually represents time or distance or some other intrinsically meaningful quantity. Bivariable data, on the other hand, is often observed as a function of space and the axes must therefore be arbitrarily specified by the observer. This distinction is important because for a finite region Ω, if L is not chosen carefully, the quantity

$$\int_{\Omega} (Lf)^2 d\Omega$$

and thus solution to (4.2) can depend on the choice of axes used to parametrize f.

Since a reasonable definition of smoothness should not depend on the choice of axes, particularly when this choice is entirely arbitrary, the partial differential operators used to define smoothness must be restricted to those which do not depend on this choice. This means that L must be chosen from the class of linear partial differential operators which are invariant to both rotations and translations. In other words, if T is either a rotation or a translation, then L must satisfy $L(f \circ T) = (Lf) \circ T$ for all possible functions f. This class of operators is the ring generated by the Laplacian operator Δ, defined by $\Delta f = f_{xx} + f_{yy}$ for any $f \in C^2(\mathbb{R}^2)$. In other words, any partial differential operator L which is invariant to both translations and rotations can be expressed as

$$L = \sum_{i=0}^{p} a_i \Delta^i$$

for some non-negative integer p and constants a_i. This result is, in fact, true for partial differential equations in n variables for any $n > 1$ ([22, p.67]). Therefore the bivariable L-spline smoothing problem in its most general form can be expressed as that of finding the function f which minimizes

$$\sum_{i=1}^{n} (f(\mathbf{p}_i) - z_i)^2 + \lambda \int_{\Omega} \left(\sum_{i=0}^{p} a_i \Delta^i f \right)^2 d\Omega. \tag{4.3}$$

4.2 A Finite Dimensional Approximation

The method presented in this thesis will compute an estimate of the solution to the minimization problem (4.3) in a computationally convenient finite-dimensional space of functions called a finite element space. The presentation will initially be restricted to the case where $L = \Delta$; the problem to be solved will be that of finding the function $\hat{f}$ which minimizes

$$\sum_{i=1}^{n} (f(\mathbf{p}_i) - z_i)^2 + \lambda \int_{\Omega} (\Delta f)^2 \, d\Omega. \tag{4.4}$$

Later, this solution will be generalized to the case where L is a general polynomial in Δ. First, a variational formulation of the minimization criterion (4.4) will be derived and the domain Ω over which f is to be estimated will be triangulated. The next step defines a piecewise quadratic function over the triangulation which satisfies the variational form. Finally, the solution will be generalized to accommodate the situation in (4.3) where $L = \sum a_i \Delta^i$.

A full exposition of the solution method requires the presentation of a certain amount of background theory, mainly regarding the definitions and properties of certain function spaces and some of the theory behind the finite element method. Although the problem being discussed here is one of estimating a function on $\mathbb{R}^2$, the method used is readily generalizable to the estimation of functions of $\mathbb{R}^3$. No method which penalizes a second order partial differential operator can be generalized to functions of $\mathbb{R}^n$ because for $n > 3$ the space in which the problem is defined (H^2, to be defined later) contains discontinuous functions and hence point evaluation is not well-defined. For more details on this restriction, the reader can refer to [11, Thm.1.4.6].

Since the method can be generalized to $\mathbb{R}^3$, whenever possible the definitions given will be phrased in terms on $\mathbb{R}^n$. First of all, the function spaces used in the construction and analysis of the solution method will be presented and discussed. Following this, a variational formulation of the minimization criterion (4.3) will be derived. Then the elements of finite element theory necessary to define the solution space will be outlined, followed by the method of discretizing the variational form of (4.3) into the matrix expression which will be solved to yield the solution. Finally, the method will be generalized to $L = \sum a_i \Delta^i$.

4.3 Partial Derivatives and the H^m Function Spaces

The most important function space used in the finite element method is a specific variety of Hilbert space called a *Sobolev space*. For more information about the theory of Sobolev spaces, the reader may consult [10] or [11]. Sobolev spaces are defined in terms of the existence and smoothness of certain generalized, or weak, partial derivatives. For the sake of completeness, both the usual partial derivative and the weak partial derivative will be defined here. Let the multi-index $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ be an m-tuple of non-negative integers and let $|\alpha| = \sum_{i=1}^{m} \alpha_i$. Define the partial derivative operator D^α by

$$D^\alpha f = \frac{\partial^{|\alpha|}}{\partial^{\alpha_1} x_1 \cdots \partial^{\alpha_m} x_m} f$$

for any function $f : \mathbb{R}^m \to \mathbb{R}$.

Definition 5 (Weak derivative) *For a domain $\Omega \in \mathbb{R}^m$, let f be a real-valued function in $\mathcal{L}^1(\Omega)$ and let α be an m-tuple of non-negative integers. We say that f has a* **weak derivative** *$f^{(\alpha)}$ if there exists a function $g \in \mathcal{L}^1(\Omega)$ such that*

$$\int_\Omega g(x)\phi(x)dx = (-1)^{|\alpha|} \int_\Omega f(x) D^\alpha \phi(x) dx \quad \forall \phi \in C_0^\infty(\Omega),$$

where $C_0^\infty(\Omega)$ denotes the space of infinitely differentiable functions which vanish on the boundary of Ω. When such a function exists, we define $f^{(\alpha)} = g$.

Note that if $f \in C^{|\alpha|}$, then the weak derivative $f^{(\alpha)}$ is identical to the usual partial derivative $D^\alpha f$ and that if this derivative exists then the two notions of derivative are equivalent. The weak derivative is used as follows to define the Sobolev spaces $H^m(\Omega)$, together with their norms.

Definition 6 ($\mathbf{H}^m$) *For a non-negative integer m and a domain $\Omega \in \mathbb{R}^n$, the Sobolev space $\mathbf{H}^m(\Omega)$ is defined to be the space*

$$H^m(\Omega) := \quad \Big\{ f : \Omega \to \mathbb{R} \, \Big| \, \int_\Omega (f^{(\alpha)})^2 < \infty \quad \forall |\alpha| \le m \Big\}.$$

The subspace of functions in $H^m(\Omega)$ which have normal derivative equal to zero on the boundary of Ω will be referred to as $H_0^m(\Omega)$. The H^m spaces are important in the smoothing problem because, by definition, some set of partial derivatives of f must exist and be square integrable for the penalty functional to be well-defined. The H^m spaces are therefore the natural ones in which to search for solutions to the smoothing problem. It should perhaps be noted here that the H^m spaces are a special case of the more general class of Sobolev spaces.

4.4 Finite Element Spaces of Functions

The finite element method is a general method for approximating the solutions to partial differential equations (PDEs). In its simplest form, it involves generating a variational formulation of the PDE to be solved and then defining a function in a piecewise manner which satisfies the variational form in some convenient, but restricted, function space defined over the domain of the problem. The finite element method has become a standard tool for the engineer and there are numerous texts describing the procedures of defining the discrete, variational simplification of the problem to be solved and of computing the solutions. While any one of these texts will serve as an adequate reference for the material in this thesis, a particularly readable source is a series of six texts by J. Tinsley Oden, written in collaboration with Graham F. Carey, and Erin B. Becker. Volumes I [8], II [13], and IV [40], in particular, contain virtually all of the background needed in a form which is easy to understand and well illustrated with thoughtfully selected examples.

A finite element space of functions is a finite-dimensional subspace of some Sobolev space which is constructed in a piecewise manner so as to facilitate computation while providing a good estimate of the larger class of functions being approximated. This can be made both more precise and more general as follows:

Definition 7 (Triangulation) *A subdivision of a domain Ω is a finite collection of open sets $\{\mathcal{K}_i\}$ such that*

- $\mathcal{K}_i \cap \mathcal{K}_j = \emptyset \quad if \quad i \neq j$ *and*
- $\bigcup \bar{\mathcal{K}}_i = \bar{\Omega}$.

A triangulation of a domain $\Omega \in \mathbb{R}^2$ is a subdivision consisting of triangles such that no vertex of any triangle lies on the interior of an edge of any other triangle.

The notation $\bar{\mathcal{K}}_i$ and $\bar{\Omega}$ in the above definition denote the closure of the sets $\mathcal{K}_i$ and Ω, respectively. Although the open sets $\mathcal{K}_i$ in a subdivision can be arbitrary, in practice they are usually polytopes or else polynomial transformations of polytopes. Each of these open sets is used to define a finite element, or a bounded subset of $\mathbb{R}^n$ along with a finite-dimensional space of functions defined on that subset. A finite element is defined in complete generality as follows (recall that the space of continuous, linear functionals on a function space $\mathcal{P}$ is denoted $\mathcal{P}'$).

Definition 8 (Finite element) *A* **finite element** *is a triple* $(\mathcal{K}, \mathcal{P}, \mathcal{N})$ *where*

- $\mathcal{K} \subset \mathbb{R}^n$ *is a domain with a piecewise smooth boundary (the* **element domain***),*
- $\mathcal{P}$ *is a finite dimensional space of functions defined on* $\mathcal{K}$ *(the* **shape functions***),*
- $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_d\}$ *is a basis for* $\mathcal{P}'$ *(the* **nodal variables***).*

The context in which these definitions are put into practice is that of defining functions on a domain $\Omega \subset \mathbb{R}^n$. A subdivision of Ω is constructed and each element $\mathcal{K}_i$ of that subdivision is made into the domain of a finite element $(\mathcal{K}_i, \mathcal{P}_i, \mathcal{N}_i)$. Then a separate function is defined on each element of the subdivision as a linear combination of its shape functions. In other words, the functions on each $\mathcal{K}_i$ are chosen from among the elements of the finite dimensional function space $\mathcal{P}_i$. These functions are constructed in such a way as to satisfy certain continuity conditions on the boundaries of their domains and then the resulting collection of finite element functions defines a piecewise function on all of Ω.

In fact, the univariable polynomial spline functions can be seen to be finite element functions. If the knots $x_1 \cdots x_n$ are ordered so that $x_1 < x_2 < \cdots < x_n$, then the domain Ω of the spline function can be considered to be either the whole real line or else the interval $[x_1, x_n]$. In the first case, the subdivision is the collection of intervals $\{(-\infty, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, \infty)\}$ and in the second case, the subdivision is the collection $\{(x_1, x_2), \ldots, (x_{n-1}, x_n)\}$. For a cubic spline, the function space $\mathcal{P}$ on a given interval in the subdivision is defined to be the polynomials of degree at most three and the continuity conditions are that the functions on adjacent intervals, along with their first and second derivatives, be continuous at their common endpoint.

Since this thesis is concerned only with domains in $\mathbb{R}^2$, the rest of this discussion will be given in this context; however, all of the theory presented is easily generalized to $\mathbb{R}^n$. For a domain $\Omega \in \mathbb{R}^2$, the sets in a subdivision are usually triangles or quadrilaterals. One advantage of triangles and hence of triangulations is that any arbitrary configuration of non-collinear points can be made into the vertex set of a triangulation and there exist good algorithms for doing this automatically ([45]). The only type of subdivision to be considered in this exposition will be the triangulation although the method described can be modified to apply to quadrilaterals as well.

As is often the case in univariable functional estimation, the function spaces used in finite element estimation are spaces of polynomials. Let P_k denote the space of polynomials in 2 variables of degree less than or equal to k. For Lagrange elements, the space of shape functions, $\mathcal{P}$, for a given finite element is chosen to be P_k for some integer k. A convenient procedure for constructing a basis for this space is as follows. First, select a basis $\{\mathcal{N}_i\}_{i=1}^d$ for P_k', where $d = \dim(P_k)$, and let $\{\Phi_i\}_{i=1}^d \subset P_k$ be the dual basis to $\{\mathcal{N}_i\}_{i=1}^d$ such that

$$\mathcal{N}_i(\Phi_j) = \begin{cases} 1 & : \quad i = j \\ 0 & : \quad i \neq j \end{cases}$$

Then for any function $f \in \mathcal{P}_k$,

$$f = \sum_{i=1}^{d} \mathcal{N}_i(f)\Phi_i. \tag{4.5}$$

These three objects, a subdivision $\{K_i\}$, a finite dimensional space of functions $\mathcal{P}_i$ defined for each K_i, and a dual basis $\mathcal{N}_i$ for each $\mathcal{P}_i$, are all that are required for the formal definition of a finite element.

4.5 Triangular Quadratic Lagrange Elements

Given that the finite element solution of the smoothing problem requires polynomials of at least degree 2, the simplest solution will use P_2. It is well-known [11, p.70] that the dimension of P_k is $\binom{k+2}{2}$ and hence that the dimension of P_2 is 6. P_2 will serve as the space $\mathcal{P}$ of shape functions for the finite elements.

The quadratic (triangular) Lagrange element is constructed as follows. Let $\mathcal{T}_i$, the element domain, be the triangle defined by the three non-collinear vertices $\mathbf{p}_{i1}$, $\mathbf{p}_{i2}$, and $\mathbf{p}_{i3}$. Furthermore, let $\mathbf{p}_{i4}$, $\mathbf{p}_{i5}$, and $\mathbf{p}_{i6}$ be the midpoints of the edges $\overline{\mathbf{p}_{i1}\mathbf{p}_{i2}}$, $\overline{\mathbf{p}_{i2}\mathbf{p}_{i3}}$, and $\overline{\mathbf{p}_{i3}\mathbf{p}_{i1}}$, respectively. Now define the shape functions $\{\Phi_{ik}\}$, for $k = 1, \ldots, 6$, on $\mathcal{T}_i$ by

$$\Phi_{ik}(\mathbf{p}_{ij}) = \begin{cases} 1 & : \quad k = j \\ 0 & : \quad k \neq j \end{cases}$$

Since P_2 is a six-dimensional space, specifying the value of Φ_{ik} at six points actually constitutes a precise definition of Φ_{ik}. These shape functions correspond to the nodal variables. The shape functions $\{\Phi_{ik}\}$, which are thus far defined only on the triangle $\mathcal{T}_i$, are extended to all of Ω by giving them the value zero outside of $\mathcal{T}_i$.

Since these six functions form a basis for $P_2(\mathcal{T}_i)$, it follows that any function $f \in P_2(\mathcal{T}_i)$ can be expressed as

$$f = \sum_{j=1}^{6} f(\mathbf{p}_{ij})\Phi_{ij}.$$

Furthermore, for any function g defined on $\mathcal{T}_i$, the function

$$\mathcal{I}_{\mathcal{T}}(g) = \sum_{i=1}^{6} g(\mathbf{p}_{ij})\Phi_{ij}.$$

interpolates g at the points $\{\mathbf{p}_{ij}\}$.

For a triangulation $\mathcal{T}$ of $\Omega \subset \mathbb{R}^2$ consisting of t different triangular elements $\{\mathcal{T}_i\}_{i=1}^{t}$, the set of vertices and edge midpoints of $\mathcal{T}$ is called the set of nodes of $\mathcal{T}$. Since each node of a given triangular element domain $\mathcal{T}_i$ is associated with a nodal variable (evaluation at that node) on $\mathcal{T}_i$, one can associate each node with a nodal variable on Ω by defining that nodal variable to be the sum of the nodal variables for each element in which the given node appears.

Definition 9 ($\mathbf{H}_h^1$) *Let a triangulation $\mathcal{T} = \{\mathcal{T}_i\}$ be given. Then $H_h^1(\Omega)$ is defined to be the space of functions f in $H^1(\Omega)$ which, when restricted to the closure of any $\mathcal{T}_i$, are in $\mathcal{P}_2$. $H_{0h}^1(\Omega)$ is defined to be the subspace consisting of functions $f \in H_h^1(\Omega)$ which equal zero on the boundary of Ω and have normal derivative equal to zero on the boundary of Ω.*

Since the quadratic Lagrange function on any given element is completely determined by its values at the six nodes of that element, any function in $H_h^1(\Omega)$ is completely determined by its values at all of the nodes of $\mathcal{T}$. In addition, since any quadratic function on a line is completely defined by its values at three distinct points, the functions in H_h^1 are continuous over element boundaries.

It should be noted at this point that only domains with piecewise linear boundaries can be triangulated. If the domain Ω to be smoothed does not have such a boundary, two approaches can be adopted. The first approach approximates the boundary by a piecewise linear closed curve. For practical purposes, if the linear pieces are made small enough, such an approximation can often be acceptable. The second approach constructs the smoothing function on a domain which is larger than Ω so that the boundary of Ω lies in the interior of the larger domain. Unless it is desirable for the smoothed function to have the value zero on the entire boundary, the function will have to be estimated on such a larger domain anyway.

4.6 A Variational Formulation for the Laplacian Smoothing Problem

The first step in constructing a finite element solution is to restate the problem in a variational form which, along with the selection of an appropriate finite element space and boundary conditions, will be used to define the solution. As an example of a variational formulation of a PDE, let $g(x, y)$ be a real-valued function of the plane and consider the problem of finding a function f which satisfies the PDE

$$f_{xx} + f_{yy} = g \tag{4.6}$$

over some domain $\Omega \subseteq \mathbb{R}^2$. One way of attacking such a problem is to search for a function $f \in H^2(\Omega)$ which satisfies

$$\int_\Omega (f_{xx} + f_{yy})v = \int_\Omega gv \quad \forall v \in H^2(\Omega). \tag{4.7}$$

Note that, if f and v are chosen to belong to an arbitrary function space H then that space must be a subset of H^2 in order for this expression to be well-defined. It is certainly true that any function f which satisfies (4.6) must also satisfy (4.7). It is also true that if (4.7) is true for a given f then (4.6) must be true almost everywhere. The second equation, (4.7), is called a variational formulation of (4.6).

In the so-called *Galerkin* method, a variational formulation is used to compute an approximate solution to a PDE. This consists of defining a finite dimensional subspace H of the infinite dimensional function space in which the solution to the PDE is assumed to lie and then searching for an approximate solution to the variational formulation which lies in H. For the example above, this would mean defining a finite dimensional space $H \subset H^2$ and searching for a function $\bar{f} \in H$ such that

$$\int_\Omega (\bar{f}_{xx} + \bar{f}_{yy})\bar{v} = \int_\Omega \bar{v}g \quad \forall \bar{v} \in H.$$

The Galerkin method, together with other, similar methods, is the basis for the finite element method.

The variational approach used to approximate the bivariable L-spline in this thesis is a modification of a method presented in [13, pp.128-130] and [40, pp.144-150] for the solution of the biharmonic equation. The smoothing problem being considered in the following sections is that in (4.4) which attempts to minimize the sum of squared interpolation error using the Laplacian functional as the penalty functional. In section 4.11, the solution method will be extended to penalizing an arbitrary polynomial in the Laplacian. As in the case of ordinary differential equations, a partial differential equation has a unique solution only if sufficient boundary conditions are specified. The boundary condition imposed by the method to be presented in this section is that $\partial_\nu f$, the outward normal derivative of f, equal zero on the boundary $\partial\Omega$ of Ω.

For the purpose of image registration, one might desire a smooth function which is equal to zero on the boundary of Ω. This might occur, for example, if one were registering a rectangular image to another rectangular image with the same shape and size of boundary. In such a case, one would not want the estimated registration function to deform the boundary of the image. This special type of boundary condition, namely that $f = 0$ on $\partial\Omega$, can easily be enforced in a finite element setting. The method for doing so will be discussed in a later section of this chapter. Enforcing this extra boundary condition corresponds to solving the constrained problem of minimizing the interpolation sum of squares with a Laplacian penalty functional in the space of sufficiently smooth functions whose value *and* normal derivative equal zero on $\partial\Omega$.

The following theorem is proven in [10, p.35].

Theorem 4.1 (Characterization Theorem) *Let V be a linear space, and suppose that $a(\cdot,\cdot)$ is a symmetric, positive-definite linear form on V. In addition, let F be a linear functional on V. Then the quantity*

$$a(u,u) - 2F(u)$$

attains its minimum if and only if

$$a(u,v) = F(v) \textit{ for all } v \in V.$$

The proof of the characterization theorem also shows that the u which minimizes $a(u,u) - 2F(u)$, if it exists, is unique.

Note that the minimization criterion in (4.4) can be rewritten as

$$\int_\Omega (\Delta u)^2 + \lambda \sum_{i=1}^{n} u(\mathbf{p}_i)^2 - 2\lambda \sum_{i=1}^{n} u(\mathbf{p}_i) z_i + \lambda \sum_{i=1}^{n} (z_i)^2$$

and, since the last term does not depend on the function u, this means that the penalized fit is the function which minimizes the expression $B(u, u) - 2F(u)$ with $B(u, v) = \int_\Omega (\Delta u)(\Delta v) + \lambda \sum u(\mathbf{p}_i)v(\mathbf{p}_i)$ and $F(u) = \lambda \sum u(\mathbf{p}_i)z_i$.

It is a well-known fact from the theory of partial differential equations that if $\Delta f = 0$ on a compact domain Ω and the normal derivative of f is zero on $\partial\Omega$ then the function f must be constant on Ω ([22]). This result can be used to show that B is positive definite on H_0^2. Suppose that $B(f, f) = 0$ for some $f \in H_0^2(\Omega)$. Then $\int (\Delta f)^2 = 0$ and $\sum f(\mathbf{p}_i)^2 = 0$. The fact that $\int (\Delta f)^2 = 0$ implies that f is constant. If f is constant, then the fact that $\sum f(\mathbf{p}_i)^2 = 0$ implies that f must be zero on Ω.

Since B is positive definite on H_0^2, the characterization theorem implies that the minimizer of (4.4) in H_0^2 is the function $u \in H_0^2$ which satisfies the variational form $B(u, v) = F(v)$, or

$$\int_\Omega (\Delta u)(\Delta v) + \lambda \sum_{i=1}^{n} u(\mathbf{p}_i)v(\mathbf{p}_i) = \lambda \sum_{i=1}^{n} v(\mathbf{p}_i)z_i \qquad \forall v \in H_0^2. \tag{4.8}$$

One way to solve this problem approximately is to modify (4.4) by searching for a pair $(u, s) \in H_0^1 \times H^1$ of functions which satisfy the following pair of conditions:

$$s = \Delta u, \tag{4.9}$$

$$\int_\Omega s(\Delta v) + \lambda \sum_{i=1}^{n} u(\mathbf{p}_i)v(\mathbf{p}_i) = \lambda \sum_{i=1}^{n} v(\mathbf{p}_i)z_i, \quad \forall v \in H_0^1. \tag{4.10}$$

If such a pair of functions can be found then the function u would also be a solution of (4.4) in H^1. Also, any pair $(u, s) \in H_0^1 \times H^1$ which satisfied

$$\int_\Omega -(\Delta u)\ t + \int_\Omega st = 0 \qquad \forall t \in H^1 \tag{4.11}$$

would also have to satisfy (4.9) almost everywhere.

Green's theorem states that

$$\int_\Omega u \partial_i v = \int_{\partial\Omega} uv\nu_i - \int_\Omega v \partial_i u,$$

where ν_i is the i^{th} component of the outward normal ν, $\partial_\nu u$ is the directional derivative $\nabla u \cdot \nu$, and $\partial_i w$ is the partial derivative of w taken with respect to the i^{th} coordinate. Using this theorem and the fact that $\partial_\nu u = 0$ on $\partial\Omega$, it can be shown that

$$\int_\Omega (\Delta u)t = -\int_\Omega \nabla u \cdot \nabla t$$

and therefore (4.11) can be rewritten as

$$\int_\Omega \nabla u \cdot \nabla t + \int_\Omega st = 0 \quad \forall t \in H^1. \tag{4.12}$$

Similarly, (4.10) can be rewritten as

$$\int_\Omega \nabla s \cdot \nabla v - \lambda \sum_{i=1}^{n} u(\mathbf{p}_i)v(\mathbf{p}_i) = -\lambda \sum_{i=1}^{n} v(\mathbf{p}_i)z_i, \quad \forall v \in H_0^1. \tag{4.13}$$

The problem in (4.8) has now been simplified to one of finding a pair of functions $(s, u) \in H^1 \times H_0^1$ which satisfy both of the following conditions:

$$\begin{aligned} \int_\Omega \nabla s \cdot \nabla v - \lambda \sum u(\mathbf{p}_i)v(\mathbf{p}_i) &= -\lambda \sum v(\mathbf{p}_i)z_i \quad \forall v \in H_0^1 \\ \int_\Omega \nabla t \cdot \nabla u + \int_\Omega st &= 0 \qquad\qquad \forall t \in H^1. \end{aligned} \tag{4.14}$$

Since $H^2 \subset H^1$, this constitutes searching for a solution in a larger class of functions than is necessary. From the point of view of constructing finite element function spaces, however, solving this more general problem is computationally less expensive than the original problem.

The characterization theorem shows that the solution to (4.8), if it exists, is unique. In the literature of finite element theory, the existence of a solution to such a variational problem is usually established with the help of the Lax-Milgram theorem, which appears in [11] as follows.

Theorem 4.2 (Lax-Milgram Theorem) *Given a Hilbert space $(V, (\cdot,\cdot))$, a continuous, coercive bilinear form $a(\cdot,\cdot)$ on V and a continuous linear functional $F \in V'$, there exists a unique $u \in V$ such that*

$$a(u, v) = F(v) \text{ for all } v \in V.$$

Unfortunately, the Lax-Milgram theorem is not applicable in our case because the bilinear form $\int(\Delta u)(\Delta v) + \lambda \sum u(\mathbf{p}_i)v(\mathbf{p}_i)$ is not positive-definite on H_0^2. A good problem for future research would be to prove (or disprove) the existence of a solution to (4.8). It will, however, be shown in section 4.10 that the finite element approximation developed in the next section (section 4.7) does possess a unique solution.

4.7 The Finite Element Solution

Assume that $\{(\mathbf{p}_1, z_1), \ldots, (\mathbf{p}_d, z_d)\}$ are d observations of an unknown function, all of which lie inside some domain $\Omega \in \mathbb{R}^2$. Assume further that $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^d$ is a triangulation of Ω. Then the finite element version of the problem in (4.14) is defined by simply restricting the search for a solution to the finite element space $H_{0h}^1 \times H_h^1$.

The finite element approximation to the solution of (4.4) consists of constructing a triangulation of Ω and then finding $(u, s) \in H_{0h}^1 \times H_h^1$ such that

$$\begin{aligned} \int_\Omega \nabla s \cdot \nabla v - \lambda \sum u(\mathbf{p}_i)v(\mathbf{p}_i) &= -\lambda \sum v(\mathbf{p}_i) z_i \quad \forall v \in H_{0h}^1 \\ \int_\Omega \nabla t \cdot \nabla u + \int_\Omega st &= 0 \qquad\qquad \forall t \in H_h^1. \end{aligned} \tag{4.15}$$

where H_{0h}^1 is the subset of H_h^1 consisting of those functions which have normal derivatives equal to zero on the boundary of Ω.

Since every function in H_h^1 is, by definition, a quadratic polynomial on any given finite element (triangle), the value of a function is defined by its value at each of the six nodes. Solving the finite-element problem in (4.15), then, amounts to solving a linear system of equations whose unknowns are precisely the values of the solution at each of the nodes. This linear system can be represented by a matrix equation with a vector of unknowns and the solution to the problem will then be the solution of the matrix equation.

4.8 Setting Up the Linear System

Assume first that the data being smoothed occur at d points $\{\mathbf{p}_1, \ldots, \mathbf{p}_d\}$ in the plane and that to each point $\mathbf{p}_i$ there corresponds an observed function value z_i. In order to solve the system in (4.15), one constructs a system of matrices as follows. Suppose that there are n nodal variables (vertices and edge midpoints) in the triangulation $\mathcal{T}$ and let $\{\psi_i\}_{i=1}^n$ be the set of nodal basis functions corresponding to the nodal variables. Since the data points $\{\mathbf{p}_1, \ldots, \mathbf{p}_d\}$ are nodes of $\mathcal{T}$ and hence correspond to d nodal basis functions $\{\psi_{(1)}, \ldots, \psi_{(d)}\}$, assume that $\psi_{(i)} = \psi_i$ for each $1 \leq i \leq d$. In summary, there are n nodal variables (the set of vertices and edge midpoints) in the triangulation $\mathcal{T}$ of Ω. Each of these n nodal variables corresponds to one of n basis functions $\{\psi_1, \ldots, \psi_n\}$ for $H_h^1(\Omega)$, the space of piecewise quadratic functions on Ω. Furthermore, the functions are indexed so that $\{\psi_1, \ldots, \psi_d\}$ correspond to the nodal variables which are the d data points being smoothed.

By construction, H_{0h}^1 lies inside the span of $\{\psi_i\}$ and $\{\psi_i\}$ is a basis for H_h^1. Let ψ be the $n \times 1$ vector of the $\{\psi_i\}$ so that any function in H_h^1 or H_{0h}^1 can be uniquely expressed as $\mathbf{u}^t\psi$ for some $\mathbf{u} \in \mathbb{R}^m$. In addition, let ψ_x and ψ_y be the vectors of first order partial derivatives of ψ.

Now it is necessary to define a non-standard matrix and a vector called $\mathbf{P}$ and $\mathbf{z}$, respectively. Let $\mathbf{P}$ be an $n \times n$ matrix which is zero everywhere except for the d diagonal entries $\mathbf{P}_{1,1}, \cdots, \mathbf{P}_{d,d}$, each of which is equal to one. This is well-defined since $d \leq n$ by construction. Let $\mathbf{z}$ be the $n \times 1$ vector which is zero everywhere but the first d entries, where the i^{th} entry is equal to z_i. In other words, $\mathbf{z}^t = [z_1, z_2, \ldots, z_d, 0, \ldots, 0]$.

The problem in (4.15) can now be expressed as the problem of finding $(\mathbf{u}, \mathbf{s}) \in \mathbb{R}^n \times \mathbb{R}^n$ satisfying

$$
\begin{aligned}
\int (\mathbf{v}^t\psi_x\psi_x^t\mathbf{s}) + \int (\mathbf{v}^t\psi_y\psi_y^t\mathbf{s}) + \lambda\mathbf{v}^t\mathbf{P}\mathbf{u} &= \lambda\mathbf{v}^t\mathbf{z} \qquad \forall \mathbf{v} \in \mathbb{R}^n \\
\int (\mathbf{t}^t\psi_x\psi_x^t\mathbf{u}) + \int (\mathbf{t}^t\psi_y\psi_y^t\mathbf{u}) - \int (\mathbf{t}^t\psi\psi^t\mathbf{s}) &= \mathbf{0} \qquad \forall \mathbf{t} \in \mathbb{R}^n.
\end{aligned}
$$

By linearity, the coefficient vectors can be taken outside the integral and the two equations above can also be written as

$$\begin{aligned}\mathbf{v}^t \int (\psi_x \psi_x^t + \psi_y \psi_y^t)\mathbf{s} + \lambda \mathbf{v}^t \mathbf{P}\mathbf{u} &= \lambda \mathbf{v}^t \mathbf{z} \quad \forall \mathbf{v} \in \mathbb{R}^n \\ \mathbf{t}^t \int (\psi_x \psi_x^t + \psi_y \psi_y^t)\mathbf{u} - \mathbf{t}^t \int (\psi \psi^t)\mathbf{s} &= \mathbf{0} \quad \forall \mathbf{t} \in \mathbb{R}^n.\end{aligned}$$

Since these equations must be true for all $\mathbf{v}$ and $\mathbf{t}$, it follow that $(\mathbf{u}, \mathbf{s})$ is a solution to the above system of equations if and only if

$$\begin{aligned}\int (\psi_x \psi_x^t + \psi_y \psi_y^t)\mathbf{s} + \lambda \mathbf{P}\mathbf{u} &= \lambda \mathbf{z} \\ \int (\psi_x \psi_x^t + \psi_y \psi_y^t)\mathbf{u} - \int (\psi \psi^t)\mathbf{s} &= \mathbf{0}.\end{aligned}$$

If one defines the matrix $\mathbf{K}_0$ to be $\int(\psi\psi^t)$ and the matrix $\mathbf{K}_1$ to be $\int(\psi_x\psi_x^t+\psi_y\psi_y^t)$, this system can be expressed more compactly as

$$\begin{aligned}\mathbf{K}_1\mathbf{s} + \lambda \mathbf{P}\mathbf{u} &= \lambda \mathbf{z} \\ \mathbf{K}_0\mathbf{s} - \mathbf{K}_1^t\mathbf{u} &= \mathbf{0},\end{aligned} \tag{4.16}$$

and can be solved for its unique solution using linear algebra.

One might notice here that while H^1_{0h} is certainly contained in the span of $\{\psi_i\}$, these two sets are not equal. This is because H^1_{0h} was defined as all functions in $H^1_h(\Omega)$ which have zero normal gradient on the boundary. It might thus seem that the smoothing problem in (4.16) is not well-posed. In fact, the constraint that u have normal gradient equal to zero on the boundary is included in the formulation of (4.15) and hence in that of (4.16) as a *natural boundary condition.* A discussion of the difference between essential and natural boundary conditions can be found in virtually any text on the finite element method. A particular reference is [13, pp.18-19]. Suffice it to say here that natural boundary conditions are typically enforced by the Green's formula used in defining the variational form of the problem rather than in the construction of the finite element space. Furthermore, although the solution u will be an *approximation* of the true solution, which will have zero normal derivative on $\partial\Omega$, the normal derivative of u itself will not generally be zero on $\partial\Omega$.

4.9 Solutions Which Are Constrained To Equal Zero On The Boundary

Restricting the solution to the subspace of H^1_{0h} consisting of functions which equal zero on $\partial\Omega$ requires only a small modification of the construction presented in the previous section. Suppose that m of the n nodes of $\mathcal{T}$ lie in the interior of $\mathcal{T}$. Suppose further that the nodes, and hence the basis functions ψ_i, are numbered so that $\psi_1 \ldots \psi_m$ correspond to the interior nodes. For notational convenience, define the set $\{\phi_1, \ldots, \phi_m\}$ by the relation $\phi_j = \psi_j$ for $j = 1, \ldots, m$. If a function $f \in H^1_{0h}$ which vanishes on $\partial\Omega$ is represented as a linear combination of the ψ_i, then the coefficients of ψ_{m+1} through ψ_n will necessarily be zero. It follows that the ϕ_j span the space of functions in which the constrained solution is to be sought.

The constrained version of the problem in (4.15) can thus be rephrased as that of finding a pair $(\mathbf{u}, \mathbf{s}) \in \mathbb{R}^m \times \mathbb{R}^n$ satisfying

$$\begin{aligned} \int (\mathbf{v}^t \phi_x \psi_x^t \mathbf{s}) + \int (\mathbf{v}^t \phi_y \psi_y^t \mathbf{s}) + \lambda \mathbf{v}^t \mathbf{P} \mathbf{u} &= \lambda \mathbf{v}^t \mathbf{z} \qquad \forall \mathbf{v} \in \mathbb{R}^m \\ \int (\mathbf{t}^t \psi_x \psi_x^t \mathbf{u}) + \int (\mathbf{t}^t \psi_y \phi_y^t \mathbf{u}) - \int (\mathbf{t}^t \psi \phi^t \mathbf{s}) &= \mathbf{0} \qquad \forall \mathbf{t} \in \mathbb{R}^n. \end{aligned}$$

The only difference here is that the vector $\mathbf{z}$ will now be of length m, rather than n, and the square matrix $\mathbf{P}$ will have dimension m rather than n. The new vector $\mathbf{z}$ is obtained by using the first m entries of the old vector $\mathbf{z}$ (from the previous section). Since all of the data points must correspond to interior nodes, $d \leq m$ and no non-zero entries of $\mathbf{z}$ are lost in this truncation. Similiarly, the new matrix $\mathbf{P}$ is simply the submatrix obtained by removing the last $n - m$ rows and columns of the old matrix $\mathbf{P}$.

Using the same argument as in the previous section, it follows that $(\mathbf{u}, \mathbf{s})$ is a solution to the problem above if and only if

$$\begin{aligned} \int (\phi_x \psi_x^t + \phi_y \psi_y^t) \mathbf{s} + \lambda \mathbf{P} \mathbf{u} &= \lambda \mathbf{z} \\ \int (\psi_x \phi_x^t + \psi_y \phi_y^t) \mathbf{u} - \int (\psi \psi^t) \mathbf{s} &= \mathbf{0}. \end{aligned}$$

Defining the matrix $\mathbf{K}_0$ as before and the matrix $\mathbf{K}_1$ as $\int \phi\psi^t$, the solution can again be expressed as

$$\begin{aligned} \mathbf{K}_1\mathbf{s} + \lambda\mathbf{P}\mathbf{u} &= \lambda\mathbf{z} \\ \mathbf{K}_0\mathbf{s} - \mathbf{K}_1^t\mathbf{u} &= \mathbf{0}, \end{aligned}$$

Thus the only difference between solving the unconstrained and the constrained problems is that in the constrained version, $\mathbf{z}$, $\mathbf{P}$ and $\mathbf{K}_1$ are slightly smaller. $\mathbf{K}_1$ in the constrained problem is constructed by deleting the last $n-m$ *rows* of $\mathbf{K}_1$ from the unconstrained problem.

4.10 Uniqueness of Finite Element Solution

If the matrix $\mathbf{K}_0$ were invertible, then one could write the vector $\mathbf{s}$ in (4.16) as $\mathbf{s} = \mathbf{K}_0^{-1}\mathbf{K}_1^t\mathbf{u}$. In this case, the first equation could be written as $(\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t + \lambda\mathbf{P})\mathbf{u} = \lambda\mathbf{z}$. Again, if the matrix $[\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t + \lambda\mathbf{P}]$ were invertible, then the solution $\mathbf{u}$ to the finite element smoothing problem could be obtained as

$$\mathbf{u} = \lambda[\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t + \lambda\mathbf{P}]^{-1}\mathbf{z}. \tag{4.17}$$

A closer examination of the properties of the matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ is enough to show that the system of equations in (4.16) does, indeed, admit a unique solution $(\mathbf{u}, \mathbf{s})$. Recall that these matrices were defined as $\mathbf{K}_0 = \int(\psi\psi^t)$ and $\mathbf{K}_1 = \int(\psi_x\psi_x^t + \psi_y\psi_y^t)$, where ψ and is the vector of basis functions for the space H_h^1 and ψ_x and ψ_y are the vectors of partial derivatives of those basis functions. Vectors such as ψ, ψ_x and ψ_y can be used to create *Gram* matrices, which are defined as follows.

Definition 10 *Let $\{v_1, \ldots, v_k\}$ be a set of k given vectors in an inner product space $(V, \langle\cdot,\cdot\rangle)$. The* **Gram matrix** *of the vectors $\{v_1, \ldots, v_k\}$ with respect to the inner product $\langle\cdot,\cdot\rangle$ is the $k \times k$ matrix $\mathbf{G} = [g_{ij}]$ defined by $g_{ij} = \langle v_i, v_j\rangle$.*

It can be seen from this definition that $\mathbf{K}_0$ is the Gram matrix of the vectors $\{\psi_1, \dots, \psi_n\}$. A property of Gram matrices is that they are always positive semi-definite and are positive definite if and only if the vectors defining the matrix are linearly independent ([25]). Since $\{\psi_1, \dots, \psi_n\}$ are a basis, they must necessarily be linearly independent and hence $\mathbf{K}_0$ is positive definite. Since positive definite matrices are always of full rank, it follows further that $\mathbf{K}_0$ is invertible.

The following argument shows that the matrix $\mathbf{K} = [\mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t + \lambda \mathbf{P}]$ is positive definite. Since $\mathbf{P}$ is positive semi-definite by definition and $\mathbf{K}_0$ is positive definite, $\mathbf{K}$ must at least be positive semi-definite. Now we suppose that $\mathbf{c}^t \mathbf{K} \mathbf{c} = 0$ and show that $\mathbf{c}$ must equal $\mathbf{0}$. Since

$$\mathbf{c}^t \mathbf{K} \mathbf{c} = \mathbf{c}^t [\mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t + \lambda \mathbf{P}] \mathbf{c} = \mathbf{c}^t \mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t \mathbf{c} + \mathbf{c}^t \lambda \mathbf{P} \mathbf{c}$$

and both terms on the right are non-negative, it follows that $\mathbf{c}^t \mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t \mathbf{c} = 0$ and $\mathbf{c}^t \lambda \mathbf{P} \mathbf{c} = 0$. Since $\mathbf{K}_0$, and hence $\mathbf{K}_0^{-1}$, is positive definite, the fact that $\mathbf{c}^t \mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t \mathbf{c} = 0$ implies that $\mathbf{K}_1^t \mathbf{c} = \mathbf{0}$. It follows that $\mathbf{c}^t \mathbf{K}_1^t \mathbf{c} = 0$.

Now

$$\mathbf{c}^t \mathbf{K}_1 \mathbf{c} = \mathbf{c}^t \int (\psi_x \psi_x^t + \psi_y \psi_y^t) \mathbf{c} = \int [\mathbf{c}^t (\psi_x \psi_x^t + \psi_y \psi_y^t) \mathbf{c}] = \|\mathbf{c}^t \psi_x\|_2^2 + \|\mathbf{c}^t \psi_y\|_2^2,$$

where the norm used is the $\mathcal{L}^2$-norm. Since both terms are non-negative, $\mathbf{c}^t \psi$ is a piecewise quadratic finite element function both of whose partial derivatives vanish and hence $\mathbf{c}^t \psi$ must be constant. The vector $\mathbf{c}^t$ contains the values of $\mathbf{c}^t \psi$ evaluated at each of the nodes of Ω and hence must be of the form $[\alpha, \alpha, \dots, \alpha]$ for some constant α.

Recall that all of the entries of the matrix $\lambda \mathbf{P}$ are zero except for d of the diagonal entries, which are λ. It follows that the number $\mathbf{c}^t \lambda \mathbf{P} \mathbf{c} = [\alpha, \alpha, \dots, \alpha] \lambda \mathbf{P} [\alpha, \alpha, \dots, \alpha]^t$ is simply $d \lambda \alpha^2$. Since λ and d are nonzero, the fact that $d \lambda \alpha^2 = 0$ implies that α, and hence the vector $\mathbf{c}$, is zero. This establishes that $\mathbf{K}$ is positive definite.

Since $\mathbf{K} = [\mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t + \lambda \mathbf{P}]$ is positive definite and, consequently, invertible, the finite element problem must have a unique solution. A similiar argument shows that constrained problem also has a unique solution.

4.11 Penalizing Higher Powers of the Laplacian

Given the technique presented thus far, it is possible to penalize any polynomial function of the Laplacian operator. First of all, suppose one wanted to smooth the observations $\{(\mathbf{p}_1, z_1), \ldots, (\mathbf{p}_d, z_d)\}$ while penalizing $L = \Delta^n$, the n^{th} power of the Laplacian, by finding the function $f \in H_0^{2n}$ which minimizes

$$\int_\Omega (\Delta^n f)^2 + \lambda \sum_{i=1}^{d} (f(\mathbf{p}_i) - z_i)^2$$

or, equivalently, such that

$$\int_\Omega (\Delta^n f)(\Delta^n v) + \lambda \sum_{i=1}^{d} f(\mathbf{p}_i) v(\mathbf{p}_i) = \lambda \sum_{i=1}^{d} v(\mathbf{p}_i) z_i, \quad \forall v \in H_0^{2n}. \tag{4.18}$$

This can be reduced to a problem in H_0^1 as follows. Let $f_0 = f$ and let $f_i = \Delta^i f$ for $i = 1 \ldots 2n-1$. It is then evident that the integral $\int (\Delta^n f)(\Delta^n v)$ can be replaced by $\int f_n (\Delta^n v)$. Furthermore, if one assumes that f_n satisfies the boundary condition $\partial_\nu f = 0$ on the boundary of Ω, one can apply Green's theorem twice to obtain $\int f_n(\Delta^n v) = \int f_{n+1}(\Delta^{n-1} v)$. If one imposes the same boundary condition on $f_{n+1}, \ldots, f_{2n-2}$, one can inductively see that $\int f_n(\Delta^n v) = \int f_{2n-1} \Delta v$. Imposing the same boundary condition on v then yields $\int f_n(\Delta^n v) = \int \nabla f_{2n-1} \cdot \nabla v$.

Therefore, if we define the space $H_0^1(\Omega)$ to be that subspace of $H^1(\Omega)$ of functions which have normal derivatives equal to zero on the boundary of Ω, the problem in (4.18) can be simplified to that of finding the n-tuple of functions $(f_0, f_1, \ldots, f_{2n-1}) \in (H_0^1)^{2n-1} \times H^1$ which satisfy the following $2n$ equations (the second equation must hold for each j in $\{1, \ldots, 2n-1\}$):

$$\begin{aligned}
\int \nabla f_{2n-1} \cdot \nabla v_0 \;-\; \lambda \sum_{i=1}^{d} f_0(\mathbf{p}_i) v_0(\mathbf{p}_i) &= \;-\lambda \sum_{i=1}^{d} v_0(\mathbf{p}_i) z_i \quad \forall v \in H_0^1 \\
\int \nabla f_{j-1} \cdot \nabla v_j \;+\; \int_\Omega f_j v_j &= \quad 0 \qquad\qquad \forall v_j \in H_0^1 \\
\int \nabla f_{2n-2} \nabla v_{2n-1} \;+\; \int_\Omega f_{2n-1} v_{2n-1} &= \quad 0 \qquad\qquad \forall v_{2n-1} \in H^1.
\end{aligned} \tag{4.19}$$

For the general case where $L = \sum_{k=1}^{m} a_k \Delta^i$, it suffices to solve the set of equations in (4.19) with the first equation changed to

$$\int \sum_{k=1}^{m} a_k \left(\nabla f_{2k-1} \cdot \nabla v\right) \; - \; \lambda \sum_{i=1}^{d} f_0(\mathbf{p}_i) v(\mathbf{p}_i) = -\lambda \sum_{i=1}^{d} v(\mathbf{p}_i) z_i \quad \forall v \in H_0^1.$$

The method for computing a finite element approximation to the smoothing function defined by the equations in (4.19) is exactly analogous to that discussed earlier for penalizing the square of the Laplacian. Recall that ψ was defined to be the $n \times 1$ vector of basis functions of H_h^1 and that ψ_x and ψ_y were defined to be the $n \times 1$ vectors of partial derivatives of ψ. Furthermore, the $n \times 1$ vector $\mathbf{z}$ was defined to equal zero everywhere but for the first d entries. The i^{th} entry of $\mathbf{z}$, for $1 \leq i \leq d$, equals z_i, the observed value at the i^{th} data point.

The matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ are again defined by $\mathbf{K}_0 = \int \psi\psi^t$ and $\mathbf{K}_1 = \int(\psi_x\psi_x^t + \psi_y\psi_y^t)$ and $\mathbf{P}$ is the $n \times n$ matrix which is zero everywhere except for the entries $\mathbf{P}_{ii}$, for $1 \leq i \leq d$, which equal 1. Then the discretized version of the problem in (4.19) is to find $\{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{2n-1}\} \in (\mathbb{R}^n)^{2n}$ such that

$$\begin{aligned}
\mathbf{v}_0^t\mathbf{K}_1\mathbf{u}_{2n-1} - \lambda\mathbf{v}_0^t\mathbf{P}\mathbf{u}_0 \quad &= -\lambda\mathbf{z} \quad && \forall \mathbf{v}_0 \in \mathbb{R}^n \\
\mathbf{v}_j^t\mathbf{K}_1\mathbf{u}_{j-1} + \mathbf{v}_j^t\mathbf{K}_0\mathbf{u}_j \quad &= \mathbf{0} && \forall \mathbf{v}_j \in \mathbb{R}^n \\
& && \text{for } 1 \leq j \leq 2n-2 \\
\mathbf{v}_{2n-1}^t\mathbf{K}_1^t\mathbf{u}_{2n-2} + \mathbf{v}_{2n-1}^t\mathbf{K}_0\mathbf{u}_{2n-2} \quad &= \mathbf{0} && \forall \mathbf{v}_{2n-1} \in \mathbb{R}_n.
\end{aligned}$$

Once again, since the above equations need to be satisfied for any $\{v_0, v_1, \ldots, v_{2n-1}\} \in (\mathbb{R}^n)^{2n}$, this reduces to the problem of finding $\{u_0, u_1, \ldots, u_{2n-1}\} \in (\mathbb{R}^n)^{2n}$ such that

$$\begin{aligned}
\mathbf{K}_1\mathbf{u}_{2n-1} - \lambda\mathbf{P}\mathbf{u}_0 \qquad &= -\lambda z \\
\mathbf{K}_1\mathbf{u}_{j-1} + \mathbf{K}_0\mathbf{u}_j \qquad &= \mathbf{0} \\
\text{for } 1 \leq j \leq 2n-2 & \\
\mathbf{K}_1^t\mathbf{u}_{2n-2} + \mathbf{K}_0\mathbf{u}_{2n-2} \qquad &= \mathbf{0}.
\end{aligned} \tag{4.20}$$

One can see that these equations have a unique solution using the same logic as was used in the case of the simple Laplacian. By examining the equations in (4.20) one can see that

$$\mathbf{u}_1 = -\mathbf{K}_0^{-1}\mathbf{K}_1\mathbf{u}_0$$

and, inductively, that

$$\mathbf{u}_i = (-1)^i[\mathbf{K}_0^{-1}\mathbf{K}_1]^i\mathbf{u}_0$$

for $0 \leq i \leq 2n-2$. This implies that

$$\mathbf{u}_{2n-2} = [\mathbf{K}_0^{-1}\mathbf{K}_1]^{2n-2}\mathbf{u}_0$$

and it follows that

$$\mathbf{u}_{2n-1} = -\mathbf{K}_0^{-1}\mathbf{K}_1^t[\mathbf{K}_0^{-1}\mathbf{K}_1]^{2n-2}\mathbf{u}_0$$

and hence that the first equation in (4.20) can be re-expressed as

$$\left[\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t[\mathbf{K}_0^{-1}\mathbf{K}_1]^{2n-2} + \lambda\mathbf{P}\right]\mathbf{u}_0 = \lambda\mathbf{z}.$$

The matrix multiplying $\mathbf{u}_0$ on the left can be seen to be invertible by looking at the component matrices. The fact that $\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t$ is positive definite was demonstrated when showing that the smoothing problem which penalizes the square of the Laplacian had a unique solution. The inverse $\mathbf{A}^{-1}$ of a positive definite matrix $\mathbf{A}$ is also positive definite since the eigenvalues of $\mathbf{A}^{-1}$ are the inverses of the eigenvalues of $\mathbf{A}$. The product of two positive definite matrices is also positive definite, since

$$\mathbf{u}^t\mathbf{A}\mathbf{B}\mathbf{u} = \frac{1}{||\mathbf{u}||}\mathbf{u}^t\mathbf{A}\mathbf{u}\mathbf{u}^t\mathbf{B}\mathbf{u},$$

and hence the matrix $\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t[\mathbf{K}_0^{-1}\mathbf{K}_1]^{2n-2}$ is positive definite. Since $\mathbf{P}$ is positive semi-definite and λ is non-negative, the matrix $[\mathbf{K}_1\mathbf{K}_0^{-1}\mathbf{K}_1^t[\mathbf{K}_0^{-1}\mathbf{K}_1]^{2n-2} + \lambda\mathbf{P}]$ is positive definite and hence of full rank.

To simplify the notation somewhat, define the matrix $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{K}_0^{-1}\mathbf{K}_1.$$

Then the function $u = \mathbf{u}_0^t\psi$ which solves the finite element smoothing problem of minimizing

$$\sum_{i=1}^{t}[\mathbf{u}(\mathbf{p}_i) - z_i]^2 + \lambda\int_{\Omega}[\Delta^n\mathbf{u}]^2 d\Omega$$

over all $\mathbf{u} \in H^1_{0h}$ can be solved as

$$\mathbf{u}_0 = \lambda \left[\mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t \mathbf{M}^{2n-2} + \lambda \mathbf{P}\right]^{-1} \mathbf{z}.$$

The more general problem of finding $\mathbf{u}_0 \in H^1_{0h}$ which minimizes

$$\sum_{i=1}^{t} [\mathbf{u}(\mathbf{p}_i) - z_i]^2 + \lambda \int_\Omega [\sum_{k=1}^{n} a_k \Delta^k \mathbf{u}]^2 d\Omega$$

can, similarly, be solved as

$$\mathbf{u}_0 = \lambda \left[\lambda \mathbf{P} + a_n \mathbf{K}_1 \mathbf{K}_0^{-1} \mathbf{K}_1^t \mathbf{M}^{2n-2} - \sum_{k=1}^{n-1} a_k \mathbf{K}_1 \mathbf{M}^{2m-1}\right]^{-1} \mathbf{z}. \tag{4.21}$$

The general bivariable L-spline smoothing problem, namely that of finding the function f which minimizes the expression

$$\sum_{i=1}^{t} [f(p_i) - z_i]^2 + \lambda \int_\Omega [\sum_{k-1}^{n} a_k \Delta^n f]^2 d\Omega,$$

is thus approximately solved by the finite element function with nodal values given by u_0 of equation (4.21). This solution can be generalized to the multivariable (m-dimensional) case by letting the finite elements by m-simplices. In particular, a 2-simplex is a triangle and a 3-simplex is a tetrahedron.

4.12 The finite element L-spline contrasted with the thin-plate spline

The definition of the thin-plate spline is very similar to the definition of the bivariable L-spline which penalizes the squared Laplacian. The integrand in the penalty term of the thin-plate smoothing spline is

$$f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2$$

while that in the bivariable L-spline is

$$(\Delta f)^2 = f_{xx}^2 + 2f_{xx}f_{yy} + f_{yy}^2.$$

Both of these penalties are invariant to the choice of axes and both have linear functions (planes) in their null-space. Some of the differences between the two estimation methods are discussed in this section. For the sake of brevity, a finite element L-spline will be referred to as a feL-spline.

One distinct advantage of the thin-plate spline is that its solution is C^∞. Although the radial basis function which appears in the solution to the thin-plate spline is not defined at the origin, it can be extended by continuity to an infinitely differentiable function. The feL-spline, on the other hand, is usually not even C^1.

At the level of computing numerical solutions, it is not clear that either method is really to be preferred. The matrix involved in computing the thin-plate spline will be much smaller. The reason for this is that the thin-plate spline has only one basis function for each data point plus three to define the linear part. In contrast, the feL-spline has one basis function for each node and there will usually be more nodes than data points.

The matrix arising from the finite element method will be very sparse, however, while the matrix arising from the thin-plate spline computation will be full. The time required to solve the finite element system is likely to be longer, but the time difference will not be as large as the sizes of the matrices involved might suggest.

In order to work with the solution to either of the two methods, one will typically need to compute the value of the solution at a large number of points. In the registration procedure suggested in this thesis, for example, it is necessary to evaluate the solution at each pixel in the image domain. To compute the value of the finite element solution at a given point, one must evaluate six quadratic polynomials. In order to compute the value of a thin-plate spline, on the other hand, one must evaluate one linear function and one radial basis function for each data point. If many data points are involved, evaluation of the finite element function will be substantially faster. Powell ([42]) has devised a method based on truncated Laurent series for speeding up the evaluation of thin-plate splines on a regular grid, but evaluating a finite element function is simpler and still likely to be faster.

To summarize, the computational differences between the thin-plate spline and the feL-spline arise both in computing and in evaluating the functions. While it is likely to take more time to compute the feL-spline due to having to invert a larger linear system, it is difficult to evaluate the magnitude of the difference. This difficulty is due to the fact that the feL-spline system gives rise to a very sparse matrix whereas the matrix associated with the thin-plate spline system is full. Evaluation of the solution at a given point, on the other hand, will usually be faster for the feL-spline than for the thin-plate spline. This difference in evaluation time is important because the methods of functional data analysis will often require evaluating the solution at a large number of points.

Another difference between the thin-plate spline and the feL-spline is the fact that they minimize their respective penalties over different domains. The thin-plate spline penalty is minimized over all of $\mathbb{R}^2$, but the feL-spline penalty is minimized over a finite domain Ω. While it is stated in [47] that the choice of the domain over which the thin-plate spline penalty is minimized will affect the solution to the problem, I know of no algorithm for computing the solution on any domain but $\mathbb{R}^2$.

The thin-plate spline is not generalizable to more than two dimensions. There is an analogous construction in three dimensions, but not higher. The three dimensional analogue, however, is no longer C^1 and is not differentiable at the data points ([47]). The feL-spline, on the other hand, can easily be extended to n dimensions, for any $n > 0$. The only real difference is that the finite element used in n dimensions will be an n-simplex. In three dimensions, for example, this would be a tetrahedron.

I know of no method for constructing thin-plate splines which incorporate arbitrary boundary conditions or constraints on the value of the solution. In particular, as far as I am aware, it is not possible to construct a thin-plate spline with a specified value on the boundary of a domain or on a curve inside the domain. It is not difficult to construct a feL-spline with specified values at any set of specified nodes. This only involves deleting the corresponding rows and columns of $\mathbf{K}_0$ and $\mathbf{K}_1$ and then modifying the value of certain entries of $\mathbf{z}$ to account for the fixed nodes' contributions to the penalty functional. In order to construct a feL-spline with a fixed value on the boundary of a domain Ω, one needs only fix the values of the boundary nodes. Of course, when fixing the value on the boundary of Ω, it must be remembered that the solution will have to be continuous and quadratic between adjacent vertices. Among the quadratic finite element functions on Ω with the specified values on the fixed nodes, the solution would be the one which minimized the penalized interpolation sum of squares. In this sense, the solution can be considered to be a constrained feL-spline on Ω.

The feL-spline smoothing technique can penalize the square of any linear partial differential operator which is invariant to both translation and rotation. The method for computing the thin-plate spline has been generalized somewhat, at least at the theoretical level, to higher order penalties. Mitáš and Mitášová, in [38], suggest modifying the smoothness characteristics of the thin-plate spline interpolant by changing the minimized expression to

$$\int [(f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2) + \rho(f_x^2 + f_y^2) + \tau(f_{xxx}^2 + 3f_{xxy}^2 + 3f_{xyy}^2 + f_{yyy}^2)], \qquad (4.22)$$

where τ and ρ are non-negative constants. The expression in (4.22) corresponds to the usual penalty function associated with the thin-plate spline with two added terms, one involving the first order partial derivatives and one involving the third order partials. The authors refer to splines based on (4.22) as "thin-plate splines with tension" or, alternatively, as "tension splines" because the penalty can be interpreted as the bending energy of a thin plate with tension applied to the boundary (although the solution technique still assumes that (4.22) is to be minimized over the entire plane).

Mitáš and Mitášová offer a general method for computing the Green's function of this penalty for a given τ and ρ. Using these Green's functions, one can construct the exact solution to the minimization problem. Chen and Suter, in [15], explicitly derive the Green's functions for a variety of values of ρ and τ. In the same technical report, they use several different combinations of τ and ρ to compute tension spline interpolants of a set of 25 observations in the plane. Although both the Mitáš and Mitášová and the Chen and Suter papers consider only the interpolation problem, the results can be generalized to smoothing.

Although Mitáš and Mitášová's results give a method for computing the smoothing splines based on (4.22), the basis functions for non-zero values of ρ and/or τ are complicated expressions involving modified Bessel functions of the second kind. These modified Bessel functions are defined in terms of power series and hence the evaluation of a tension spline at a large number of points will certainly be substantially more expensive than the evaluation of a finite element function. To the best of my knowledge, bivariable tension splines have not been used in any applied smoothing problem.

Chapter 5

Constructing the Matrices and Solving the Linear System

5.1 The Structure of the Problem

Recall from the equations (4.16) in section 4.8 that solving the finite-element minimization problem involves finding vectors $\mathbf{u}$ and $\mathbf{s}$ that satisfy the equations

$$\begin{aligned} \mathbf{K}_1\mathbf{s} + \lambda\mathbf{P}\mathbf{u} &= \lambda\mathbf{z} \\ \mathbf{K}_0\mathbf{s} - {\mathbf{K}_1}^t\mathbf{u} &= 0. \end{aligned}$$

$\mathbf{P}$ is an easily constructed diagonal matrix but the construction of $\mathbf{K}_0$ and $\mathbf{K}_1$ is non-trivial. In fact, the construction of $\mathbf{K}_0$ and $\mathbf{K}_1$ constitutes a substantial part of the computational load involved in computing an feL-spline. These steps should therefore be implemented in as efficient a manner as possible. This chapter offers a detailed description of how to construct $\mathbf{K}_0$ efficiently. The construction of $\mathbf{K}_1$ is somewhat more complicated but follows the same principles. A suggestion on the most efficient manner in which to solve the resulting linear system will also be offered. The explanation will be for solving the unconstrained smoothing problem but the techniques presented are also applicable to solving the constrained case.

Recall that the matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ are defined to be $\int_\Omega(\psi\psi^t)$ and $\int_\Omega(\psi_x\psi_x^t +$

$\psi_y \psi_y^t$), respectively, where ψ is the vector of basis functions for H_h^1. The vectors ψ_x and ψ_y are the first-order partial derivatives of ψ.

Thus the ijth entry of $\mathbf{K}_0$ is $\int(\psi_i \psi_j)$ and the ijth entry of $\mathbf{K}_1$ is

$$\int \left(\frac{\partial}{\partial x}\psi_i \frac{\partial}{\partial x}\psi_j + \frac{\partial}{\partial y}\psi_i \frac{\partial}{\partial y}\psi_j \right).$$

In order to compute the entries in $\mathbf{K}_0$ and $\mathbf{K}_1$, one needs to bear in mind that each ψ_i is a shape function associated with a node n_i and that each node n_i is either a vertex or an edge midpoint of some triangle (element domain) in the triangulation of Ω. This shape function is defined piecewise on each element domain $\mathcal{K}_j$ according to whether or not $n_i \in \mathcal{K}_j$. If the node n_i is not a vertex or an edge midpoint of $\mathcal{K}_j$, then ψ_i equals zero on all of $\mathcal{K}_j$. If, on the other hand, n_i is a node of $\mathcal{K}_j$, then ψ_i is the quadratic polynomial constructed so as to equal one at n_i and zero at each other node of $\mathcal{K}_j$. As was mentioned before, since there are six nodes per element and the space of quadratic bivariate functions is of dimensions six, this characterization uniquely determines the value of ψ_i on each element.

Since the shape functions are defined piecewise on each element, the matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ can be computed element by element. Each element contains six nodes and so will contribute to six squared, or thirty-six, components of each of the matrices. In addition, since the shape functions ψ_i vanish on most elements of the triangulation, most components of $\mathbf{K}_0$ and $\mathbf{K}_1$ will be zero. This sparsity property of the matrices is a general characteristic of the finite element method and makes the matrix computations involved in solving finite element problems surprisingly efficient.

Given the fact that the ψ_i are defined piecewise on each element, it would seem at first glance that evaluating the integrals involved in constructing $\mathbf{K}_0$ and $\mathbf{K}_1$ would require computing the shape functions for each element. In fact, the integrals can be evaluated without ever computing the individual shape functions at all. This is done by mapping each element onto one standard element, called the *master element*, using an affine transformation and then doing all of the necessary computations on the master element. The rest of this chapter contains a complete description of how this is done. *The material in sections 5.2, 5.3, 5.4 and 5.5 paraphrases an account of the same procedure which is found in [8, Ch.5].* Sections 5.6 and 5.7 are drawn from my own experience with implementing the feL-spline computations in MATLAB.

5.2 The Master Element

This section defines the geometry of the master element precisely and defines the notation needed to present the method for using the master element to integrate shape functions over an arbitrary element. For the remainder of this chapter, it will be assumed that $\mathcal{K}$ is an arbitrary element of the triangulated domain Ω with nodes $n_i = (x_i, y_i)$ and corresponding shape functions ϕ_i, for $i = 1, \cdots, 6$. It will also be assumed that n_1 is a vertex node and that the nodes are numbered in a counter-clockwise fashion around $\mathcal{K}$. Thus n_2 is the first edge midpoint proceeding counter-clockwise from n_1, n_3 is the next vertex, and so on.

This definition is not completely consistent with the formal definition of a finite element, where $\mathcal{K}$ was simply the domain of the element and the element itself was the triple $(\mathcal{K}, \mathcal{P}, \mathcal{N})$. In the formal definition, $\mathcal{P}$ was the set of shape functions and $\mathcal{N}$ was the set of nodal variables. Nonetheless, the material in this chapter is much easier to present if it is simply assumed that $\mathcal{K}$ is a triangular element which comes with six shape functions and six nodes. To further abuse the notation, however, a function defined on the domain of the element $\mathcal{K}$ will still be referred to as a function defined on $\mathcal{K}$. Therefore the symbol $\mathcal{K}$ will actually be used to mean two different things: a *finite element* and the *domain* of a finite element. The context will make clear which meaning is intended. The master element is defined as follows.

Definition 11 (Master Element) *The master element, $\widehat{\mathcal{K}}$, is defined to be the triangle with vertices at the points $(0,0)$, $(1,0)$ and $(0,1)$. The nodes of the master element are defined to be $\widehat{n}_1 = (0,0)$, $\widehat{n}_2 = (1/2, 0)$, $\widehat{n}_3 = (1,0)$, $\widehat{n}_4 = (1/2, 1/2)$, $\widehat{n}_5 = (0,1)$ and $\widehat{n}_6 = (0, 1/2)$. The shape functions of the master element are defined as usual and are denoted as $\widehat{\phi}_i$ for $i = 1, \cdots, 6$.*

For the sake of clarity, a point in the domain of $\mathcal{K}$ will be parametrized as (x, y) and a point in the domain of $\widehat{\mathcal{K}}$ will be parametrized as (ξ, ν). One can verify by direct evaluation at the $\widehat{n}_i$ that the explicit forms of the shape functions are

$$\widehat{\psi}_1 = 1 - 3\xi - 3\nu + 2\xi^2 + 4\xi\nu + 2\nu^2$$

$$\widehat{\psi}_2 = 4(\xi - \xi^2 - \xi\nu)$$

$$\widehat{\psi}_3 = 2\xi^2 - \xi$$

$$\widehat{\psi}_4 = 4\xi\nu$$

$$\widehat{\psi}_5 = 2\nu^2 - \nu$$

$$\widehat{\psi}_6 = 4(\nu - \xi\nu - \nu^2).$$

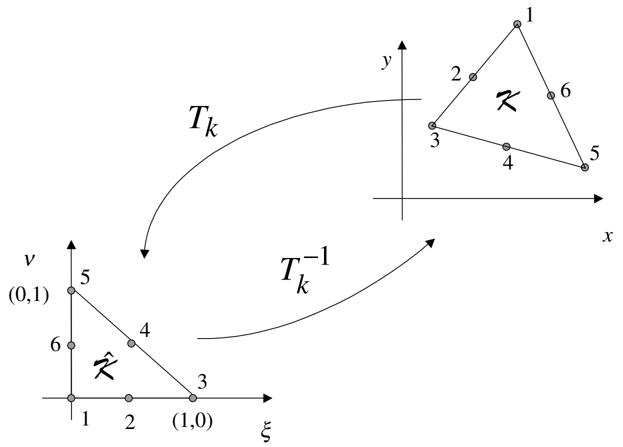

Figure 5.1: Transformation from an arbitrary element $\mathcal{K}_i$ to the master element.

For any nondegenerate element $\mathcal{K}$ (that is, a triangle whose vertices are not collinear), there is an affine transformation $T_{_K} : \widehat{\mathcal{K}} \to \mathcal{K}$ defined by the change-of-coordinate functions

$$T_{_K}: \begin{array}{rcl} x & = & x(\xi, \nu) \\ y & = & y(\xi, \nu). \end{array}$$

This transformation, which is illustrated in figure 5.1, maps the vertices of $\widehat{\mathcal{K}}$ to the vertices of $\mathcal{K}$ and maps straight lines to straight lines. There is a corresponding inverse transformation, which is also an affine transformation:

$$T_{_K}^{-1}: \begin{array}{rcl} \xi & = & \xi(x, y) \\ \nu & = & \nu(x, y). \end{array}$$

5.3 Computing On The Master Element

It will be assumed in this section that the change-of-coordinate functions are continuously differentiable. The next section will show that this assumption is correct. It follows from the continuous differentiability assumption that the infinitesimal quantities dx and dy are related to $d\xi$ and $d\nu$ by the matrix expression

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \nu} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \nu} \end{bmatrix} \begin{bmatrix} d\xi \\ d\nu \end{bmatrix}.$$

The square matrix in this equality is called the *Jacobian* matrix $\mathbf{J}$ of $T_{_K}$. Assuming that $|\mathbf{J}| \neq 0$, the inverse relation can also be expressed using $\mathbf{J}^{-1}$:

$$\begin{bmatrix} d\xi \\ d\nu \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} dx \\ dy \end{bmatrix} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} \frac{\partial y}{\partial \nu} & -\frac{\partial x}{\partial \nu} \\ -\frac{\partial y}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix}. \tag{5.1}$$

Of course, this relation can also be expressed as

$$\begin{bmatrix} d\xi \\ d\nu \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \nu}{\partial x} & \frac{\partial \nu}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix}. \tag{5.2}$$

By equating terms in (5.1) and (5.2), one can observe that

$$\begin{array}{ll} \frac{\partial \xi}{\partial x} = \frac{1}{|\mathbf{J}|}\frac{\partial y}{\partial \nu}, & \frac{\partial \xi}{\partial y} = -\frac{1}{|\mathbf{J}|}\frac{\partial x}{\partial \nu} \\ \frac{\partial \nu}{\partial x} = -\frac{1}{|\mathbf{J}|}\frac{\partial y}{\partial \xi}, & \frac{\partial \nu}{\partial y} = \frac{1}{|\mathbf{J}|}\frac{\partial x}{\partial \xi}. \end{array} \tag{5.3}$$

These equalities will be essential when using the master element to compute the entries of both $\mathbf{K}_0$ and $\mathbf{K}_1$.

A function g defined on $\mathcal{K}$ can be thought of a function $\widehat{g}$ on $\widehat{\mathcal{K}}$ by the equality

$$g(x, y) = g(x(\xi, \nu), y(\xi, \nu)) = \widehat{g}(\xi, \nu).$$

Similarly, one can express a function $\widehat{g}$ on $\widehat{\mathcal{K}}$ as a function on $\mathcal{K}$:

$$\widehat{g}(\xi, \nu) = \widehat{g}(\xi(x, y), \nu(x, y)) = g(x, y).$$

Furthermore, a quadratic function $\widehat{g}$ on $\widehat{\mathcal{K}}$ can be defined using the shape functions $\widehat{\phi}_i$ and the value of $\widehat{g}$ at the nodes of $\widehat{\mathcal{K}}$. Denoting the value of $\widehat{g}$ at the node $\widehat{n}_i$ by $\widehat{g}_i$, this definition of $\widehat{g}$ takes the form

$$\widehat{g}(\xi, \nu) = \sum_{i=1}^{6} \widehat{g}_i \widehat{\psi}_i(\xi, \nu),$$

and follows from the definition of the shape functions in section 4.5.

The linear change-of-coordinate functions $x(\xi, \nu)$ and $y(\xi, \nu)$ can therefore be explicitly defined as

$$T_K : \begin{array}{l} x = \sum_{i=1}^{6} x_i \widehat{\psi}_i(\xi, \nu) \\ y = \sum_{i=1}^{6} y_i \widehat{\psi}_i(\xi, \nu). \end{array} \tag{5.4}$$

In this equation, x_i and y_i denote the x and y values at node n_i in $\mathcal{K}$. It can be seen from (5.4) that the change-of-coordinate functions are continuously differentiable.

Now the equalities in (5.3) can be reexpressed in terms of the master element shape functions and the nodes of $\mathcal{K}$:

$$\begin{aligned} \frac{\partial \xi}{\partial x} &= \frac{1}{|\mathbf{J}|} \sum_{i=1}^{6} y_i \frac{\partial \widehat{\psi}_i}{\partial \nu} \\ \frac{\partial \xi}{\partial y} &= -\frac{1}{|\mathbf{J}|} \sum_{i=1}^{6} x_i \frac{\partial \widehat{\psi}_i}{\partial \nu} \\ \frac{\partial \nu}{\partial x} &= -\frac{1}{|\mathbf{J}|} \sum_{i=1}^{6} y_i \frac{\partial \widehat{\psi}_i}{\partial \xi} \\ \frac{\partial \nu}{\partial y} &= \frac{1}{|\mathbf{J}|} \sum_{i=1}^{6} x_i \frac{\partial \widehat{\psi}_i}{\partial \xi}. \end{aligned} \tag{5.5}$$

In the same manner, the determinant of the Jacobian matrix, $|\mathbf{J}|$ can be expressed as

$$|\mathbf{J}| = \Big\{\sum_{i=1}^{6} x_i \frac{\partial \widehat{\psi}_i}{\partial \xi}\Big\}\Big\{\sum_{i=1}^{6} y_i \frac{\partial \widehat{\psi}_i}{\partial \nu}\Big\} - \Big\{\sum_{i=1}^{6} x_i \frac{\partial \widehat{\psi}_i}{\partial \nu}\Big\}\Big\{\sum_{i=1}^{6} y_i \frac{\partial \widehat{\psi}_i}{\partial \xi}\Big\}. \tag{5.6}$$

By substituting the derivatives of the shape functions from (5.2) into this equation and remembering that the edge midpoints (x_2, y_2), (x_4, y_4), and (x_6, y_6) can be expressed as the means of the adjacent vertices, one can show that (5.6) can be expressed more simply as

$$|\mathbf{J}| = x_1(y_3 - y_5) + x_3(y_5 - y_1) + x_5(y_1 - y_3). \tag{5.7}$$

Note that this expression is only correct if the nodes are numbered counterclockwise, beginning at a vertex. If they are numbered clockwise beginning from a vertex, then the sign in (5.7) will change.

5.4 Computing $\mathbf{K}_0$

To construct $\mathbf{K}_0$, it is necessary to compute $\int \psi_i \psi_j$ for each $1 \leq i \leq n$ and v $1 \leq j \leq n$. As was mentioned before, this computation is performed separately for each element $\mathcal{K}$ and the computations themselves are done exclusively on the master element. Using $|\mathbf{J}|$, the integral

$$\int_{\mathcal{K}} \psi_i(x, y)\psi_j(x, y)dxdy$$

can be evaluated as

$$\int_{\widehat{\mathcal{K}}} \widehat{\psi}_i(\xi, \nu)\widehat{\psi}_j(\xi, \nu)|\mathbf{J}|d\xi d\nu. \tag{5.8}$$

It is more convenient to use a quadrature rule, which approximates the integral of a function as a weighted mean of the value of the function at a number of specified points in the domain, than it is to evaluate this integral directly. Since the integrand is a polynomial, a quadrature rule can be chosen which will evaluate the integral exactly. Since each $\widehat{\psi}_i$ is quadratic and $|\mathbf{J}|$ is a scalar, the integrand is a quartic polynomial. A quadrature rule which evaluates the integral of any quartic polynomial exactly will use six points and computes the integral of a quartic function $\widehat{g}$ as

$$\int_{\widehat{\mathcal{K}}} \widehat{g}(\xi, \nu) d\xi d\nu = \sum_{q=1}^{6} w_q \widehat{g}(\xi_q, \nu_q),$$

where the (ξ_q, ν_q) are the six quadrature points and the w_q are the corresponding weights. The quadrature points and weights for the master element are presented in table 5.1. Both this quadrature rule and the one which will be presented in section 5.5, as well as many others, can be found in [26].

ξ	ν	$\mathbf{w}$
0.81684757	0.09157621	0.10995174
0.09157621	0.81684757	0.10995174
0.09157621	0.09157621	0.10995174
0.10810302	0.44594849	0.22338159
0.44594849	0.10810302	0.22338159
0.44594849	0.44594849	0.22338159

Table 5.1: Quadrature points and weights for a quartic rule on the master element (taken from [26]).

With this quadrature rule and the expression in (5.8), $\mathbf{K}_0$ can now be constructed element by element. Assume that the triangulation contains t nodes (vertices and edge midpoints) $\{n_1, \ldots, n_t\}$. Assume that each element (triangle) $\mathcal{K}_e$ has six nodes $\{n_{e1}, \ldots, n_{e6}\}$, numbered counter-clockwise starting from a vertex. Denote by $|\mathbf{J}_e|$ the determinant of the Jacobian matrix of the transformation from $\widehat{\mathcal{K}}$ to $\mathcal{K}_e$. Then the algorithm for constructing $\mathbf{K}_0$ is as follows.

Algorithm for constructing $\mathbf{K}_0$

Initialize $\mathbf{K}$ to be a $t \times t$ zero matrix.

For each $\mathcal{K}_e$

  For each $i \in \{n_{e1}, n_{e2}, \ldots, n_{e6}\}$

    For each $j \in \{n_{e1}, n_{e2}, \ldots, n_{e6}\}$

      $\mathbf{K}_{ij} := \mathbf{K}_{ij} + |\mathbf{J}_e| \sum_{q=1}^{6} w_q \widehat{\psi}_i(\xi_q, \nu_q)\widehat{\psi}_j(\xi_q, \nu_q)$

    End

  End

End

$\mathbf{K}_0 = \mathbf{K}$

5.5 Computing $\mathbf{K}_1$

The method for computing $\mathbf{K}_1$ differs from that for computing $\mathbf{K}_0$ in only three details. Since the entries of $\mathbf{K}_1$ involve partial derivatives of the shape functions, the integrands will be cross-products of *linear* functions and hence the quadrature rule needs to be exact only for quadratic polynomials. Such a rule uses three, rather than six, quadrature points. Expressing the partial derivatives of the shape functions of an arbitrary element $\mathcal{K}$ exclusively in terms of the master element shape functions requires an application of the chain rule and the resulting expressions are somewhat more complicated than those for computing $\mathbf{K}_0$.

The only other difference is that the size of $\mathbf{K}_1$ is different from that of $\mathbf{K}_0$. Recall that if there are t nodes in the triangulation and m of those nodes are not on the boundary, then $\mathbf{K}_0$ will be $t \times t$ but $\mathbf{K}_1$ will be only $m \times t$. Since the number of boundary nodes is usually small compared to the total number of nodes, there is little loss of efficiency entailed in simply computing the $t \times t$ matrix

$$\int_\Omega \left(\frac{\partial}{\partial x}\psi_i \frac{\partial}{\partial x}\psi_j + \frac{\partial}{\partial y}\psi_i \frac{\partial}{\partial y}\psi_j\right)$$

and then obtaining $\mathbf{K}_1$ by deleting the rows of that matrix corresponding to boundary nodes. Since the final algorithm is identical in structure to that in the previous section, only the quadrature rule and the expression in the integrand will be explained in this section.

The main result of this section is expression of the quantities

$$\frac{\partial\widehat{\psi}_i}{\partial x} \text{ and } \frac{\partial\widehat{\psi}_i}{\partial y}$$

as functions of the nodes of $\mathcal{K}$ and of the ψ_i as follows. As a consequence of the chain rule,

$$\frac{\partial\widehat{\psi}_i}{\partial x} = \frac{\partial\widehat{\psi}_i}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial\widehat{\psi}_i}{\partial \nu}\frac{\partial \nu}{\partial x} \quad \text{and} \quad \frac{\partial\widehat{\psi}_i}{\partial y} = \frac{\partial\widehat{\psi}_i}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial\widehat{\psi}_i}{\partial \nu}\frac{\partial \nu}{\partial y}.$$

By combining these equalities with the equalities in (5.5), one can express these partial derivatives as

$$\begin{aligned} \frac{\partial\widehat{\psi}_i}{\partial x} &= f_{ix} = \frac{1}{|\mathbf{J}|}\left\{ \frac{\partial\widehat{\psi}_i}{\partial \xi}\sum_{k=1}^{6} y_k \frac{\partial\widehat{\psi}_k}{\partial \nu} - \frac{\partial\widehat{\psi}_i}{\partial \nu}\sum_{k=1}^{6} y_k \frac{\partial\widehat{\psi}_k}{\partial \xi}\right\} \\ \frac{\partial\widehat{\psi}_i}{\partial y} &= f_{iy} = \frac{1}{|\mathbf{J}|}\left\{ -\frac{\partial\widehat{\psi}_i}{\partial \xi}\sum_{k=1}^{6} x_k \frac{\partial\widehat{\psi}_k}{\partial \nu} + \frac{\partial\widehat{\psi}_i}{\partial \nu}\sum_{k=1}^{6} x_k \frac{\partial\widehat{\psi}_k}{\partial \xi}\right\}. \end{aligned} \tag{5.9}$$

A quadrature rule which is exact for quadratic functions on the master element is presented in table 5.2. This rule only requires the function to be evaluated at the midpoints of each edge.

ξ	ν	**w**
0	0.5	1/6
0.5	0	1/6
0.5	0.5	1/6

Table 5.2: Quadrature points and weights for a quadratic rule on the master element (taken from [26]).

Using this quadrature rule and the functions in (5.9), the final result required for constructing $\mathbf{K}_1$ is

$$\int_{\mathcal{K}} \left(\frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial x} + \frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial y}\right)dxdy =$$

$$\sum_{q=1}^{3} w_q|\mathbf{J}|(f_{ix}(\xi_q,\nu_q)f_{jx}(\xi_q,\nu_q) + f_{iy}(\xi_q,\nu_q)f_{jy}(\xi_q,\nu_q)).$$

5.6 A Note on Implementation

Many mathematical software packages, including S-plus and MATLAB, include a Delaunay triangulation routine. Given an $p \times 2$ matrix $\mathbf{X}$ whose rows represent p points in the plane, such a routine will generate a Delaunay triangulation of the convex hull of the points in $\mathbf{X}$. The properties of Delaunay triangulations will be discussed in more detail in section 6.5.1. While there are usually many possible triangulations of a given set of points, a Delaunay triangulation is considered to be close to optimal for implementing the finite element method. A triangulation is usually represented as an $e \times 3$ matrix $\mathbf{T}$, each of whose rows represent the indices of the vertices of one triangle in counter-clockwise order. Sometimes the order will be clockwise but, in all cases, the order will be the same for every row. The indices refer to rows of $\mathbf{X}$. For example, if $[3, 12, 5]$ is a row of $\mathbf{T}$, then the points in the third, twelfth and fifth rows of $\mathbf{X}$ are vertices of one triangle.

One way in which to generate nodes for the finite-element L-spline is to triangulate a set of points which includes the data points. It is important to make sure that the data points are not on (or too near) the boundary of the convex hull of the set of points, otherwise they will automatically receive a value of zero in the finite-element solution. Then, beginning with the matrices $\mathbf{X}$ and $\mathbf{T}$ as already defined, augment $\mathbf{X}$ by adding a row for the midpoint of each edge in the triangulation. Also, add three columns to $\mathbf{T}$ containing the rows of the midpoints of the three edges. In the augmented version of $\mathbf{T}$, the ijth element will contain the row number of the jth node of the ith element (triangle).

The computations involved in implementing the algorithm in section 5.4 can be performed efficiently using appropriate subsets of the (augmented) matrices $\mathbf{X}$ and $\mathbf{T}$. Denote by $\mathbf{A}(i,j)$ the ijth element of the matrix $\mathbf{A}$. Similarly, let the ith row and jth column of $\mathbf{A}$ be denoted $\mathbf{A}(i,\cdot)$ and $\mathbf{A}(\cdot,j)$, respectively. Using the expression for $|\mathbf{J}|$ in (5.7), the vector $\mathbf{j}$ of determinants of the Jacobians for each element is simply

$$\mathbf{j} = \mathbf{X}(\cdot,1).*(\mathbf{T}(\cdot,3)-\mathbf{T}(\cdot,5))+\mathbf{X}(\cdot,3).*(\mathbf{T}(\cdot,5)-\mathbf{T}(\cdot,1))+\mathbf{X}(\cdot,5).*(\mathbf{T}(\cdot,1)-\mathbf{T}(\cdot,3)),$$

where the operator ".*" represents element-wise multiplication of two same-sized matrices or vectors. It will be useful to define the similar operator "$.^2$" to mean the element-wise square of a matrix. For example, $\mathbf{X}.^2$ is the same size as $\mathbf{X}$ but has ijth element equal to $\mathbf{X}(i,j)^2$.

Now define the 6×6 matrix $\widehat{\psi}$ so that the ith row of $\widehat{\psi}$ consists of the coefficients of 1, ξ, ν, ξ^2, $\xi\nu$ and ν^2 in the master element shape function $\widehat{\psi}_i$. For example, $\widehat{\psi}_3 = 2\xi^2 - \xi$ so the third row of $\widehat{\psi}$ is $[0,-1,0,2,0,0]$. The vector $\widehat{\psi}*[1,\xi,\nu,\xi^2,\xi\nu,\ nu^2]^t$ contains the values of each of the $\widehat{\psi}_i$ evaluated at (ξ,ν). The vector used in this operation is so useful that it merits being given a name. Define the vector $\mathbf{v}ec(\xi,\nu)$ by

$$\mathbf{v}ec(\xi,\nu) = [1,\xi,\nu,\xi^2,\xi\nu,\nu^2]^t.$$

The algorithm for computing $\mathbf{K}_0$ requires evaluating each $\widehat{\psi}_i$ at each of the six quadrature points (ξ_j,ν_j) (for $j=1,\ldots,6$). This can be done in one step by constructing a 6×6 matrix $\mathbf{Q}$ whose jth column is $\mathbf{v}ec(\xi_j,\nu_j)$. Then the matrix $\widehat{\psi}*\mathbf{Q}$ has the value $\psi_i(\xi_j,\nu_j)$ as its ijth entry.

The integrand in the construction of $\mathbf{K}_0$ can be evaluated for each pair (i,j) in one step as follows. Let $\mathbf{Y}$ be the 36×6 matrix whose first six rows are each equal to the first row of $\widehat{\psi}*\mathbf{Q}$, whose second six rows are each equal to the second row of $\widehat{\psi}*\mathbf{Q}$, and so on. Let $\mathbf{Z}$ be the 36×6 matrix obtained by stacking six copies of $\widehat{\psi}*\mathbf{Q}$ on top of each other. Define w to be the vector $[w_1,\ldots,w_6]$ of quadrature weights, and compute the vector $(\mathbf{Y}.*\mathbf{Z})*w$. Reshaping this vector into a 6×6 matrix (whose first column is the first six entries of $(\mathbf{Y}.*\mathbf{Z})*w$, and so on) yields a matrix $\mathbf{H}$ whose

ijth entry is precisely equal to

$$\sum_{k=1}^{6} \widehat{\psi}_i(\xi_k, \nu_k)\widehat{\psi}_j(\xi_k, \nu_k).$$

Both MATLAB and S-plus support the construction $\mathbf{A}(\cdot, [c_1, \ldots, c_k])$, which represents a matrix with k columns whose jth column is the c_jth column of $\mathbf{A}$. One can perform the same construction to concatenate rows of $\mathbf{A}$ into a new matrix ($\mathbf{A}([r_1, \ldots, r_l], \cdot)$), or even form subsets of the rows and columns using $\mathbf{A}([r_1, \ldots, r_l], [c_1, \ldots, c_k])$. This last matrix is $k \times l$ with ijth entry equal to $\mathbf{A}(r_i, c_j)$. On the other hand, if $\mathbf{B}$ is an arbitrary $k \times l$ matrix, the assignment

$$\mathbf{A}([r_1, \ldots, r_l], [c_1, \ldots, c_k]) = B$$

changes the $r_i c_j$th entry of $\mathbf{A}$, $\mathbf{A}(r_i, c_j)$, to the value $\mathbf{B}(i, j)$. Using this construction, the algorithm of section 5.4 can be reformulated as

5.6.1 New Algorithm for Computing $\mathbf{K}_0$

Initialize $\mathbf{K}_0$ to be a $t \times t$ zero matrix.

Compute the augmented matrices $\mathbf{X}$ and $\mathbf{T}$.

Compute the Jacobian vector $\mathbf{j}$.

Compute the matrix $\mathbf{H}$.

For $i = 1$ to $length(\mathbf{j})$

$\quad\mathbf{K}_0([\mathbf{T}(i, \cdot), \mathbf{T}(i, \cdot)] := \mathbf{K}_0([\mathbf{T}(i, \cdot), \mathbf{T}(i, \cdot)] + \mathbf{j}(i)\mathbf{H}$.

End

This version of the algorithm replaces three nested loops with one loop and is far more efficient that the more naive algorithm of section 5.4. An efficient algorithm for constructing $\mathbf{K}_1$, which also loops only once through the elements, can be formulated in a similar manner.

5.7 Solving the Linear System

Recall from section 4.8 that solving the feL-spline problem involves finding vectors $\mathbf{u}$ and $\mathbf{s}$ which satisfy

$$\begin{aligned} \mathbf{K}_1\mathbf{s} + \lambda\mathbf{P}\mathbf{u} &= \lambda z \\ \mathbf{K}_0\mathbf{s} - \mathbf{K}_1{}^t\mathbf{u} &= 0. \end{aligned} \tag{5.10}$$

The vector of interest is actually $\mathbf{u}$.

There are two ways in which one might solve this system. The first method involves two steps: first use the second equation to express $\mathbf{s}$ as $\mathbf{K}_0{}^{-1}\mathbf{K}_1{}^t\mathbf{u}$, then substitute this expression into the first equation to obtain $\mathbf{K}_1\mathbf{K}_0{}^{-1}\mathbf{K}_1{}^t\mathbf{u} + \lambda\mathbf{P}\mathbf{u} = \lambda\mathbf{z}$. Then

$$\mathbf{u} = (\mathbf{K}_1\mathbf{K}_0{}^{-1}\mathbf{K}_1{}^t + \lambda\mathbf{P})^{-1}\lambda\mathbf{z}.$$

Alternatively, the problem can be solved in one step by re-expressing 5.10 in block matrix form as

$$\begin{bmatrix} \mathbf{K}_1 & \lambda\mathbf{P} \\ \mathbf{K}_0 & \mathbf{K}_1{}^t \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \lambda\mathbf{z} \\ 0 \end{bmatrix}. \tag{5.11}$$

This expression can be solved for both $\mathbf{u}$ and $\mathbf{s}$ simultaneously by inverting the action of the large matrix.

At first glance, it might appear that the first method would be faster. It involves inverting two matrices of order t, while the second involves inverting one matrix of order $2t$. Since the inversion of a $t \times t$ is, in general, an order $\mathcal{O}(t^3)$ operation, one might expect that the second method would take approximately four times as long as the first. In fact, the second method is much *more* efficient than the first. For example, doing the calculation for the male face example in chapter 6 using the first method on a Sun Ultra 60 machine took about 25 minutes, while the same operation using the second method took about 26 seconds.

The reason for this (perhaps surprising) difference lies in the sparsity of the matrices involved. Each of the matrices $\mathbf{K}_0$, $\mathbf{K}_1$ and $\mathbf{P}$ is very sparse. In the male face example, more than 99% of the entries were zero in each matrix. 99.77% of the entries in the block matrix of expression (5.11) were zero. The inverse of $\mathbf{K}_0$, on the other hand, was full; none of its 3481^2 entries was zero. Therefore the second step of the first method involves inverting the action of a full matrix. This is a much more computationally intensive operation than inverting a very sparse matrix. In addition, the first step of the first method involves actually computing the inverse of $\mathbf{K}_0$. This is more complex than simply inverting the action of a matrix on a single vector, an operation which can be accomplished by row reduction or by back substitution with a sparse matrix.

The linear system in (5.10) should always be solved using the block matrix method in (5.11), rather than by the two-step method.

Chapter 6

An Application of the Bivariate L-Spline to Image Registration

6.1 The Data

Digital images of the human face are an example of bivariate functional data with landmarks. With grayscale being a function ranging over the interval from zero (pure black) to one (pure white), a two-dimensional black and white image can be thought of as a real-valued function of $\mathbb{R}^2$. An RGB image, in which a colour is represented as the triple of red, green, and blue intensity values, can be treated as a vector valued function of $\mathbb{R}^2$. The human face also contains a set of distinguished landmarks which clearly occur in different relative locations in different faces. Possible choices of landmarks include the eyes, nose, mouth, and boundary of the face. Registration of these landmarks can not only yield an estimate of an 'average', or mean, face but can also allow one to explore the extent to which shape and position of the features contribute to the variation between different faces. Such a question might well be of interest to a psychologist attempting to understand how humans recognize each other.

Our modern, image-rich world is teeming with images of faces. One could presumably download any number of digital face images from the world wide web and, with the help of a scanner, could obtain many more such images from photographs and magazines. In spite of this wide availability of digital images of faces, however, obtaining suitable data for face registration proves to be considerably more difficult than one might expect. The problem lies in the fact that, although all *faces* are structurally comparable, the same is not true for all *images* of faces.

The assumption that each image contains analogous features implies that each face is viewed from the same angle in space. It is clear, for example, that an image of a face viewed in profile and another of a face viewed head-on differ by much more than just the location of the eyes and mouth. One would contain two eyes, for example, while the other would contain only one. These two images are structurally very different and cannot be included in the same sample of functional data. Even if two faces are viewed from approximately head on, a difference of a few degrees in the horizontal plane could result in the left ear being invisible in the first but both ears being visible in the second. A few degrees more and the left ear would be visible while the right is not. To make matters even more complicated, one can imagine two faces viewed head on in which one person has long hair and the other is bald. In such a situation, the two faces cannot be registered simply by warping the domain since no amount of warping will transform a full head of hair into a bald head.

The data used in this analysis were obtained from Dr. Nikolaus Troje, currently a visiting scholar in the Queen's University Department of Psychology, and have been pre-processed to have approximately the same structural content. Figure 6.1 shows an example of one of the images. The faces are all viewed in the classic 'mug-shot' pose and have been cropped to remove all of the head above the hairline and all of the body below the neck. The expressions on the different faces are all approximately constant, with the facial muscles relaxed and the eyes looking directly ahead towards the viewer. In addition, the background behind each face has been blacked out, the images have been scaled to approximately the same size, and have all been embedded in the center of a 256×256 pixel colour JPEG image file. Of the 201 faces in the database, approximately half are male.

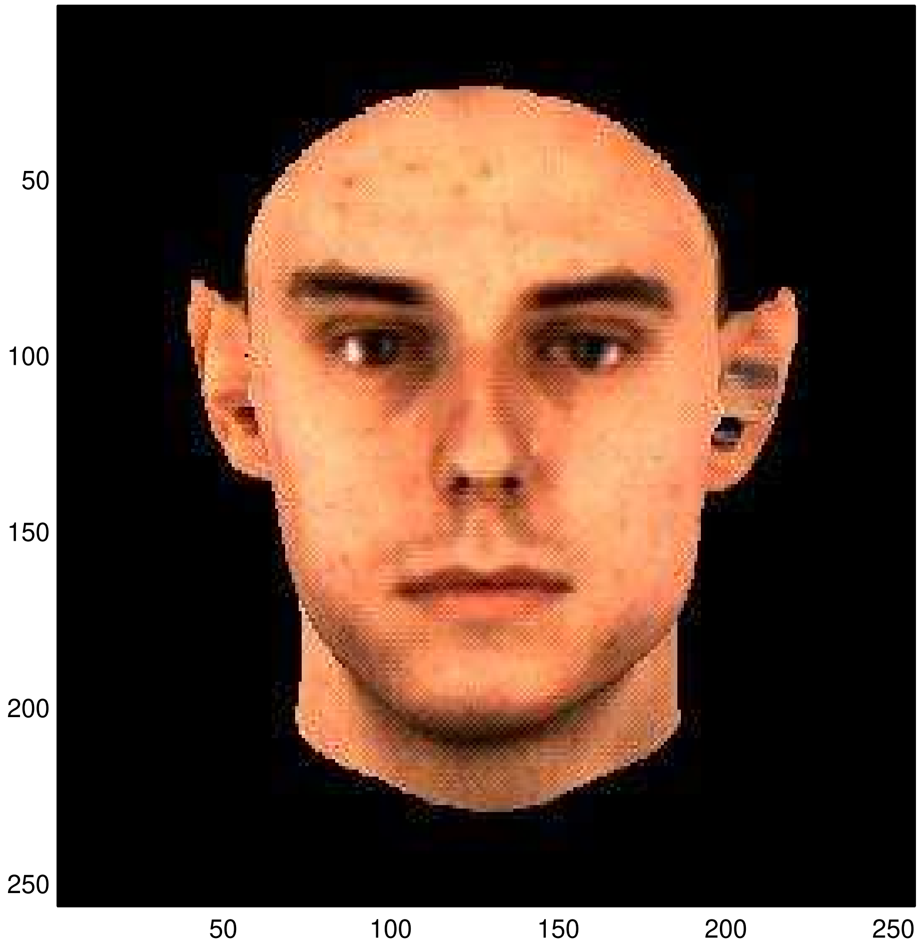

Figure 6.1: A sample male face

Since all of the images have the same shape and size, the registration functions should not change the overall shape or size of the images being registered. For this reason, all of the FeL-splines used to estimate the registration functions are constrained to equal zero on the boundary of the image domain Ω.

6.2 The Landmarks

Some features in the faces which might be considered as landmarks are the eyes, eyebrows, nose, mouth, and boundary; all of these but the eyebrows were registered. Since they were often partly obscured by the hair, no attempt was made to register the ears. The neck also was ignored, as it was decided that the neck is not part of the face. Choosing which features to use as landmarks is not a trivial task and the choices made here are by no means unique. In addition, once the landmarks were chosen, the relatively poor resolution of the images meant that the landmark positions could only be located approximately. The best guess at the location of the respective landmarks was obtained by clicking the mouse on the image while it was displayed on a computer screen.

6.2.1 The Eyes

An ideal strategy for registering the eyes would be to define the upper and lower eyelids of each eye to be two one-dimensional landmarks. Unfortunately the lighting and resolution in the images themselves were such as to make the exact location of the eyelids and corners of the eyes somewhat ambiguous. Due to this ambiguity, it was decided to use the best guess at the corners of the eyes, along with the points at the top of the upper and the bottom of the lower eyelids, to define the eyes. These four point landmarks should result in registering all of the eyes to the same position on the face and should also remove the variation arising from different people opening their eyes more or less than others. Since the different faces showed some variation in the location of the pupils, the point at the center of each pupil was also used as a landmark.

6.2.2 The Nose

Eight point landmarks were used to define the nose. These points were the leftmost point, the rightmost, the lowest, and the highest point of the nose itself along with the leftmost and rightmost points of each nostril. It was found to be especially difficult to unambiguously locate the top of the nose between the eyes. Nonetheless, since there was perceived to be a certain amount of variance in the location of this point in the sample of faces, this landmark was included.

6.2.3 The Mouth

Five point landmarks were chosen to define the mouth. These points were located at the two extreme corners of the mouth as well as the top of the upper lip, the center of the line between the upper and lower lips, and the bottom of the lower lip. The latter three points were chosen to lie on the vertical line through the center of the nose. Since there was considerable variance in the thickness and shape of the lips, it might have been better to use the outline of the mouth as a one-dimensional curve landmark.

6.2.4 The Boundary: a One-Dimensional Landmark

In order to meaningfully register the boundaries of two faces, corresponding points on the boundary must be mapped to each other. It would be unacceptable, for example, to superimpose one boundary exactly onto the other in such a way as to map the chin of one face onto the forehead or the ear of another. This implies that there must at least one zero-dimensional landmark identified on each boundary. The approach adopted in this analysis was to identify the points of intersection of a vertical line through the center of the nose with the boundary. This specified one point each at the top of the forehead and at the bottom of the chin.

Using a sequence of other points arbitrarily chosen along the entire boundary, the two arcs between these points were estimated and parameterized by arc length. A set of fifty evenly spaced points were then selected on each arc as further landmarks on the boundary. One could choose any number of intermediate points on the arcs, but fifty seemed to provide an adequate registration. The decision that fifty was adequate was based on the observation that the mean of the registered faces appeared to have a well-defined boundary, suggesting that the individual image boundaries had been properly registered to the global mean.

Using two arcs to define the boundary effectively defined the top point of the forehead and the bottom point of the chin to be point landmarks embedded within the one-dimensional boundary. The choice to include two point landmarks, rather than only one, on the boundary is important because the human face is not perfectly symmetric. Thus if the boundary were parameterized as a closed curve beginning and ending at, for example, the bottom point of the chin, then the midpoint of that curve would not generally correspond to the top point of the forehead. A registration based on specifying only the chin, therefore, would not necessarily result in the forehead being mapped exactly onto the top of the face.

Another way of expressing this problem is that, although the bottom point of the chin is a landmark which is identifiable on any face, the antipodal point on the boundary of a given face does not, in general, correspond structurally to the antipodal point on the boundary of another face. The antipodal point here refers to the point on the closed curve which is equidistant from the chin in either direction. In fact, the faces were initially registered using only the bottom of the chin as a single distinguished point on the boundary and the resulting deformed images were judged to be unsatisfactory. It was assumed for the purpose of the analysis that the midpoint of the left (respectively right) arc between the top of the forehead and the bottom of the chin on a given face does correspond to the analogous point on any other face. Although this is likely not strictly true, this assumption leads to registered faces which appear satisfactory to the eye.

Thus 25 significant point landmarks were chosen, including the chin and forehead. In addition, a number of points along the arc of the boundary were also chosen to define the boundary. The boundary was subsequently reparameterized by arc length and 100 equidistant points (50 on either side) were registered. The original points chosen to define the boundary, with the exception of the chin and forehead, were discarded after the reparametrization. Figure 6.2 displays the landmarks which were chosen to define one face. One should note that the points along the boundary are those which were initially chosen to define the boundary and not those which were produced by the reparametrization. Figure 6.3 displays the derived set of landmarks that were actually registered.

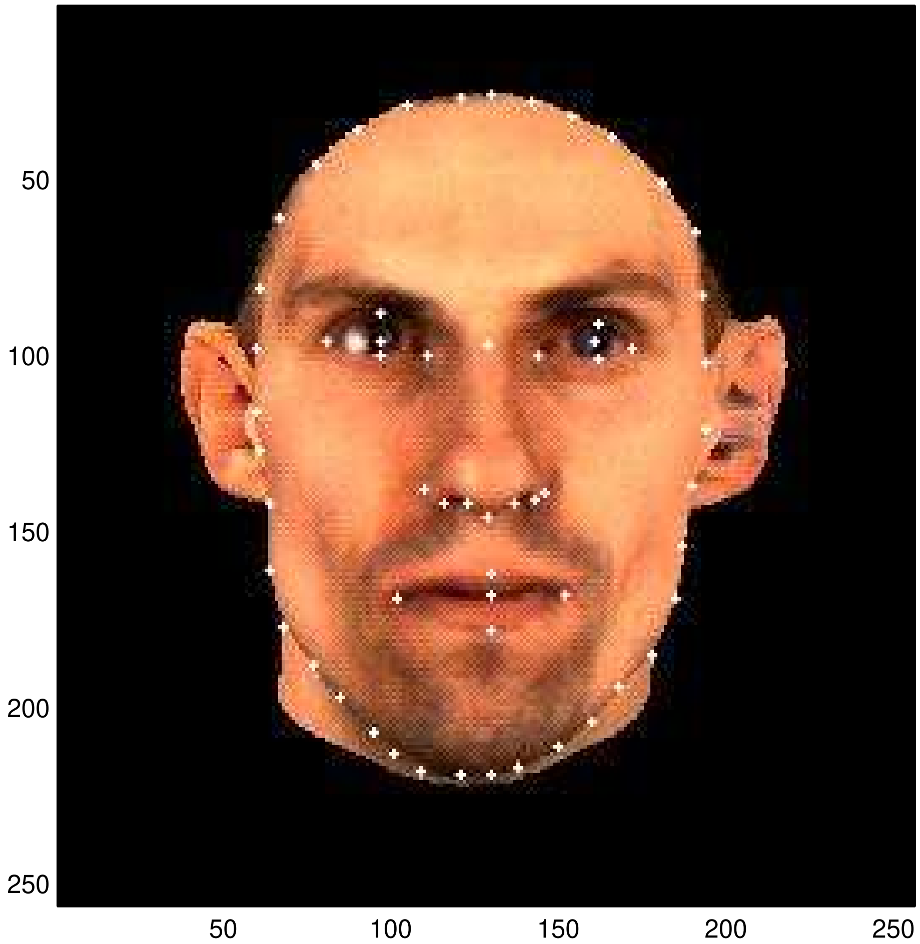

Figure 6.2: A sample of selected landmarks.

6.2.5 Locating the Landmarks on a Given Face

Of the 201 faces in the database obtained from Dr. Troje, 20 female faces and 20 male faces were selected and were registered first as two separate samples. The landmarks were located on each face by clicking on the image with a mouse. The two arcs of the boundary were defined by selecting two sequences of points around the boundary and then smoothing each coordinate of each sequence separately. The resulting curves were then reparameterized by arc length to produce the curves $b_l(t) = [x_l(t), y_l(t)]$ and $b_r(t) = [x_r(t), y_r(t)]$ for $0 \leq t \leq 1$. With the lowest point on the chin and the highest point on the boundary of the forehead defined as b_1 and b_2, respectively, these

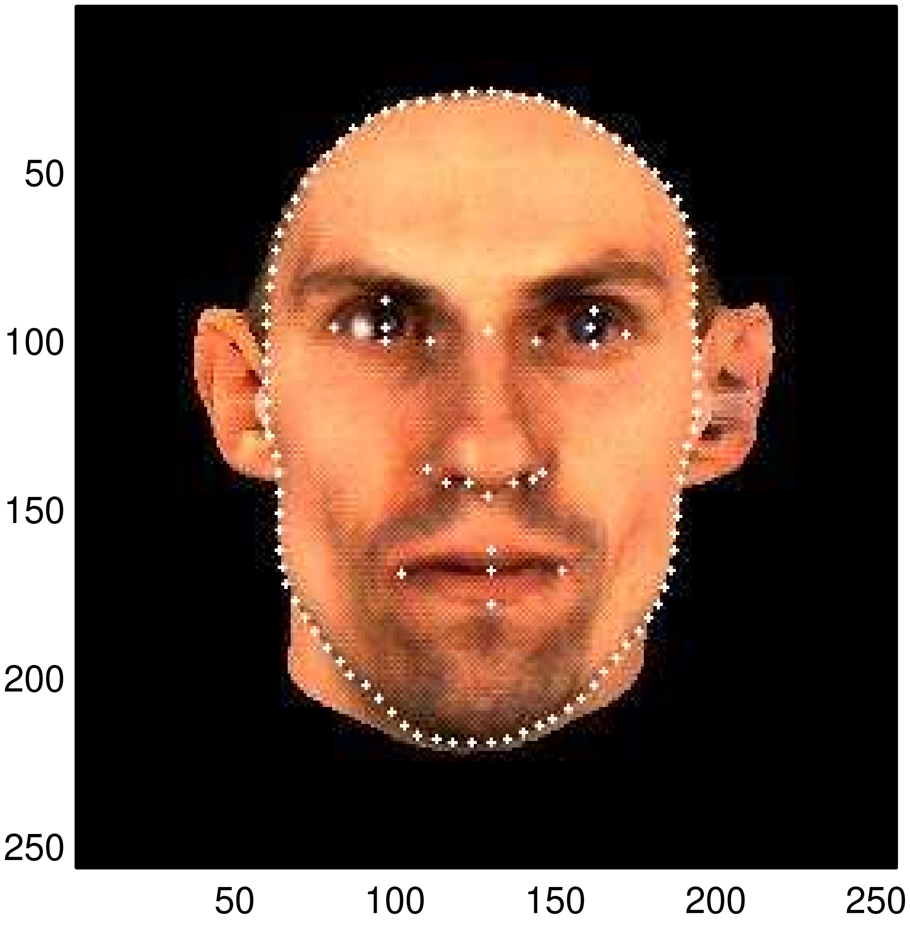

Figure 6.3: A sample of final landmarks.

curves were defined such that $b_r(0) = b_l(1) = b_1$ and $b_r(1) = b_l(0) = b_2$. Thus, for a given face, b_r was the right-hand boundary beginning at the chin and ending at the forehead, b_l was the left-hand boundary beginning at the forehead and ending at the chin, and the union of b_l and b_r was the entire boundary.

6.3 Computing the Registered Faces

This section will discuss the procedure necessary for registering a set of 20 faces to their mean. Denote the set of 20 images by $\{F_1, \ldots, F_{20}\}$, with the k^{th} image F_k being a 256-by-256-by-3 array of integers in the range from 0 to 255. These correspond to

RBG jpeg files which are each 256-by-256 arrays of 3-tuples. Each 3-tuple denotes the colour at a given pixel by the respective intensities of the red, green, and blue components. Each image F_k is considered to be a sample of a continuously supported, vector-valued function on the domain $[1, 256] \times [1, 256] \subset \mathbb{R}^2$; it is useful to denote that unknown function as f_k.

Before registering the 20 faces, it was necessary to estimate the mean location of each landmark. This was done by placing the 121 point landmarks for each face into a 121×2 landmark matrix $\mathbf{L}_k$, for $1 \leq k \leq 20$, and defining the mean $\bar{\mathbf{L}} = (20)^{-1} \sum_{k=1}^{20} \mathbf{L}_k$ to be the set of landmark locations of the mean face. Note that this means that the pair $[\mathbf{L}_k(i,1), \mathbf{L}_k(i,2)]$ on the i^{th} row of $\mathbf{L}_k$ represents the location of the i^{th} landmark in the image F_k (or, equivalently, in the function f_k).

The goal of the registration was to estimate a smooth deformation function $d_k(x,y) = (d_{kx}(x,y), d_{ky}(x,y))$ for each $1 \leq k \leq 20$ such that, for each $1 \leq i \leq 121$,

$$\Big[\mathbf{L}(i,1) - d_{kx}(\mathbf{L}(i,1), \mathbf{L}(i,2)), \mathbf{L}(i,2) - d_{ky}(\mathbf{L}(i,1), \mathbf{L}(i,2))\Big] = [\bar{\mathbf{L}}_k(i,1), \bar{\mathbf{L}}_k(i,2)].$$

If we extend the functional notation $d_k(x,y)$ to all $n \times 2$ matrices $\mathbf{X}$ by letting $d_k(\mathbf{X})$ be the $n \times 2$ matrix whose i^{th} row is obtained by applying d_k to the i^{th} row of $\mathbf{X}$, then this last equality can be expressed much more simply as

$$\mathbf{L}_k - d_k(\mathbf{L}_k) = \bar{\mathbf{L}}, \tag{6.1}$$

for all $1 \leq k \leq 20$.

The function $d_k = (d_{kx}, d_{ky})$ was estimated by using the finite element L-spline smoother penalizing the Laplacian. The set of argument values used to estimate each of d_{kx} and d_{ky} was the values in the rows of $\bar{\mathbf{L}}$, and the function values associated with these arguments were the first column of $\mathbf{L}_k - \bar{\mathbf{L}}$ for d_{kx} and the second column of $\mathbf{L}_k - \bar{\mathbf{L}}$ for d_{ky}, respectively. The resulting function d_k is such that $d_k(\mathbf{L}_k)$ is approximately equal to $\mathbf{L}_k - \bar{\mathbf{L}}$ and hence it can be seen by substitution that the expression (6.1) is approximately true.

Figure 6.4 displays an example of a face before and after it is warped.

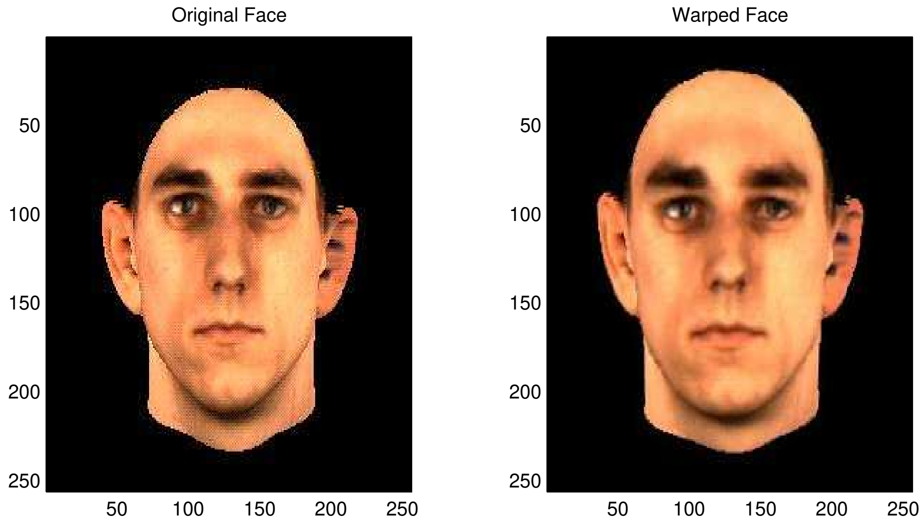

Figure 6.4: An example of a warped face

6.4 Results

Three separate samples were registered in order to estimate mean faces: 20 male faces, 20 female faces, and the male and female faces combined into a sample of 40. In each of the cases, the mean after registration appears substantially better than the mean before registration. The results are displayed in figures 6.5, 6.6, and 6.7. It is interesting to note that, at least to the author of this thesis, the mean of the registered male faces looks distinctly male while the mean obtained from the female faces looks distinctly female.

The best results appear to be in the boundary and the nose area, both of which appear very crisp and sharp in each of the post-registration means. The eyebrows, on the other hand, did not contribute landmarks to the registration functions and consequently were not registered at all and in none of the three examples do the eyebrows appear to be significantly better defined after registration. This would strongly suggest that the shape and position of the eyebrows are subject to a lot of variation, even when the variation attributable to the other major features of the face is removed.

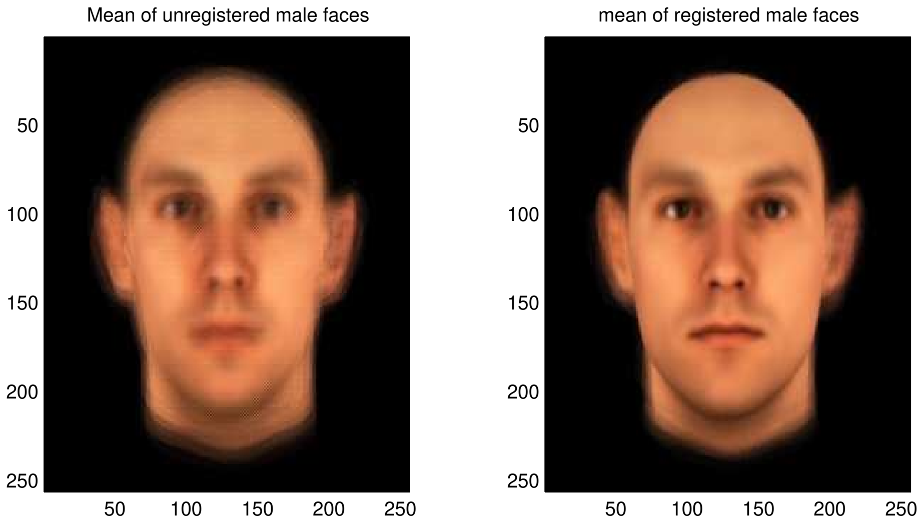

Figure 6.5: Male faces: unregistered and registered means

The mouth, although certainly substantially better estimated after registration than before, is not as distinct as one might hope in the post-registration means. This might be partially a consequence of the difficulty encountered in precisely locating the landmarks which were chosen to define the mouth. The two corners of the mouth, in particular, proved surprisingly difficult to identify.

The misregistration of the mouth is probably largely due to the fact that the mouth was defined by point landmarks when it should really be thought of as a set of three curves. Most importantly, there appears to be a great deal of variation present in the shape of the upper and lower lips and thus the four points chosen to define the outer boundary of the mouth were insufficient to properly register them. Perhaps a better scheme would be to retain those point landmarks but to embed them in a closed curve defining the entire boundary. This would require parameterizing four curves: namely the left and right sides of both the upper and lower boundaries of the lips.

Many of the methods of univariable functional data analysis should, in principle, be applicable to registered bivariable data. By performing a principal components analysis on the registered faces, for example, one could compute a set of *eigenfaces*

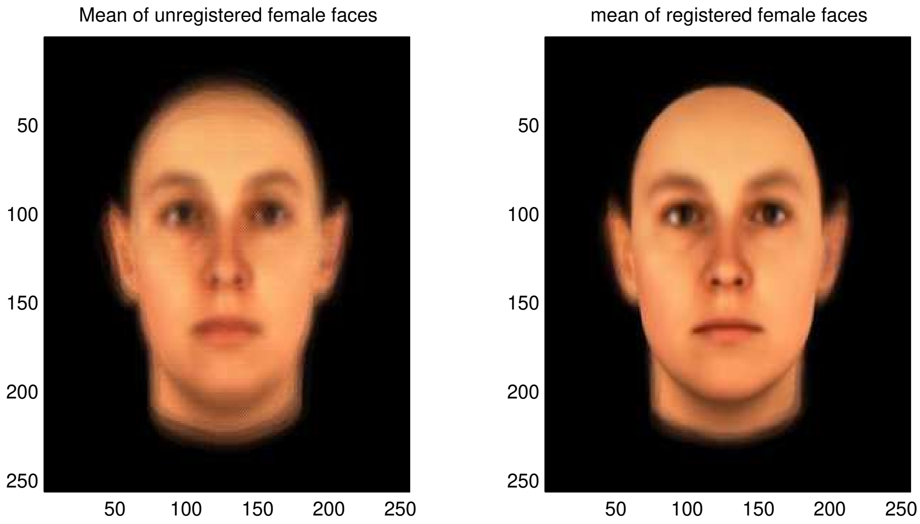

Figure 6.6: Female faces: unregistered and registered means

and use them to explore the structure of the range variation present in the faces. In a similiar manner, one could use a set of *eigenwarps* to study the variation in domain (shape variation). In addition to generalizing univariable methods to image data, perhaps the functional data analysis community may even develop new tools specific to the study of multivariable data. In any event, an effective registration technique is a necessary first step to the analysis of most functional data.

6.5 The General Registration Procedure

This section describes in detail each of the steps involved in producing a registered image. Although the procedure will be described in terms of one of the faces used in this chapter, the steps involved would be similiar for any bivariable registration problem. There are five steps involved in registering a set of images, assuming that the source and target landmark locations have already been determined. First, a domain Ω must be specified and triangulated. Next, the two deformation functions must be computed. These deformation functions must then be evaluated at each pixel in the domain. The original image is then deformed by changing the location of

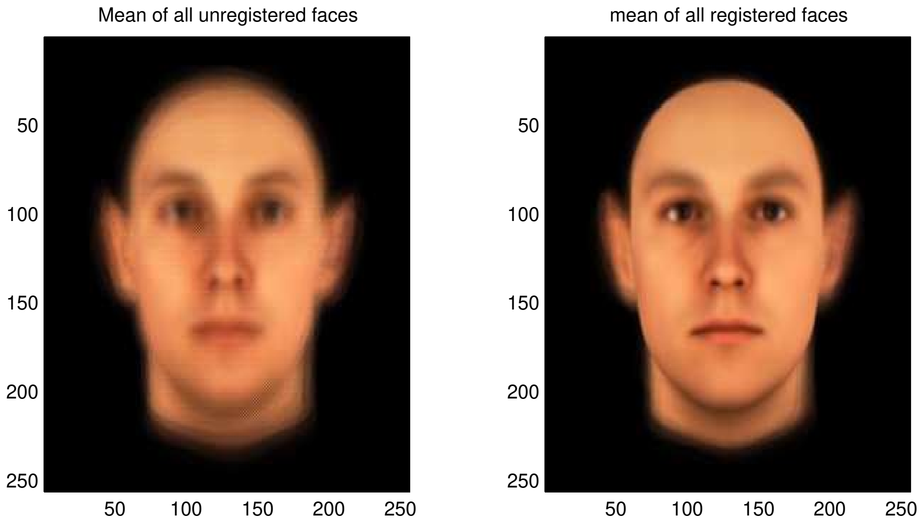

Figure 6.7: All faces: unregistered and registered means

each pixel. The registered image is then produced by estimating the intensity values of the deformed image at the original pixel locations. All of the computations for the examples in this chapter were performed on a Sun Ultra-60 workstation.

6.5.1 Triangulation of Ω

Once the domain Ω has been defined, it is necessary to choose a set of vertices for the triangulation of Ω. The local accuracy of a finite element estimate at a given point depends on the shape and size of the triangle covering that point. Much research has been done on exactly how the accuracy varies with the shape and size of the elements, but the results are both too numerous and too complex to review here. The interested reader should refer to the article "Principles for Design of Finite-Element Meshes", by Melosh and Utku [36], as well as to the many references contained therein. The finite element estimate will, in general, be better when the difference between the lengths of the largest and smallest sides of any given triangles is not too large. The local accuracy will also be greater for smaller triangles. Hence, one should avoid long, pointy triangles and one should try to make the triangles as small as is computationally reasonable. To ensure that the accuracy is relatively uniform over the domain, one should avoid having some triangles much larger or smaller than others.

Given a set $\mathcal{P}$ of at least four non-collinear points, there will almost always be multiple ways of triangulating the convex hull of $\mathcal{P}$ using the points in $\mathcal{P}$ as vertices. A Delaunay triangulation of $\mathcal{P}$ is a (not necessarily unique) triangulation whose set of vertices is $\mathcal{P}$ and in which the circle passing through the vertices of any given triangle contains no other vertices. Delaunay triangulations tend to be close to optimal, in terms of producing the most accurate finite element estimate. Furthermore, there exists a fast algorithm for producing a Delaunay triangulation from any set of non-collinear points.

A good strategy for triangulating Ω is to select a fairly uniformly distributed set of points, in the sense that the distance between an arbitrary point and its nearest neighbour should be roughly constant, and then to compute a Delaunay triangulation. Overall, the estimate will be better when more triangles are used, but more triangles means more nodes and hence a greater computational load in setting up and solving the linear system. If the computations involved in setting up and solving the linear system take too long, one should consider either using fewer nodes or switching to a faster machine.

Since the face images were recorded as a 256×256 pixel grid, it was natural to take Ω to be the square $[1, 256] \times [1, 256]$. Each axis was divided into twenty-nine regular sub-intervals, each of the same size. The endpoints of these sub-intervals defined thirty values on each axis. Placing one node at each combination of the x and y values created nine hundred equally spaced nodes on a regular lattice. Since the data points were not located at nodes, the data points were inserted and, for each data point, the nearest lattice node was removed. The purpose of removing the lattice nodes was to avoid creating triangles with very short sides.

This set of vertices was then triangulated using the MATLAB Delaunay triangulation procedure. Following this triangulation step, a new node was created at the midpoint of each edge. The end result was 3481 nodes. The nodal locations were stored as a 3481×2 matrix. For the purpose of computing the matrix $\mathbf{K}_1$, it was also necessary to record which nodes were on the boundary of Ω. This information was stored in an indicator vector of length 3481. For the male face example, the process of adding the midpoint nodes and labelling the exterior nodes took 4.5 seconds and the triangulation of the original nine hundred lattice points took 0.04 seconds.

6.5.2 Computation of Deformation Functions

Let the location of the ith landmark in the source image be (x_{si}, y_{si}) and the location of the ith landmark in the target image be (x_{ti}, y_{ti}). Then the observed value of the x coordinate deformation function d_x at (x_{si}, y_{si}) is $x_{si} - x_{ti}$. Similarly, the observed value of the y coordinate deformation function d_y at (x_{si}, y_{si}) is $y_{si} - y_{ti}$. These values must be placed in the vector $\mathbf{z}$ in the linear system in (5.11) and the matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ must be computed from the coordinates of the nodes.

Since the matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ are the same for d_x and d_y, both coordinate deformation functions can be calculated simultaneously by replacing the block matrix

$$\begin{bmatrix} \lambda \mathbf{z} \\ 0 \end{bmatrix} \quad \text{by} \quad \begin{bmatrix} \lambda \mathbf{z}_x & \lambda \mathbf{z}_y \\ 0 & 0 \end{bmatrix} \tag{6.2}$$

in the block matrix expression (5.11). The vector $\mathbf{z}_x$ contains the observed differences between the source and target in the x coordinate and the vector $\mathbf{z}_y$ contains the observed differences in the y coordinate. In the example where twenty faces were registered to the same mean, a block matrix with forty columns was substituted in (6.2) to compute all forty deformation functions simultaneously. This saves computational time both in computing the matrices $\mathbf{K}_0$ and $\mathbf{K}_1$ and in solving the linear system.

Each deformation function is defined by a set of three objects: the nodes, the triangulation, and the value of the function at each node (nodal values). The set of forty deformation functions for twenty faces was defined by one set of nodes, one triangulation, and forty vectors of nodal values.

In the example of registering the male faces, the computation of $\mathbf{K}_0$ took 13.4 seconds, the computation of $\mathbf{K}_1$ took 12.0 seconds, and the solution of the linear system took 25.6 seconds. Solving the linear system for only one deformation function took less than three seconds less than solving it for all forty simultaneously. Of the 3481 nodes, 3249 were interior nodes.The large square matrix in (5.11) was of order 6730 but 99.77 percent of its entries were zero.

6.5.3 Evaluation of the Deformation Functions on the Pixel Grid

Before evaluating the deformation functions at a given pixel, it is necessary to determine which triangle that pixel belongs to. The actual evaluation of the finite element function at the pixel is done by mapping the triangle to the master element and performing the evaluation there. MATLAB has an efficient vectorized routine for determining which element of a triangulation a given point belongs to. All of the computations involved in evaluating the function can also be vectorized and performed simultaneously for all of the pixels.

Evaluation of one deformation function on the entire 256×256 pixel grid took 2.3 seconds. Ten percent of that time was used in determining which triangle each pixel belonged to and the remaining time was used in evaluating the finite element function.

6.5.4 Registering the Image by Deforming the Pixel Grid

This is not really a separate step, as no computation is required. It is, however, a distinct step in conceptualizing exactly how the registered image is created. Suppose that the ith pixel has coordinates (x_i, y_i) and that, at that pixel, the source image has intensity value $I(x_i, y_i)$. Then, if the registration function is defined by the coordinate deformations d_x and d_y, the deformed image has this same value at the location

$$\Big(x_i - d_x(x_i, y_i), y_i - d_y(x_i, y_i)\Big).$$

In other words, the registered image has the same range as the source image, but a different domain. Thus, just as the intensity values at each point of the pixel grid represent a discrete sample of the source image, those same intensity values at the deformed pixel grid represent a discrete sample of the registered image. This is a non-standard sample, however, as it is not supported on a regular lattice.

Strictly speaking, one should make the (somewhat subtle) distinction between a registered image, a registered function, and a registered observation. The original *image*, $\{I(p_i)\}$, is a set of intensity values associated with the pixels $\{p_i\}$ which, when plotted on a computer screen, displays a face. In assumptions of functional data analysis, this image is assumed to be an *observation* of an underlying bivariable *function* f by associating a point (x_i, y_i) with each pixel p_i and saying that f has been observed to take on the value $I(p_i)$ at the point (x_i, y_i). The function f will be estimated, through some estimation technique, by a function $\widehat{f}$.

The set $\{I_r(x_i - d_x(x_i, y_i), y_i - d_y(x_i, y_i))\}$, defined by

$$I_r(x_i - d_x(x_i, y_i), y_i - d_y(x_i, y_i)) = I(x_i, y_i),$$

is assumed to be a sample of the *registered function* $\widehat{f}_r$, defined by

$$\widehat{f}_r(x - d_x(x, y), y - d_y(x, y)) = f(x, y),$$

and hence can be thought of as a *registered sample.* The set $\{I_r(x_i - d_x(x_i, y_i), y_i - d_y(x_i, y_i))\}$ cannot, however, be thought of as a *registered image* because the points $(x_i - d_x(x_i, y_i), y_i - d_y(x_i, y_i))$ will not lie on a regular grid and hence cannot be used as pixel locations. The registered image $\{I_r(p_i)\}$ must be created by estimating the set $\{\widehat{f}_r(x_i, y_i)\}$.

6.5.5 Standardizing the Registered Image

To store the registered image in standard image format, as well as to display the image, it is necessary to obtain values of the image at the usual pixel grid locations. How best to do this is a smoothing problem. Since the images are black, and hence have intensity zero, on the boundary of the domain, one could use a feL-spline to smooth the registered pixels and then evaluate the function at the original pixel location. This involves a lot of nodes, even if one discards many of those in the background (all of which have intensity zero), and would take a long time to compute on a workstation. However, the solution would have the advantage of simultaneously representing the source and target image in functional form (that is, $\widehat{f}$ and $\widehat{f_r}$), depending on whether the original or the deformed nodes were used to define the feL-spline.

Another strategy would be to use a kernel smoother to estimate values at the original pixel locations. The data is distributed fairly densely on the domain and is equal to zero close to the boundary, so a kernel smoother with a very small bandwidth could be expected to perform fairly well.

The simplest approach, and the approach used for the examples in this chapter, is to estimate the intensity values at the original pixel locations by linear interpolation from the intensity values of the three closest deformed pixel locations. Although this is perhaps not an optimal strategy, it seemed to produce adequate images. For a sample face, this interpolation process took 7.1 seconds.

Chapter 7

Computational Experiments

7.1 Goals and Limitations of Computational Experiments

Although the results presented in Chapter 6 show that the finite element registration technique developed in this thesis can be useful in registering bivariable functional data containing landmarks, a number of questions still need to be addressed before the functional data analysis community can be confident that the technique is accurate and robust for general data. The most important issue is probably that of the accuracy of the registration method: how well can the feL-spline reproduce an arbitrary deformation from a set of landmark data?

The process of defining the precise location of a landmark by eye is likely to be noisy. Even automatic image processing techniques for landmark selection, such as edge-detection algorithms for identifying boundaries, will always contain some random error. Before deciding to use finite-element registration, one would like to have some idea of how this noise is likely to affect the final deformation. This is a question of sensitivity to extra sources of variation. In addition to knowing how landmark noise will affect the registration, it would be interesting to have some idea of just how much noise is involved in manually selecting landmarks.

The registration function obtained from a given set of data depends on the choice of the smoothing parameter λ. While the choice of λ could be optimized using the technique of cross-validation (as was discussed in section 2.2), this step might be avoided if the quality of the registration is not highly dependent on the value of λ. It would therefore be useful to investigate the effect on the registration process of selecting different values for λ.

The boundary conditions imposed by the variational form of the minimization criterion result in the deformation functions having value zero and having zero normal derivative on the boundary of the region Ω over which that registration function is estimated. In practice, this means that Ω must be chosen to be larger than the image being registered unless the boundary of that image is not being deformed. When registering the faces in chapter 6, Ω was chosen to be a rectangular box enclosing the face being registered. It is hoped that, as long as Ω is large enough, the boundary conditions will not have a great effect on the registration of the image. Indeed, the crispness of the boundaries of the means obtained in the face registration example suggests that this is true. The question remains, however, of how large Ω needs to be to effectively register the boundary of the image.

One can look for insight on the answers to questions such as these by performing computational experiments. To decide how well the finite-element registration method is able to model a given deformation, for example, one could estimate a known deformation and then compare the result with the true answer. Estimating a variety of different known deformations will give an idea of how the technique works in general. Ideally, one would like to choose a random sample of deformation functions from the population of possible deformations. In practice, this is not possible for several reasons. The set of possible deformations is both ill-defined and infinite-dimensional and hence one cannot sample from it in any truly random manner. Even if one could randomly select a deformation, the distribution of landmarks will still affect the accuracy of the estimate. No computational experiment can be truly random and hence any any insight arising from such experiments should be considered with some caution.

A variety of experiments were conducted to explore the qualities and limitations of the registration method presented in this thesis. All factors which were thought to be important to the accuracy of the registration were examined. Descriptions of those experiments, along with the results, are presented in the following sections. Although the experiments were performed using only the face data of chapter 6, it is reasonable to expect that the results are relevant to the more general situation of registering images with clearly defined boundaries and interior landmarks.

7.2 Accuracy of Estimation

In order to assess how well the finite-element registration method is able to register functional data, it would be helpful to apply random deformations to an image, or set of images, containing landmarks and to then assess how closely those deformations are reproduced by the method. Since there is no obvious method for choosing deformations in a completely random manner, it was decided to test the method using the twenty deformation functions used to register the female faces in chapter 6.

The test procedure was to apply a known deformation (one of the twenty female registration functions) to the landmarks in the male face depicted in figure (6.1) and then use the feL-spline registration method to register the deformed landmarks to their original positions. This corresponds to registering the deformed face to its undeformed original using perfect (error-free) landmark data. In the latter problem, the deformed face would be the face obtained by applying the known deformation function to the face in (6.1). The deformed landmark positions, obtained by applying the known deformation to the original landmark positions, would then correspond to the true, error-free landmark locations in the deformed face.

Denote the known deformation function by D_k and the deformation function estimated from the deformed landmark locations by D_e. Since a perfect registration method should map the deformed image *exactly* back to the original, D_e should invert the action of D_k and the composition $D_e \circ D_k$ should be the identity deformation. In other words, the equality $D_e \circ D_k(x, y) = (x, y)$ should hold for every point $(x, y) \in \Omega$. The registration error of the estimate D_e at the point (x, y) can be measured by the distance

$$D_e \circ D_k(x, y) - (x, y) \tag{7.1}$$

between where the point $D_k(x, y)$ is sent by the estimated registration function D_e and where it *should* be sent if D_e were really the inverse of D_k.

The overall error of an estimate D_e was estimated by the mean of the pixelwise registration errors, as defined in (7.1), taken over all the pixels in the smallest rectangle containing the original face. This rectangle was bounded on the left by the minimal x-value of the boundary, on the right by the maximal x-value of the boundary, on the bottom by the minimal y-value of the boundary, and on the top by the maximal y-value of the boundary. Since the feL-spline boundary conditions result in an error of zero at the boundary of Ω, the mean error taken over the smallest rectangle containing the face is smaller than the mean taken over all of Ω.

The inverse deformation of each of the twenty female registration functions was estimated and the results are presented in table 7.1 on page 119; all results are reported to two significant figures. The figures reported for each trial are the mean registration error, the root mean square registration error, the maximum registration error, and the percentage of pixels registered to within one pixel-width and to within a half pixel-width of their correct position. The mean error over all twenty tests is summarized in the bottom row of the table.

One can see from the bottom line of the table that, overall, the registration was quite accurate. The mean error over all twenty trials was only 0.43 pixel-widths and the average maximum error was 2.5 pixel-widths. On average, over all twenty trials, 92% of the pixels were registered to within an error of less than one pixel-width and 73% were actually registered with error less than a half pixel-width.

A closer examination of the other rows, representing the twenty individual trials, shows that the fourth trial was much worse than any of the others. In the fourth trial, the mean error was 1.3 pixel widths and the worst case error was 11 pixel-widths. Only 66% of the registered pixels in trial four had errors of less than one pixel width. A further calculation showed that 15% of the pixels had errors of more than two pixel-widths and that 10% actually had errors of more than three pixel-widths. On the other hand, only 3% of the pixels in the undeformed face were less than one pixel-width away from their corresponding pixels in the deformed face used in trial four (ie. $\|D_k(p_i) - p_i\| \geq 1$ for 97% of the pixels p_i). Therefore registering 66% of the pixels to within one pixel-width of their original positions still constitutes a substantial improvement.

The next worst case appears to be trial thirteen, with a maximum error of 4.4 pixel-widths. In this trial, 57% of the registered pixels had errors of less than a half pixel-width and 78% of the pixels had errors of less than one pixel-width. This is much better than the worst case, trial four. A further calculation showed that 90% of the registered pixels in trial thirteen had errors of less than two pixel widths. In conclusion, the registration algorithm did not perform particularly well for one of the twenty trials, but seemed to do much better for the other nineteen. Presumably the deformation used in trial four differed in some important way from the other deformations and was more difficult to reproduce. This phenomenon merits further study.

7.3 Variance in Landmark Selection

In order to locate the position of each landmark in a given face, the face was first displayed on a computer screen. The landmarks were then localized by using the mouse to position a cross on the perceived location of each landmark and then clicking the mouse. The computer recorded the position of the cross on the image when the mouse was clicked and this position was taken to be the location of the landmark.

It was noted in Chapter 6 that there was substantial variance in the localization process. This variance is caused by at least three factors. Probably the most important source of variance is the fact that it is frequently difficult to decide exactly where a given landmark is located. The bottom of the chin, for example, was not easy to localize. The shadow under the chin made it difficult to decide exactly where the boundary of the chin was. It was also not easy to decide exactly which point on the boundary was the "bottom" point. The result was that an arbitrary decision had to be made in localizing that point, and that decision would almost certainly be slightly different each time it was made.

A different, and presumably independent, source of error is the ability of a user to position a cross exactly on a given point on the screen. Once it is decided that the cross is in the correct position, the very act of clicking the mouse button will often result in a slight movement of the mouse and hence of the cross.

Another source of error lies in the fact that the movement of an object on the screen is necessarily discrete, with adjacent positions limited by the width of the pixels themselves. The range of possible locations of landmarks on the underlying continuous function, of which the digital image is a sample, however, is assumed to be continuous. The discreteness of the pixel grid means that the cross cannot, in general, be located exactly on the position of the continuously distributed landmarks.

Given the fact that there is (almost) certain to be error in the localization of landmarks, one would like to know both how much error to expect and what effect that error is likely to have on the quality of the registration. To gain some insight into the first question, I used the mouse to localize the set of 25 point landmarks in one male face twenty times. All of the landmarks in one set were localized before beginning the next set.

The mean of a set of selected locations $\{p_{1j}, \ldots, p_{20j}\}$ for the jth landmark, where each p_{ij} is an ordered pair, is defined to be the mean $\bar{p}_j = 20^{-1} \sum_{i=1}^{20} p_{ij}$ of the ordered pairs. The standard deviation of the set of p_{ij} can then be measured by the expression

$$\left(\frac{1}{20} \sum_{i=1}^{20} \|p_{ij} - \bar{p}_j\|^2 \right)^{1/2},$$

where the norm is the usual euclidean norm. This is the root mean square of the euclidean distances of the selected locations of the jth landmark from their mean. The 25 root mean square values are displayed in table 7.2 on page 120.

The estimated standard deviations of all landmarks except for the innermost point of the left eye, which has an estimated standard deviation of 1.26 pixel-widths, are less than one pixel-width. Given the fact that the innermost point of the left eye was often obscured by the shadow of the nose, it is not surprising that that landmark displays a larger standard deviation. Indeed, it is perhaps somewhat surprising that its observed standard deviation is not even larger. The root mean square of all of the 20 selected locations of all of the 25 landmarks (500 locations in all), corresponding to the expression

$$\left[\frac{1}{25}\sum_{j=1}^{25}\left(\frac{1}{20}\sum_{i=1}^{20}\|p_{ij}-\bar{p}_j\|^2\right)\right]^{1/2},$$

was 0.66.

Figure 7.1 shows a visual display of the variance in landmark localization. Each of the landmarks positions from each of the 20 trials is displayed simultaneously in this picture. The boundary landmarks for each trial were computed in the same manner as was described in Chapter 6. That is, two boundary curves were estimated from the selected locations using smoothing splines: one curve each for the left and right sides of the face. For each of these curves, 50 equidistant points were chosen to be landmarks. From the thickness of the cloud of points representing the boundary, the variance of the boundary localization process appears to be roughly comparable to the variance of the point landmark selection process.

7.4 Effect of Landmark Variance on Registration Quality

The investigation of the amount of variance present in the process of localizing the face landmarks suggests that this variance in the face example is of the order of 0.66 pixel-widths. This estimate on its own, however, does not yield a lot of insight into

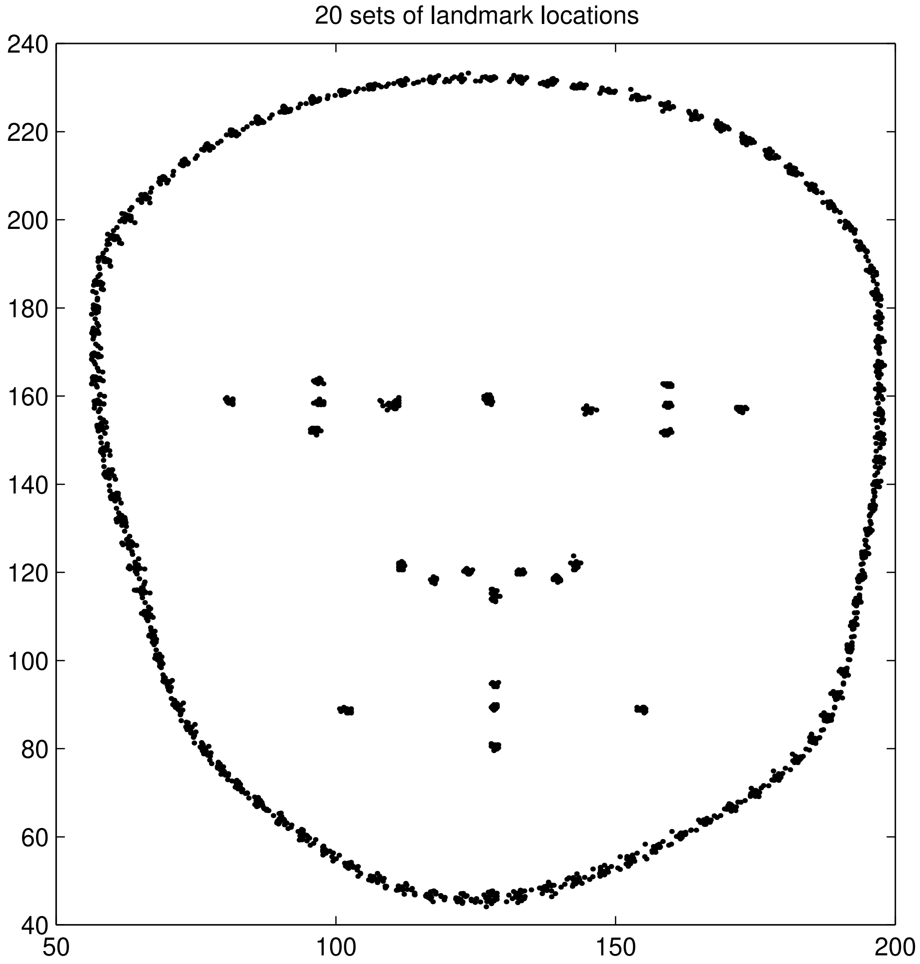

Figure 7.1: Variance in landmark localization

what effect this variance can be expected to have on the quality of the registration. To gain some idea of the magnitude of the effect of localisation variance on registration quality, further simulations were conducted to try to measure the effect of adding simulated noise to known landmark locations.

The effect of landmark variance was tested in a procedure similar to that described in section 7.2. A known deformation was applied to the landmarks in the face in figure 6.1, simulated noise was added to the deformed landmark locations, and the noisy landmarks were registered back onto the original locations. This corresponds to the problem of registering the deformed face to its original based on noisy information.

Recall that the boundary landmarks used to register the faces in chapter 6 were derived from a smaller set of manually selected boundary landmarks through a process in which the boundary was first estimated and then reparametrized. In order to properly simulate the effect of noise in the deformed landmarks, therefore, it was necessary to add the noise to the smaller set of manually selected landmarks and to derive the final set of noisy landmarks from this initial set. The precise procedure followed was to deform the set of manually selected landmarks of the original face, add noise to these, generate 50 equispaced points on each side of the (noisy) deformed boundary, and register these derived landmarks to the landmarks in the original face.

The noise at a given landmark location was modelled as the random variable (X, Y), where X and Y were independent and normally distributed with mean 0 and variance σ^2. The random variable (X, Y) can be re-expressed as $R(\cos\theta, \sin\theta)$, where the angle θ is uniformly distributed on the interval $[0, 2\pi)$, the radius R has the Rayleigh distribution with mean $\sigma(\pi/2)^{1/2}$, and R and θ are independent. Adding $R(\cos\theta, \sin\theta)$ to a landmark location thus corresponds to moving that location R pixel-widths in a (uniform) random direction. The resulting noise will referred be to as having mean deviation $\sigma(\pi/2)^{1/2}$.

Let the registration function estimated from the noise-free landmarks again be denoted by D_e and let the registration function estimated from a set of noisy landmarks be denoted by D_n. Then the error due to noise at a point (x, y) in the domain of the deformed image can be measured by the distance

$$|D_e(x, y) - D_n(x, y)| \tag{7.2}$$

between $D_e(x, y)$ and $D_n(x, y)$. This is a measure of the error due to noise alone, and does not take into account the feL-spline accuracy error examined in section 7.2. The overall error for a set of noisy landmarks was measured by the mean of the error, as measured by the expression in (7.2), taken over all of the pixels in the smallest rectangle containing the deformed image.

The noise was simulated using a random normal number generator for each value of σ in the set $\{0.1, 0.2, \ldots, 2.0\}$. For each value of σ in this set, and for each of the twenty female registration functions from chapter 6, a set of noisy landmarks was generated, registered, and the resulting mean error was recorded. All four hundred observed mean errors are displayed in figure 7.2. The x-axis represents the mean deviation of the noise and the y-axis represents the mean observed error in registration. Both axes are in units of pixel-width. There is a point for each of the 400 simulations and the solid line passes through the overall mean at each level of landmark noise.

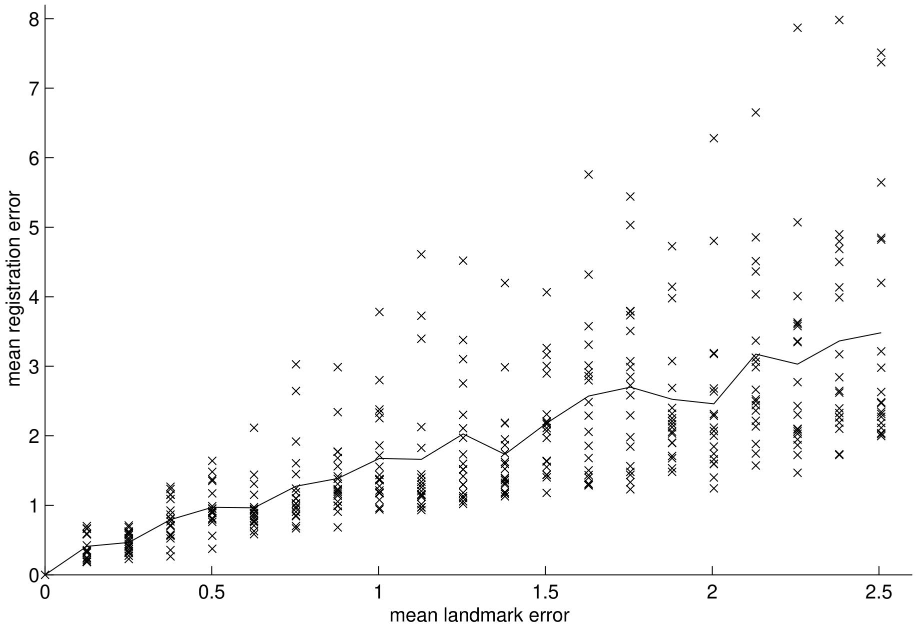

Figure 7.2: Evaluating effect of landmark noise on registration quality: simulation results

Possibly the most striking features of the simulation results are the fact that the observed error for a given quantity of landmark noise varies widely, and the fact that the distribution of observed mean errors appears to be skewed towards higher values. When noise with mean deviation 0.62 was added to the landmark locations, the mean observed registration error was 0.96 pixel-widths. The maximum observed registration error, however, was 2.12 pixel-widths. When the mean deviation of the simulated noise was 0.75, the mean observed error was 1.27 pixel-widths and the maximum observed error was 3.01 pixel-widths.

Since both 0.62 and 0.75 are relatively close to the observed landmark noise of 0.66, these figures suggest that the expected mean registration error due to landmark variance in the faces is likely to be at least one pixel-width. These results suggest that the registration algorithm is not very robust in the presence of noisy data. Further investigation of the simulation results, however, yields a somewhat more complex and more promising interpretation.

The smallest error added to the landmark locations in the simulations described in this section had a mean deviation of 0.125. Of the twenty simulations at this level, the mean observed error was 0.41 and the worst observed error was 0.70. It seems rather surprising that such a small amount of error in the landmark locations would result in such a large error in the registration function.

To examine this phenomenon further, the noisy derived landmarks were examined together. Recall that the noise was added to the landmarks which were localized using the mouse and that the boundary landmarks were then smoothed to produce a denser set of boundary landmarks from which to compute the registration function. Figure 7.3 shows examples of twenty sets of derived landmarks resulting from adding simulated noise of mean deviation 0.125 twenty times to the original landmarks in two different faces. No registration was used on the landmarks in figure 7.3: they were obtained by adding noise to the original, undeformed landmarks of the face in figure 6.1.

There are three important points to notice in figure 7.3. Firstly, the amount of noise added can be assessed by examining the 23 point landmarks in the interior of each face. These points were unaffected by the boundary smoothing. The small radii of the dot clouds corresponding to the interior landmarks indicate that the simulated noise was of small magnitude. Many of the dot clouds corresponding to derived points on the boundary, on the other hand, indicate a substantially larger amount of variance than is present in the interior landmarks. This would appear to be a product of the boundary smoothing procedure, since the original noisy boundary landmarks

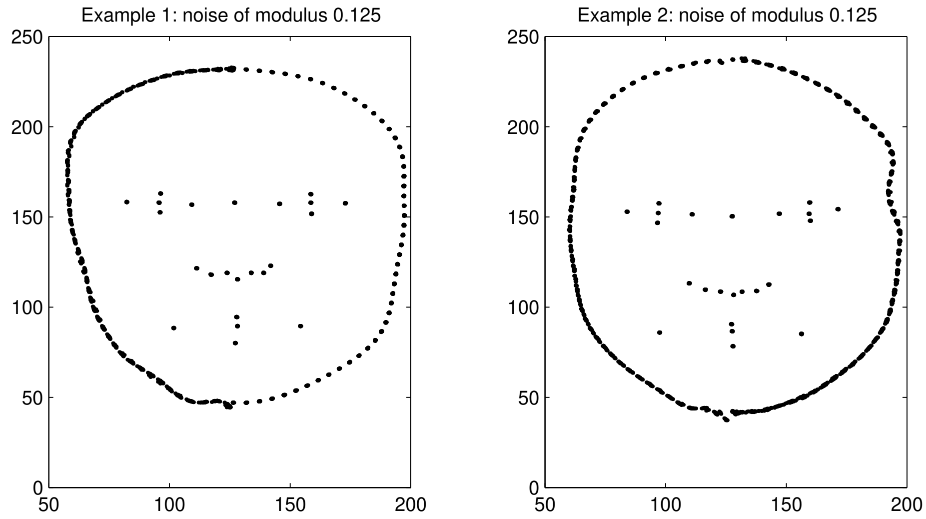

Figure 7.3: Two examples of twenty sets of derived landmarks resulting from simulated noise of modulus 0.125

would have had the same pattern of noise as is visible in the interior landmarks.

Recall the procedure for deriving the evenly spaced boundary landmarks from the original boundary landmarks. Smooth curves were estimated for the left and right halves of the boundary from the original boundary landmarks using smoothing L-splines chosen to yield smooth derivative estimates. Then, using these derivative estimates, the curves were reparametrized by arc length and fifty equally spaced points were chosen along the estimates of each half of the boundary. From the illustration in figure 7.3, it would appear that local changes in arc length arising from small amounts of noise in the original boundary landmarks are capable of producing changes in the locations of the equally spaced points which have substantially greater magnitude than the original noise.

Two other interesting features present in figure 7.3 are that some collections of landmarks appear to be more sensitive to this effect than others, and that distinguished point landmarks representing the top of the forehead and the bottom of the chin do not appear to be affected as much as the points in the interior of each boundary curve. The points on the right side of example 1 did not appear to be perturbed by much more than the the magnitude of the original noise, while the points on the left side of the same example changed by much more than the magnitude of the original noise. Although example 1 in figure 7.3 represents only twenty simulations, this difference between the left and right sides was still present after one hundred simulations. In example 2, it appears that certain points on either side of the boundary are more affected than others. Why there would be such marked differences in sensitivity to noise depending on the original landmarks is a complete mystery at this point and would be an interesting topic for further research.

The fact that the endpoints of the estimated curves would be less sensitive to noise is not particularly surprising. With only a small amount of noise added, the estimated noisy curve should still pass very close to the estimated noise-free curve. If the curve is then reparametrized by arc-length and has as domain the interval $[0, T]$, then its value at $t = 0$ and at $t = T$ should remain relatively unchanged.

The results presented in figure 7.3 suggest that the relatively large mean registration error observed in the simulations may well be heavily influenced by the error in the boundary points. Indeed, the plots in figures 7.4 and 7.5 confirm that the largest error does appear on the boundaries. The figures use grayscale to display continuous plots of the registration error in the smallest windows containing the derived landmarks. The colourbar at the side of each plot shows the correspondence between colour (grayscale) and magnitude of error. Both plots correspond to simulations with landmark error of mean 0.75. Since this was the level of error which most closely matched the level of landmark error (0.66) observed in the actual landmark localization process, it was felt that this level might best represent the behaviour of the true error due to landmark variance. The plot in figure 7.4 represents the simulation with the largest mean error and the plot in 7.5 represents the simulation with the mean error closest to the overall mean of the 20 simulations with noise of magnitude 0.75.

The fact that the lightest parts of the plots appear on the left half of the boundary shows that the error is greatest in the area of the boundary which is most sensitive to landmark noise. The error is not as great at the distinguished points on the boundary,

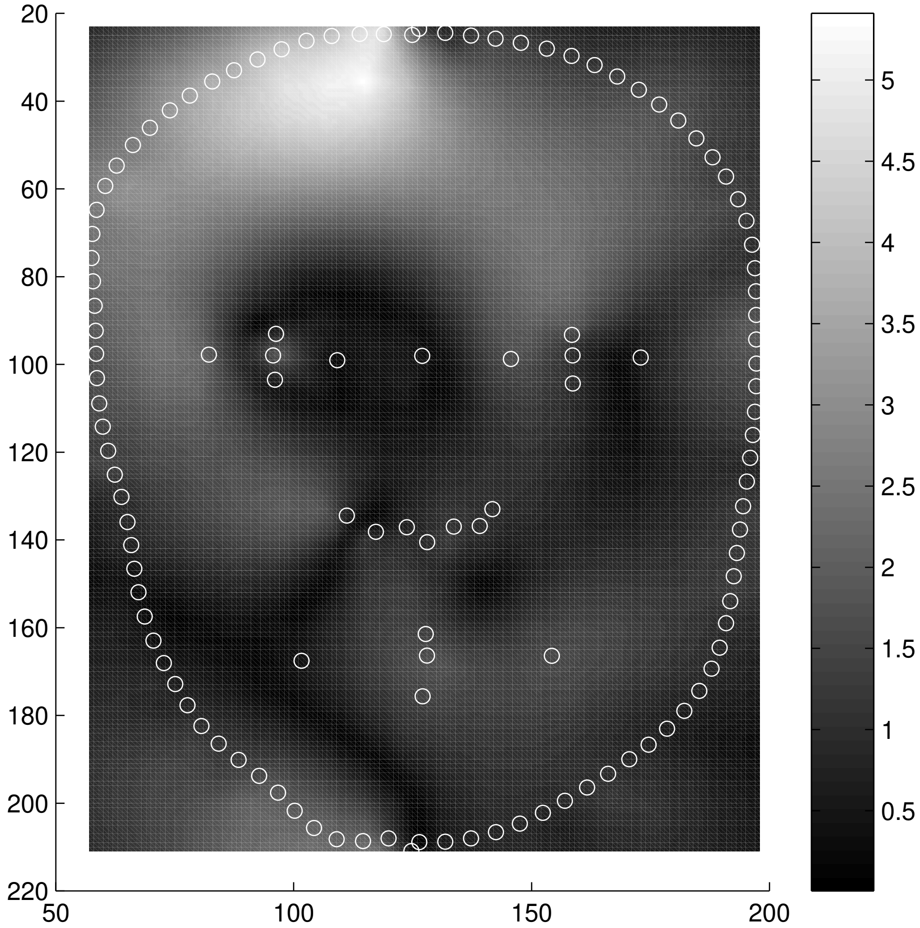

Figure 7.4: Registration error resulting from simulated landmark noise of mean modulus 0.75; the mean registration error is 3.01 pixel-widths

however. This suggests that when defining one-dimensional landmarks, a user should use any distinguished points available. The one-dimensional landmarks should be defined separately on segments between adjacent point landmarks.

7.5 Influence of the Smoothing Parameter

In order to compute a registration function using the feL-spline, it is necessary to specify a value for the smoothing parameter λ which controls the trade-off between smoothness and fidelity to the landmark data. In the face examples in Chapter

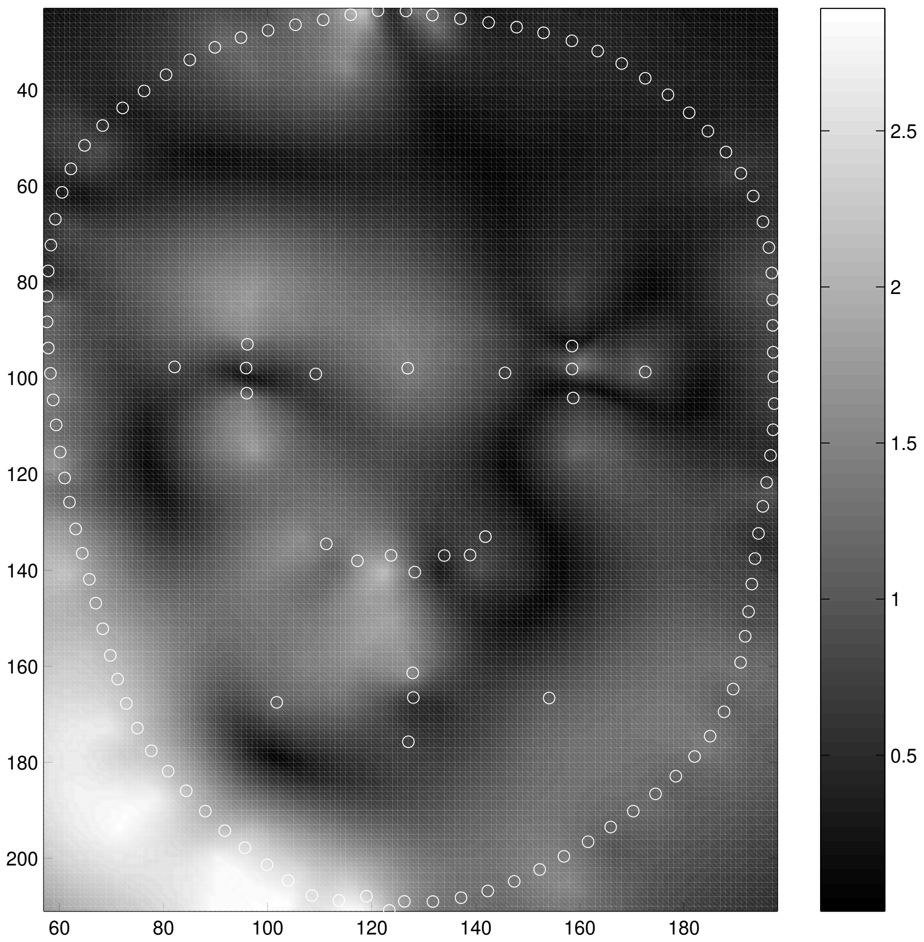

Figure 7.5: Registration error resulting from simulated landmark noise of mean modulus 0.75; the mean registration error is 1.24 pixel-widths

6, a value of 10 was arbitrarily chosen for λ. In principle, it is possible to use a cross-validation technique to allow the data to determine an optimal value for λ but this imposes an additional computational load on the registration procedure. If the particular value of λ does not have a great influence on the registration function, one can save work by choosing a value arbitrarily. If, on the other hand, the value of λ does exert a great influence on the final result then one would be well advised to implement some sort of cross-validation procedure to choose the optimal value.

In order to assess the impact of different choices of the smoothing parameter, registration functions were computed for nine different values of λ. The values chosen were $\lambda = 10^i$ for $i \in \{-2, -1.5, \ldots, 1.5, 2\}$. Beginning with a fixed set of landmarks from a male face, each value of λ was used to compute a registration function for registering those landmarks to the male mean. Thus for the same set of landmarks, nine different registration functions $R(-2,\cdot), \ldots, R(2,\cdot)$ were computed, where $R(j,\cdot)$ was computed using $\lambda = 10^j$. The mean pairwise difference in pixel location after registration was used to compare the different registration functions. This was again done only over the smallest rectangle containing all of the landmarks. For example, the mean of $\|R(-2,p) - R(2,p)\|$ taken over all of the pixels p in the smallest rectangle containing all of the landmarks is a measure of the difference between $R(-2,\cdot)$ to $R(2,\cdot)$. This was done for each of the twenty different deformation functions and the mean of the pairwise mean differences is presented, to two significant figures, in table 7.3 on page 121.

It is rather surprising that the numbers in table 7.3 are so small, as this strongly suggests that the particular value of λ chosen makes very little difference in the final registered image. The numbers in table 7.3 are actually means of mean differences, with each number being the mean of twenty different trials, each conducted with a different set of landmarks. Of the twenty trials comparing the smoothing parameter values of 10^{-2} and 10^2, the largest individual mean difference was only 0.98. Therefore in no comparison did the mean difference ever exceed one pixel-width.

The reason for the small differences between different smoothing parameters lies in the performance of the feL-spline. Increasing the value of the smoothing parameter λ corresponds to enforcing a larger degree of fidelity to the data while decreasing it corresponds to enforcing greater smoothness. Changing from a smaller value value of λ to a larger one seems to result in rather localized changes in the resulting finite-element surface. Plots of the x-coordinate transformation for a typical set of landmarks, using $\lambda = 1$ and $\lambda = 100$ respectively, are presented in figures 7.6 and 7.7.

The only noticeable difference between these two plots is that one of the small peaks in the center of the plots appears slightly sharper in figure 7.7. From table

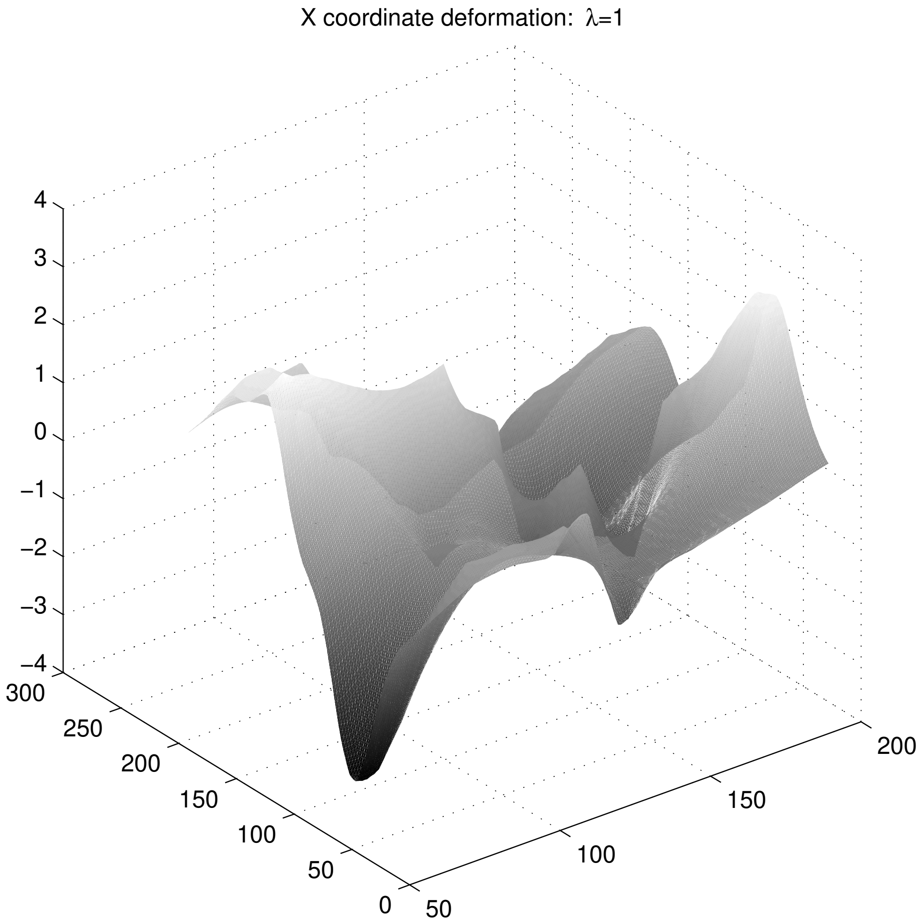

Figure 7.6: A typical x-coordinate transformation with $\lambda = 1$

7.3, it as apparent that the mean difference between any of the registration functions using $\lambda \geq 1$ is less than 0.1 pixel-widths. As λ approaches zero, the feL-spline approaches the constant surface $f(x, y) = 0$. In plotting the coordinate deformation functions obtained using various landmark sets and various values of λ, it appears that when the value of λ gets much smaller than 10^{-1}, the resulting feL-spline tends to oversmooth the data by removing any real peaks in the surface. Based on the results of this experiment, it seems reasonable to suggest that for practical purposes the smoothing parameter should be assigned a value somewhere in the range from 10^{-1} to 10^{1}. The value 10^{0}, or one, seems to be a good general choice but assigning larger values will not have a great effect on the final outcome.

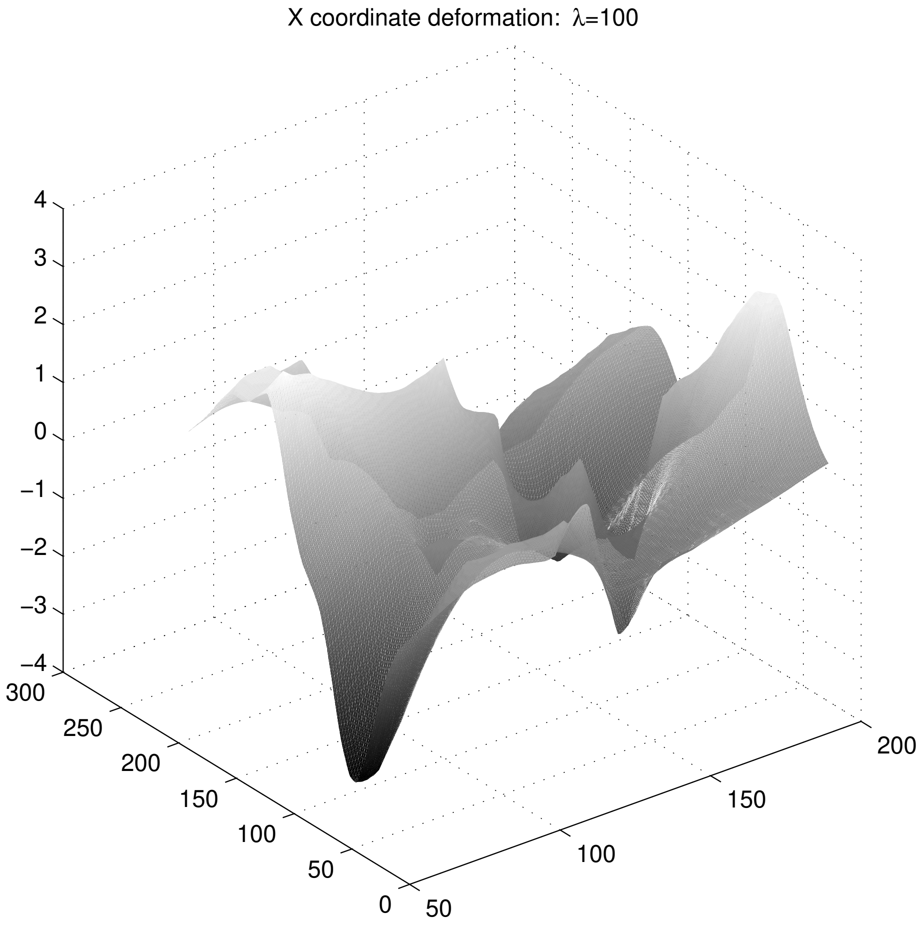

Figure 7.7: A typical x-coordinate transformation with $\lambda = 100$

7.6 The Effect of the Choice of Ω

A deformation function computed over a region Ω of the real plane using the feL-spline method will necessarily have value zero and have normal derivative equal to zero on the boundary of Ω. If, as is the general case, one is registering an image whose boundary must be deformed to obtain an adequate registration, this means that it is necessary to compute the registration function over a region that is larger than the image being registered. Since the registration function will also be continuous (C^0), the boundary conditions can be expected to also affect points close to the boundary of Ω. This suggests that, in order to allow the image boundary to be freely deformed,

Ω should be chosen so that its boundary is not too close to the boundary of the image. It is natural to wonder how far away the boundary of Ω needs to be from the boundary of the image in order to accurately register the image.

In order to explore the impact of the choice of Ω on the registration quality, a different registration function was computed for the same set of landmarks using each of twenty differently sized rectangular domains $\{\Omega_1, \ldots \Omega_{20}\}$. The rectangular domains were chosen so that, for the domain Ω_i, the distance between each side of Ω_i and the closest landmark was equal to $5i$ pixel-widths.

Let R_i be the registration function computed on domain Ω_i. The mean distance, taken over all pixels p in the region of the image, between $R_i(p)$ and R_{20} was used as a rough measure of the performance of each R_i. This decision was based on the belief that the registration function computed over the largest domain should, in general, be the most accurate. The results are presented in table 7.4 on page 121.

The mean difference seems to drop exponentially as the window gets larger. It is difficult to make any firm suggestion for the window size based on the results of this one sequence of simulations, but it appears that the registration does not improve substantially for windows larger than Ω_{10}. This suggests that the quality of the registration does not depend greatly on the distance of the boundary of Ω from the image boundary as long as that distance is larger than some threshold size.

If it is true that one can obtain a registration which is close to optimal as long as Ω is large enough, one might still ask what exactly is "large enough". The answer to this question is likely to depend on the characteristics of the data. If the differences in position of landmarks near the boundary have small magnitude compared to the landmarks further from the boundary, then one would expect that the effect of the boundary conditions would be relatively small. Also, it seems reasonable to expect that the ideal size of the domain Ω might depend on the diameter of the image being registered, with larger images requiring larger domains in order to obtain adequate registration. The precise relationship between the data and the best size of Ω is a subject for further investigation.

The rectangular domain Ω_{10} in this section, the domain which appears to be of the minimum size for optimal results, was constructed by setting the minimum distance between each edge and the image boundary landmarks to 50 pixel-widths. In the face example of chapter 6, the domain was kept constant at 256×256 pixels for each face while the size of the individual faces varied. In all of the faces, the minimum distance between the face boundary and domain boundary was the distance between the top of the head and the upper edge of the domain. For the largest face, this distance was only eleven pixel-widths; for the smallest it was only thirty pixel-widths. This suggests that perhaps the registration quality might have been improved if the domain had been chosen to be somewhat larger.

7.7 Conclusions

As was noted at the beginning of this chapter, the deformations and sets of landmarks used in the computational experiments presented here are not randomly selected from the population of all possible deformations and landmark sets. The conclusions which can be drawn from the results of these experiments may therefore not apply to all sets of bivariable functional data. However, the simulations of this chapter were chosen to address the main factors thought likely to affect the registration quality. It is reasonable to hope that the results obtained herein should be applicable to general face data and even to any data with a roughly circular boundary and landmarks within the boundary.

The tests in section 7.2 indicate that the finite-element registration method is usually accurate when applied to error-free landmark data. Twenty different known deformations were applied to a set of landmarks locations derived from a single face. A registration function was computed from each set of deformed landmark locations to register that deformed set to the original. This corresponds to registering a deformed face with that face before deformation based on error-free landmark data. The mean registration error taken over each pixel of each of the twenty faces was only 0.43

pixel-widths but one of the twenty simulations produced substantially more error than average.

The process of localizing landmarks in a given image is subject to variability and this variability was experimentally estimated in section 7.3 to have a standard deviation of approximately 0.66 pixel-widths. The simulations in section 7.4 demonstrated a surprising sensitivity to error in landmark selection. Varying degrees of simulated error were added to a set of landmarks locations and it was found that, on average, the magnitude of the resulting registration error was substantially larger than the magnitude of the landmark error. In addition, there was a substantial amount of variation in the mean registration error resulting from a given amount of landmark error. The experiment in section 7.3 only addresses the issue of within operator selection variance. The question of how much additional variance to expect if several different individuals are involved in localizing the same landmarks remains to be explored.

Closer examination of the simulation results of section 7.4 leads to the conclusion that small amounts of error in the locations of point landmarks on a one-dimensional landmark are likely to produce comparitively large local changes in the arc length of the parametrized landmark. This, in turn, results in correspondingly large changes in the locations of equally spaced points on the parametrized landmark. The observed amplification of error therefore occurs during the smoothing of the boundary and generation of equispaced points rather than in the feL-spline computation itself. The location of the endpoints of a parametrized one-dimensional landmark, on the other hand, are likely to be less affected by error in the selected landmark locations. These results suggest that one should include as many distinguished point landmarks as possible when defining one-dimensional landmarks and that the one-dimensional landmark should be parametrized separately between adjacent point landmarks.

The results of the computational experiments in section 7.5 strongly suggest that the particular choice of the smoothing parameter λ does not greatly affect the results of the the finite-element registration method. The registration function for registering each of the twenty male faces to the mean male face was estimated using nine different values of λ. The mean difference in the location of a pixel using $\lambda = 0.01$ and its location using $\lambda = 100$ was only 0.74 pixel-widths and the mean difference between using $\lambda = 1$ and $\lambda = 100$ was only 0.06 pixel-widths. The results suggest that the registration method is, for practical purposes, invariant to the choice of λ as long as $\lambda \geq 1$.

In section 7.6, a registration was computed using rectangular domains of twenty different sizes. The results indicate that as long as the minimum distance between the boundary of the registration domain Ω and the boundary of the image being registered is larger than some threshold value, the size of Ω does not have a large effect on the quality of the registration. Although the results suggest that a minimum distance of 50 pixel-widths is adequate for registering the face example, the optimal distance for an arbitrary set of data would probably depend at least on the diameter of the image and the magnitude of the deformation close to the boundary of the image. The density and uniformity of landmark distribution on the boundary are also likely to affect the optimal size of the registration domain.

trial	**mean**	**rmsq**	**max**	% <1	% <0.5
1	.31	.38	1.3	99	80
2	.45	.70	2.9	89	75
3	.36	.44	1.4	98	76
4	1.3	2.3	11	66	36
5	.31	.49	2.5	94	86
6	.56	.68	2.2	85	55
7	.39	.46	1.3	98	71
8	.38	.49	1.9	94	72
9	.55	.74	2.5	83	62
10	.23	.29	1.3	99	92
11	.39	.48	1.8	98	67
12	.35	.42	1.1	99.8	75
13	.75	1.1	4.4	78	57
14	.29	.36	1.2	99.8	83
15	.24	.35	1.9	98	88
16	.26	.31	1.4	99.7	89
17	.50	.70	3.0	85	68
18	.28	.35	1.7	99	85
19	.34	.53	2.6	93	82
20	.43	.56	2.1	92	69
mean:	.43	.61	2.5	92	73

Table 7.1: Summary of registration error in accuracy tests: mean, root mean square, maximum, percent less than one pixel, and percent less than 0.5 pixels

Landmark	**Std. deviation**
Left eye: outer corner	0.57
Left eye: inner corner	1.26
Left eye: pupil	0.64
Left eye: top	0.54
Left eye: bottom	0.62
Right eye: outer corner	0.59
Right eye: inner corner	0.71
Right eye: pupil	0.43
Right eye: top	0.49
Right eye: bottom	0.54
Nose: leftmost point	0.69
Nose: rightmost point	0.83
Nose: bottom	0.93
Nose: top	0.76
Left nostril: outer point	0.50
Left nostril: inner point	0.49
Right nostril: outer point	0.55
Right nostril: inner point	0.48
Mouth: leftmost point	0.67
Mouth: rightmost point	0.56
Mouth: top center	0.41
Mouth: middle center	0.43
Mouth: bottom center	0.57
Chin: bottom center	0.99
Forehead: top center	0.53

Table 7.2: Landmark localization: estimated standard deviation in pixels

$\mathbf{10^{-2}}$	0								
$\mathbf{10^{-1.5}}$	0.33	0							
$\mathbf{10^{-1}}$	0.53	0.22	0						
$\mathbf{10^{-0.5}}$	0.64	0.34	0.13	0					
$\mathbf{10^{0}}$	0.70	0.41	0.21	0.077	0				
$\mathbf{10^{0.5}}$	0.73	0.44	0.24	0.11	0.038	0			
$\mathbf{10^{1}}$	0.74	0.45	0.26	0.13	0.053	0.015	0		
$\mathbf{10^{1.5}}$	0.74	0.45	0.26	0.13	0.058	0.020	0.0053	0	
$\mathbf{10^{2}}$	0.74	0.46	0.26	0.14	0.060	0.022	0.0070	0.0017	0
λ	$\mathbf{10^{-2}}$	$\mathbf{10^{-1.5}}$	$\mathbf{10^{-1}}$	$\mathbf{10^{-0.5}}$	$\mathbf{10^{0}}$	$\mathbf{10^{0.5}}$	$\mathbf{10^{1}}$	$\mathbf{10^{1.5}}$	$\mathbf{10^{2}}$

Table 7.3: Mean pixel differences between registrations using different values of smoothing parameter

Ω_1	1.76	Ω_6	0.52	Ω_{11}	0.30	Ω_{16}	0.22
Ω_2	1.18	Ω_7	0.44	Ω_{12}	0.28	Ω_{17}	0.13
Ω_3	0.91	Ω_8	0.41	Ω_{13}	0.26	Ω_{18}	0.15
Ω_4	0.71	Ω_9	0.35	Ω_{14}	0.23	Ω_{19}	0.20
Ω_5	0.60	Ω_{10}	0.29	Ω_{15}	0.09	Ω_{20}	0

Table 7.4: Mean difference in registration between the registration function defined on Ω_i and that defined on Ω_{20}

Chapter 8

Conclusions And Directions For Further Research

8.1 Conclusions

While the data used in functional data analysis will almost invariably be collected in a discrete form, the data must be analyzed as continuous functions. Any distinguished landmarks in a sample of these continuous functions must occur at the same points in the domain of each element of the sample before a useful mean function can be estimated. Effective methods for estimating continuous functions from discrete data and for registering landmarks in these continuous functions are therefore essential to the analysis of functional data.

In the case of univariable functional data, the general class of L-spline smoothing splines is useful both for estimating continuous functions and for functional registration, but there exists no comparable class of estimation methods for use with arbitrarily spaced multivariable data. As landmarks are unlikely to occur on a regular grid, the capacity to estimate functions from arbitrarily spaced (scattered) data is essential for any generally useful landmark registration technique. The one exception to this lack of generally applicable multivariable function estimation tools is the thin-plate spline, which can effectively estimate smooth functions from scattered data in the plane by minimizing a penalty functional corresponding to the bending energy of a thin steel sheet. Although there are a number of registration techniques in current use, especially in the field of medical imaging, the only method in general use which registers landmarks in a smooth manner is based on the thin-plate spline.

Given a set of observations $\{(\mathbf{p}_1, z_1), \ldots, (\mathbf{p}_d, z_d)\}$ of a multivariable function, where $\mathbf{p}_i$ is a point in $\mathbb{R}^n$ and z_i is the value observed at $\mathbf{p}_i$, one can easily generalize the definition of the L-spline in the following manner. The L-spline smoothing function defined on a domain Ω by the $\{(\mathbf{p}_1, z_1), \ldots, (\mathbf{p}_d, z_d)\}$, the linear partial differential operator L, and by the positive constant λ is the function f which minimizes the expression

$$\sum_{i=1}^{d} (f(\mathbf{p}_i) - z_i) + \lambda \int_{\Omega} (Lf)^2 d\Omega. \tag{8.1}$$

The partial differential operator L is chosen to define "smoothness" by specifying that the right-hand term in (8.1) will be smaller for smoother functions and larger for less smooth functions. The left-hand term in (8.1) measures the fidelity of f to the data and the smoothing parameter λ determines the relative importance given to smoothness over fidelity.

Since the choice of axes used in parameterizing multivariate data will usually be arbitrary, only partial differential operators which are invariant to this choice should be considered as candidates for the operator L in (8.1). It has been shown that the class of such operators is the ring generated by the Laplacian Δ. A method for computing a solution to a weak variational characterization of the minimizer of (8.1), subject to certain boundary conditions, has been developed and shown to be computable. This weak solution is not exact, but approximates the true minimizer by the minimizer in a finite element space defined by quadratic Lagrange finite elements on a triangulation of the domain Ω.

The weak characterization of the minimization criterion has been shown have a unique solution and an algorithm for computing that solution has been described for any operator L and has been implemented for the operator $L = \Delta$. This algorithm has been used to register a sample bivariable data set consisting of 40 digital images of faces and a mean face was estimated from the registered faces. The mean of the registered faces appeared much sharper (less blurred) than did the mean of the unregistered faces. It was thus shown that the finite element L-spline (feL-spline) presented in this thesis can be effectively used to register bivariable functional data.

Although it is not clear that there is any great difference between the complexity of computing and using the thin-plate spline and the feL-spline, the feL-spline has three distinct advantages over the thin-plate spline. Firstly, the feL-spline can be used for estimating functions on $\mathbb{R}^n$ for any n, while the use of the thin-plate spline is restricted to the plane. Secondly, while the thin-plate spline penalty has been generalized to a three parameter penalty ([38],[15]), the feL-spline can penalize the most general possible class of linear partial differential smoothing functionals. Thirdly, there is, to the best of my knowledge, no way to incorporate any type of boundary conditions into a thin-plate spline estimate. It is easy, on the other hand, to construct feL-spline estimates which will have specified values on either (piecewise linear) curves or at isolated points.

Several types of simulations and computational experiments were conducted in an attempt to assess the accuracy and robustness of the feL-spline in registration. These tests suggested that feL-spline registration is fairly accurate when applied to error-free landmark data. Although there is some innate variability present in the process of locating landmark positions manually, the variance in the position of manually selected landmarks in the face example appears to be only about 0.7 pixel-widths; the dimension of the images themselves was 256×256 pixels.

It was found to be difficult to accurately assess the effect that this variance would have on the registration procedure itself. This was because relatively small errors in landmark location appear to lead to larger errors in arc-length when estimating one-dimensional landmarks such as boundaries. This phenomenon warrants further study. For the present, it is recommended that as many point landmarks as are available be included in parametrized curve landmarks since the error at these point landmarks will tend to be smaller.

The particular value chosen for the smoothing parameter λ appears to have relatively little effect on the quality of the registration as long as λ is not chosen to be extremely small. For practical purposes, it appears that any value for λ of one or greater will result in roughly the same registration function.

The boundary conditions imposed by the solution method force the deformation function to be zero at the boundary of the domain Ω. Thus, unless the boundary of the image does not need to be deformed by the registration procedure, Ω must be chosen to be strictly larger than the image being registered. As was hoped, it appears that the quality of the registration does not greatly depend on the choice of Ω as long as the boundary of Ω is not too close to the boundary of the image being registered.

Overall, it seems reasonable to assert that the feL-spline has the potential to be adopted as a useful registration tool by the functional data analysis community. Perhaps it may also turn out to be useful in other fields where it is necessary to estimate smooth functions from scattered multivariate data.

8.2 Directions For Further Research

8.2.1 Analysis of the Registration Technique

It would be interesting to analyze the performance and limitations of the feL-spline registration method in a more general context than was done in chapter 7. In particular, it would be useful to learn more about how to choose an optimal registration domain Ω for an arbitrary set of bivariable functional data and to further explore the effect of landmark noise on the accuracy of the registration. The selection of an optimal domain Ω could potentially be explored either experimentally or analytically. Experimental techniques could include either generating some sort of random deformation functions and landmark sets or else analyzing a variety of bivariable functional data in a manner similiar to the analysis presented in section 7.6. It is also possible that the functional analysis techniques used to analyze the estimation error of the finite-element method could be modified to analyze the relationship of that error to domain size, landmark distribution and magnitude of deformation.

One factor which might at least partly explain the sensitivity of parametrized curve landmarks to error is the choice of numerical integration technique. Numerical integration is used heavily in both the computation of the spline estimate of the curve landmark from a relatively sparse set of points on the curve and in the reparametrization of that estimate by arc length. It is possible that more stable numerical integration techniques might produce more robust estimates. This possibility might also be explored both analytically and experimentally.

Just as it is expected that the optimal size of the domain Ω probably depends on the set of landmarks, it is reasonable to expect that the quality of approximation of the feL-spline would depend on the data used to compute the estimate. For example, one would expect that it would be difficult or impossible to accurately estimate a complicated surface if there were no data points in large areas of that surface. One would also like to have more data in areas of greater variability and would need less data in areas which were relatively flat. It would be interesting to analyze the performance of the feL-spline for different data configurations.

8.2.2 A Multigrid Algorithm for the FeL-spline

The quality of a finite-element estimate of a function generally depends on the size of the individual finite elements with local estimation error being proportional to the size of those elements. This means that a triangulation composed of many small triangles will produce a more accurate estimate than a triangulation composed of relatively few large triangles. Computing a finite-element function, on the other hand, involves inverting the action of a matrix whose size depends directly on the number of finite elements involved. If there are too many finite elements in the triangulation, it can become computationally infeasible to construct the estimate.

Multigrid methods for solving finite-element problems involve computing a rough initial estimate of the solution on a coarse triangulation and then smoothing the error on progressively finer triangulations. Only the first rough estimate requires the inversion of a matrix and hence multigrid algorithms tend to have approximately $\mathcal{O}(n)$ complexity, where n is the number of nodes in the finest triangulation. A multigrid algorithm for computing the feL-spline would be valuable for solving problems on larger triangulations. If it were available, such an algorithm would make the registration technique presented in this thesis both more accurate and more efficient. The registration functions in the face example were constructed from triangulations containing about 1700 finite elements each and the computation of one registration function took about 30 seconds on a Sun Ultra5 workstation. Although no formal calculations were performed, the time required for computing the registration functions was observed to increase at a greater than linear rate with the number of finite elements used. An efficient multigrid algorithm would be useful or perhaps even necessary for computing feL-spline functions based on much larger (than 1700) collections of finite elements.

8.2.3 Bivariable Smoothing

Although this thesis has focussed exclusively on functional registration, the feL-spline could also be used for smoothing bivariable functional data. It may well be possible to express the feL-spline as a Bayesian estimator relative to some prior distribution.

It would be interesting to explore the effect of using different penalty functions in estimating the feL-spline. As was mentioned in section 4.12, Chen and Suter [15] used several different penalty functions to compute interpolating tension spline surfaces from a set of 25 data points. These different interpolants, which were obtained through altering the thin-plate spline penalty functional, appeared to be very different, with higher order penalty functionals seeming to lead to smoother looking surfaces. It is reasonable to expect that using higher-order Laplacian penalty functions might also result in smoother feL-splines.

8.2.4 Estimating Partial Derivatives

In the finite element approximation to the bivariable L-spline, the feL-spline u is estimated simultaneously with the function $s = u_{xx}+u_{yy}$. Although only the function u was used to construct the registration function, an independent estimate of the function s is also available to the user. For the purpose of studying the derivative structure of u, it would be more useful to have independent estimates of the partial derivatives u_{xx} and u_{yy}. It should be possible to achieve this goal by simply solving the following system for $(u, \chi, \varphi) \in H^1_{0h} \times H^1_h \times H^1_h$:

$$\begin{aligned} \textstyle\int(\chi v_x + \varphi v_y) - \lambda \sum u(p_i)v(p_i) &= \lambda \textstyle\sum v(p_i)z_i && \text{for all } v \in H^1_{0h} \\ \textstyle\int \chi t + \int u_x t_x &= 0 && \text{for all } t \in H^1_h \\ \textstyle\int \varphi w + \int u_y w_y &= 0 && \text{for all } w \in H^1_h. \end{aligned} \tag{8.2}$$

This corresponds to letting $\chi = u_{xx}$ and $\varphi = u_{yy}$, or $\chi+\varphi = s$ with s as defined in the formulation used in chapter 4. Its ability to simultaneously estimate a function and one or more of its derivatives is one of the most valuable characteristics of the univariable L-spline. The construction in (8.2) suggests that it may be possible to extend this valuable feature, at least to some degree, to feL-splines.

Bibliography

[1] N. Arad, N. Dyn, D. Reisfield, and Y. Yeshurin. Image warping by radial basis functions: application to facial expressions. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 56(2):161–172, 1994.

[2] R. Bajcsy and C. Borit. Matching of deformed images. In *Int. Joint Conf. Pattern Recognition*, 1982.

[3] R. Bajcsy and S. Kovacic. Multiresolution elastic matching. *Computer Vision, Graphics, and Image Processing*, 46:1–21, 1989.

[4] R. Bajcsy, R. Lieberson, and M. Reivich. A computerized system for the elastic matching of deformed radiographic images to idealized atlas images. *Journal of Computer Assisted Tomography*, 7(4):618–625, 1983.

[5] P.K. Bannerjee and A.W. Toga. Image alignment by integrated rotational and translational transformation matrix. *Physics in Medicine and Biology*, 39:1969–1988, 1994.

[6] D. C. Barber, W.B. Tindale, E. Hunt, A. Mayes, and H.J. Sagar. Automatic registration of SPECT images as an alternative to immobilization in neuroactivation studies. *Physics in Medicine and Biology*, 40:449–463, 1995.

[7] I. Barrodale, D. Skea, M. Berkley, R. Kuwohara, and R. Poeckert. Warping digital images using thin plate splines. *Pattern Recognition*, 26(2):375–376, 1993.

[8] E. Becker, G. Carey, and J. Oden. *Finite elements: an introduction*, volume 1. Prentice-Hall, New Jersey, 1981.

[9] F.L. Bookstein. *Morphometric Tools For Landmark Data: Geometry And Biology*. Cambridge University Press, 1991. Cambridge.

[10] Dietrich Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, Cambridge, 1997. Translated by Larry L. Schumaker.

[11] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, 1994. New York.

[12] M.J. Buckley. Fast computation of a discretized thin-plate smoothing spline for image data. *Biometrika*, 81(2):247–58, 1994.

[13] G. Carey and J. Oden. *Finite Elements: a Second Course*. Prentice-Hall, New Jersey, 1983.

[14] Manfredo Perdigao do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[15] F. Chen and D. Suter. Multiple order laplacian splines- including splines with tension. Technical report, Department of electrical and computer systems engineering, Monash University, Melbourne, Australia, 1996.

[16] G.E. Christensen. *Deformable Shape Models For Anatomy*. PhD thesis, Sever Institute of Technology, Washington University, Saint Louis, Missouri, August 1994.

[17] G.E. Christensen, R.D. Rabitt, and M.I. Miller. Deformable templates using large deformation kinematics. *IEEE transactions on image processing*, 5(10):1435–1447, 1996.

[18] C de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.

[19] P. Dierkcx. *Curve and Surface Fitting with Splines.* Oxford University Press, 1993.

[20] I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis.* John Wiley and Sons, 1998. New York.

[21] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des placques minces. *RAIRO Analyse Numérique*, 10:5–12, 1976. Paris.

[22] G.B. Folland. *Introduction to Partial Differential Equations.* Princeton University Press, 1995. Princeton, N.J.

[23] N. Heckman and J.O. Ramsay. Penalized regression with model-based penalties. University of British Columbia, 1999.

[24] D.L.G. Hill, D.J. Hawkes, N.A. Harrison, and C.F. Ruff. A strategy for automated multimodality image registration incorporating anatomical knowledge and imager characteristics. In H.H. Barrett and A.F. Gmitro, editors, *Information Processing in Medical Imaging*, volume 687 of *Lecture Notes in Computer Science*, pages 182–196. Springer-Verlag, Berlin, 1993.

[25] R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985. Cambridge.

[26] H. Kardestuncer. Basic concepts of the finite element method. In H. Kardestuncer, editor, *The Finite Element Handbook*. McGraw-Hill, 1987.

[27] B. Kim, B.J. Boes, K.A. Frey, S.K. Mukhopadhyay, and C.R. Ross, B.D. adn Meyer. Affine versus warping in coregistration of MR imaging and autoradiography. *Radiology*, 197:161–172, 1995.

[28] P. Lancaster and K. Šalkauskas. *Curve and Surface Fitting: an Introduction.* Academic Press, 1986.

[29] H. Lester and R.A. Simon. A survey of hierarchical non-linear medical image registration. *Pattern recognition*, 32:129–149, 1999.

[30] D.N. Levin, C.A. Pelizzari, G.T.Y. Chen, C.T. Chen, and M.D. Cooper. Retrospective geometric correlation of MR, CT and PET images. *Radiology*, 169:817–823, 1988.

[31] N. Locantore, J.S. Marron, D.G. Simpson, N. Tripoli, J.T. Zhang, and K.L. Cohen. Robust principal component analysis for functional data. *Test*, 8:1–65, 1999.

[32] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. In *Mathematical methods in biomedical image analysis*, pages 14–22. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[33] G.Q. Maguire Jr et al. Correlation methods for tomographic images using two and three dimensional techniques. In *Inform. Process. Med. Imaging, 9th International Conference*, 1985.

[34] J.B.A. Maintz, P.A. van den Elsen, and M.A. Viergever. Comparison of edge-based and ridge-based registration of CT and MR brain images. *Medical Image Analysis*, 1:151–161, 1996.

[35] J.B.A. Maintz and M.A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[36] R.J. Melosh and S. Utku. Principles for design of finite-element meshes. In H. Kardestuncer, editor, *The Finite Element Handbook*. McGraw-Hill, 1987.

[37] M. Merickel. 3D reconstruction: the registration problem. *Computer Vision, Graphics, and Image Processing*, 42:206–219, 1988.

[38] L. Mitáš and H. Mitášová. General variational approach to the interpolation problem. *Computers and Mathematics with Applications*, 16(12):983–992, 1988.

[39] S.J. Nelson, M.R. Day, P.J. Buffone, L.L. Wald, T.F. Budinger, R. Hawkins, W.P. Dillon, S. Huhn, M.D. Prados, S. Chang, and D.B. Vigneron. Alignment of volume MR images and high resolution [18f]fluorodeoxyglucose pet images for the evaluation of patients with brain tumours. *Journal of Computer Assisted Tomography*, 21(2):183–191, 1997.

[40] J. Oden and G. Carey. *Finite Elements: Mathematical Aspects.* Prentice-Hall, New Jersey, 1983.

[41] C.A. Pelizzari, G.T.Y. Chen, D.R. Spelbring, R.R. Weichselbaum, and C.T. Chen. Accurate three-dimensional registration of CT, PET, and/or MR images of the brain. *Journal of Computer Assisted Tomography*, 13:20–26, 1989.

[42] M.J.D. Powell. Tabulation of thin-plate splines on a very fine two-dimensional grid. In D. Braess and L.L. Schumaker, editors, *Numerical Methods of Approximation Theory.* Birkhaüser-Verlag, 1992.

[43] J.O. Ramsay. Estimating smooth monotone functions. *Journal of the Royal Statistical Society, Series B*, 60:365–375, 1998.

[44] J.O. Ramsay and B.W. Silverman. *Functional Data Analysis.* Springer-Verlag, 1997. New York.

[45] R. Renka. *Triangulation and Bivariate Interpolation for Irregularly Distributed Data Points.* PhD thesis, University of Texas at Austin, 1981.

[46] Larry L. Schumaker. *Spline Functions: Basic Theory.* John Wiley and Sons, Inc., 1981. New York.

[47] R. Sibson and G. Stone. Computation of thin-plate splines. *SIAM Journal of Statistical and Scientific Computing*, 12:1304–13, 1991.

[48] N. Sivakumar and J.D. Ward. On the least squares fit by radial functions to multidimensional scattered data. *Numerische Mathematik*, 65:219–243, 1993.

[49] M.P. Wand and M.C. Jones. *Kernel Smoothing*. Chapman and Hall, 1995.

[50] R.P. Woods, J.C. Maziotta, and S.R. Cherry. MRI-PET registration with automated algorithm. *Journal of Computer Assisted Tomography*, 17:536–546, 1993.

Vita

Name: Timothy Owen Ramsay

Place and year of birth: Montreal, 1967

Education: Queen's University, 1988-1993: B.Sc.H. 1993

Queen's University, 1993-1995: M.Sc. 1995

Awards: R. Samuel McLaughlin Fellowship: 1995

R. Samuel McLaughlin Fellowship: 1996

R. Samuel McLaughlin Fellowship: 1997

Ontario Graduate Scholarship in Science and Technology: 1998

Publications: *An Algorithm For Computing the Groebner Basis of a Binomial Ideal*; Master's Thesis, Queen's University