# Image Representation via Bivariate Spline Surface Smoothing

Cédric Beaulac

Université du Québec à Montréal

June 2, 2026

Joint work in collaboration with:

**Marie-Hélène Descary**
UQAM

**Mélanie Raymond**
UQAM

# Image Representation via Bivariate Spline Surface Smoothing

Functional data analysis
Image data
Motivation: edge detection
Bivariate splines
Applications and results

# Functional Data Analysis

# Functional Data

Functional data analysis (FDA) is a field of statistics concerned with studying data assumed to be realizations of smooth functions,

$$x(t),$$

or higher-dimensional objects such as surfaces,

$$x(s,t).$$

The domain may represent time, space, or more general multidimensional structures.

# Functional Data

In FDA, one observation $x_i(t)$ is therefore a function.

This contrasts with time series analysis. In FDA, a dataset consists of a collection of functions

$$S = \{x_i(t) : i = 1, \ldots, n\},$$

where all observations are defined on a common domain $T$.

In practice, the underlying function cannot be observed continuously and is only recorded at a finite number of sampling locations.

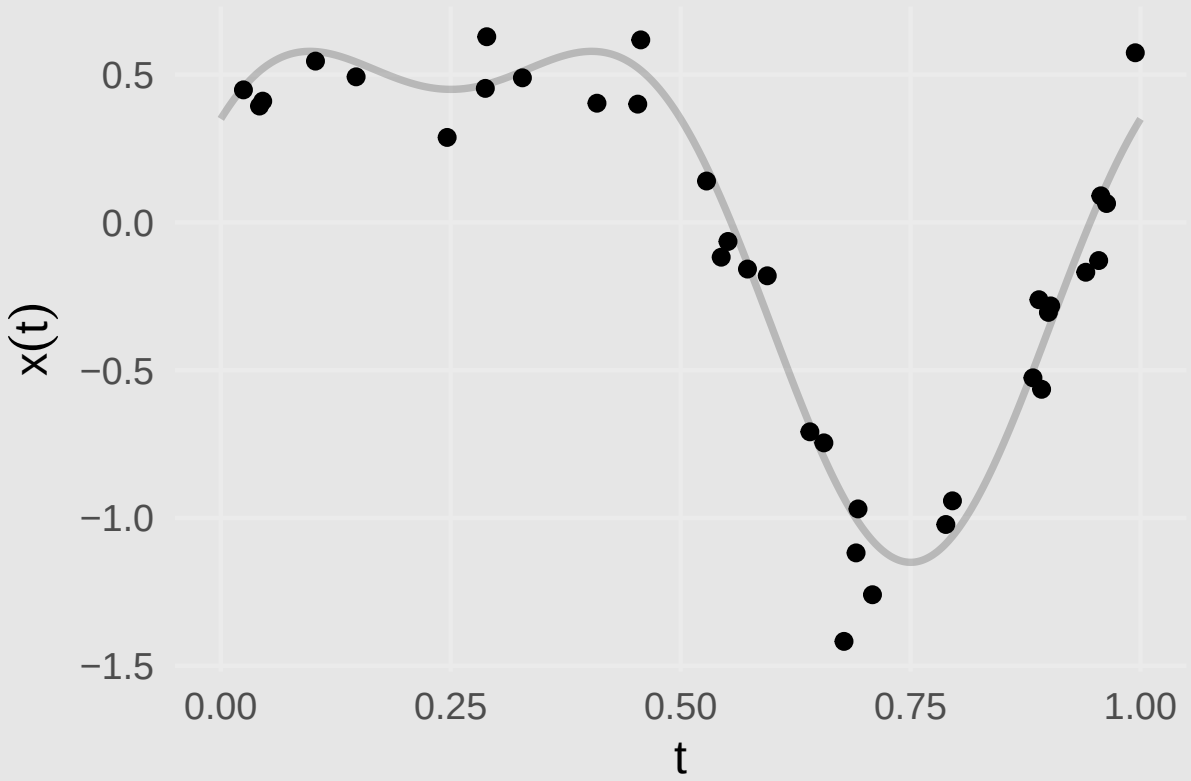



Figure: Discrete observations from an underlying smooth functional process.

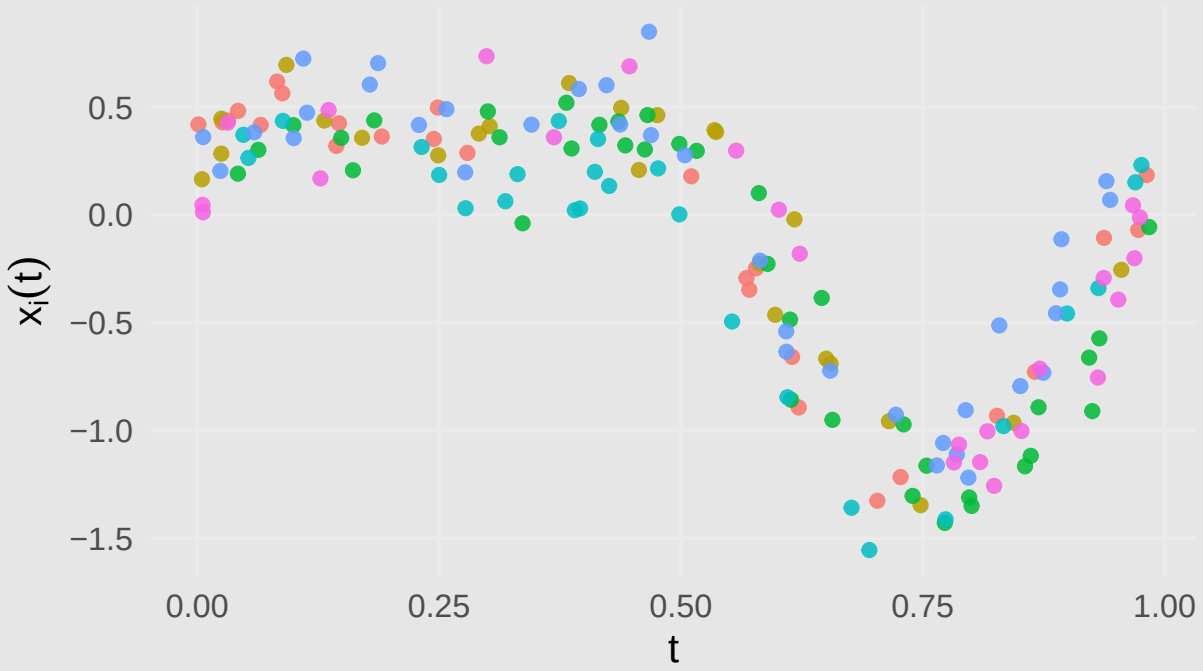



Figure: A sample of functional observations.

Note: The data may be discretely observed on different grids and sampled at a different number of locations.

# Smoothing and Functional Representation

FDA commonly assumes that the observed data arise from an underlying smooth process contaminated by measurement error.

Observed measurements are therefore often smoothed before analysis.

# Basis Expansions

A standard approach consists of representing functions using basis expansions,

$$\tilde{x}(t) = \sum_{k=1}^{K} c_k B_k(t),$$

where $\{B_k(t)\}$ is a collection of basis functions.

Common choices include:

- Fourier bases,
- B-splines,
- wavelets.

# Why Functional Representations?

Smoothing the data to recover a functional representation offers several advantages:

- Noise reduction through smoothing,
- Dimension reduction via basis coefficients,
- Continuous evaluation over the entire domain,
- Natural computation of derivatives,
- Analysis of irregularly sampled data.

# Basis Expansions

The representation

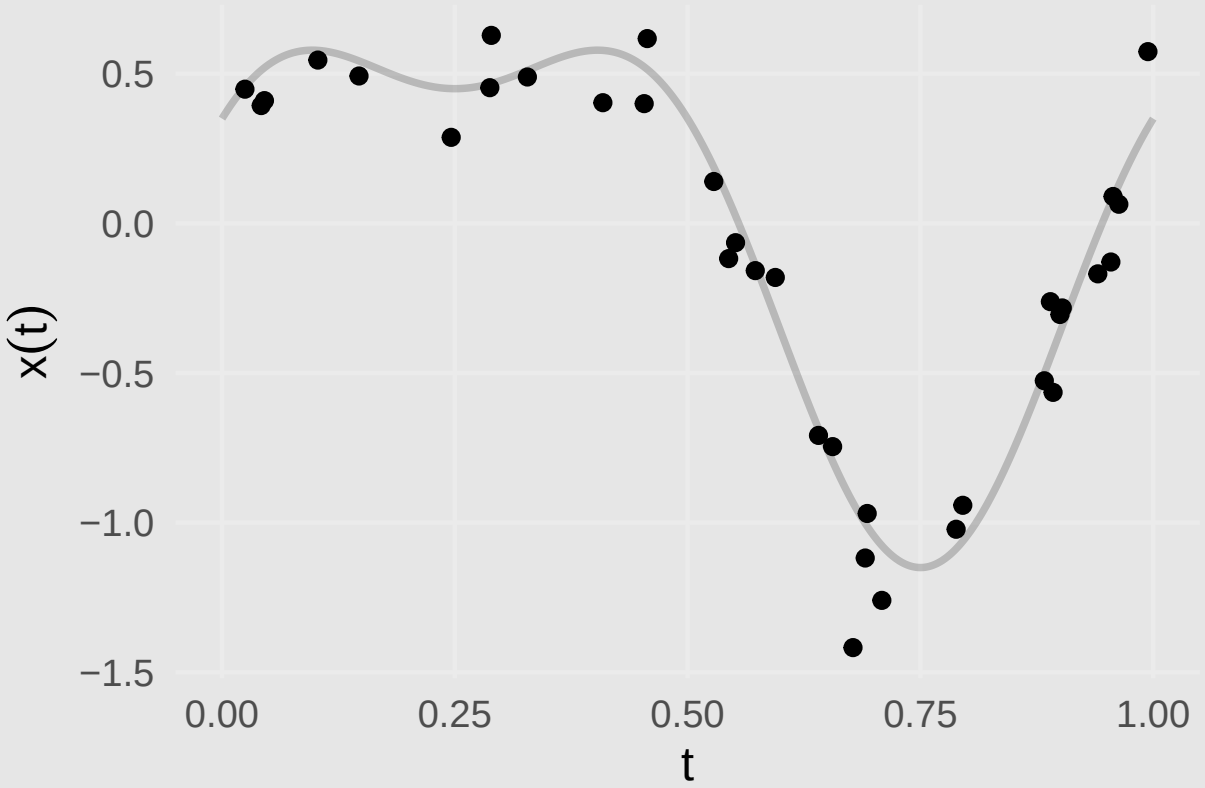
$$\tilde{x}(t) = \sum_{k=1}^{K} c_k B_k(t),$$

allows us to evaluate the function anywhere on the domain, even when the data $x_i(t)$ are observed on different grids, using only a small number of coefficients.

More importantly, learning such representations is usually straightforward and boils down to minimizing a mean squared error criterion:

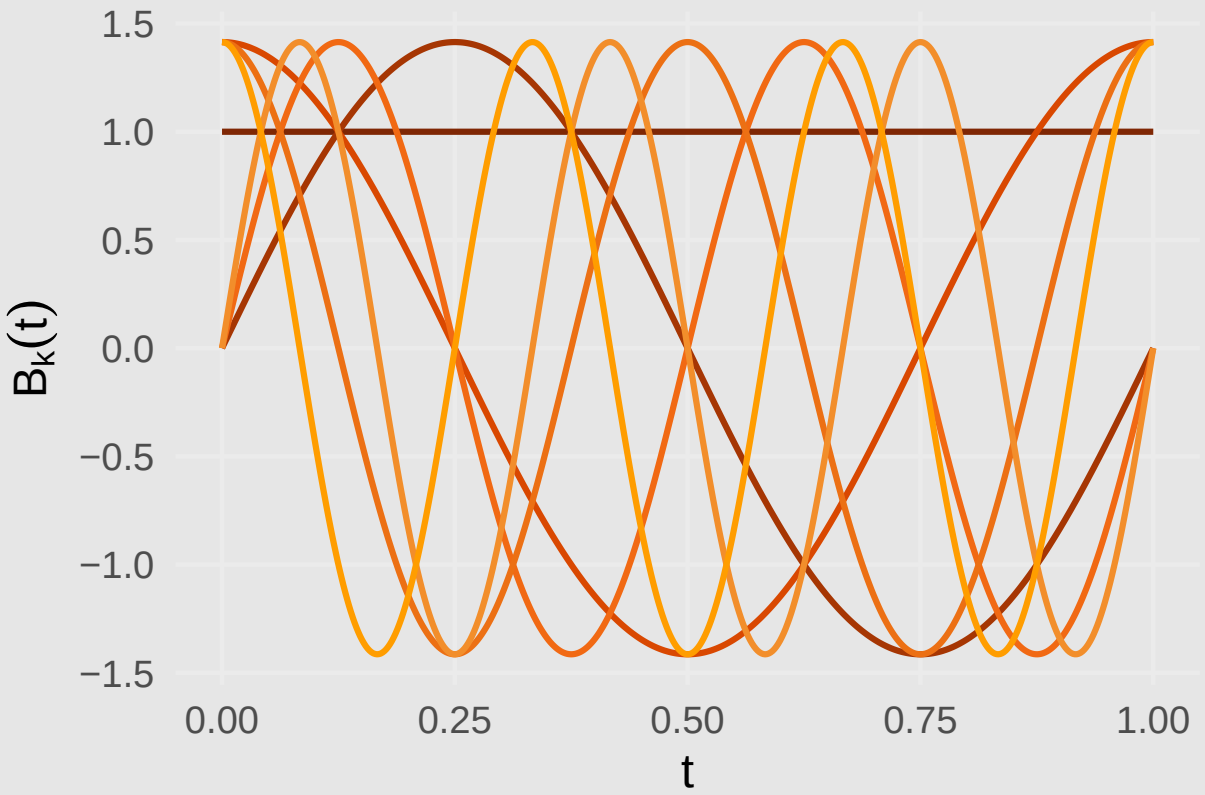
$$\sum_j \left[ x(t_j) - \sum_{k=1}^{K} c_k B_k(t_j) \right]^2 .$$

# Example: Functional Smoothing



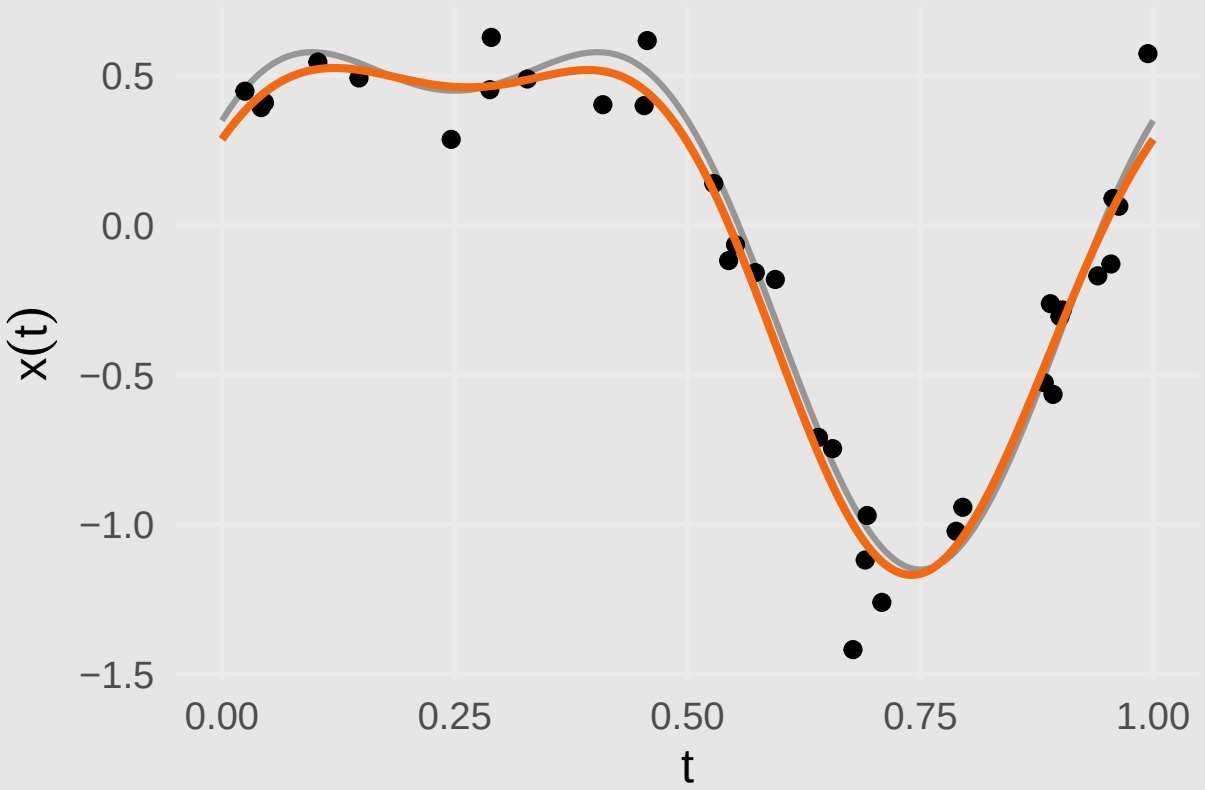

Underlying smooth process.

# Example: Fourier Basis



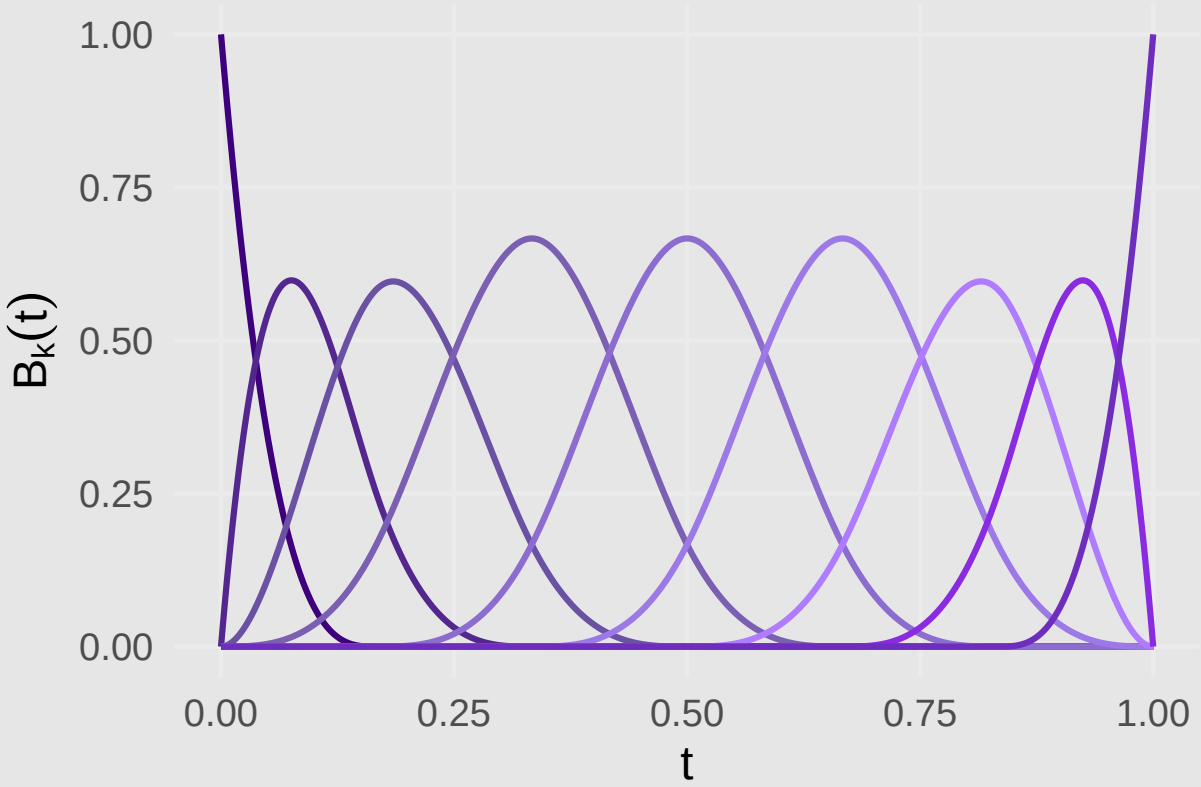

Fourier basis functions.

# Example: Smoothed Reconstruction



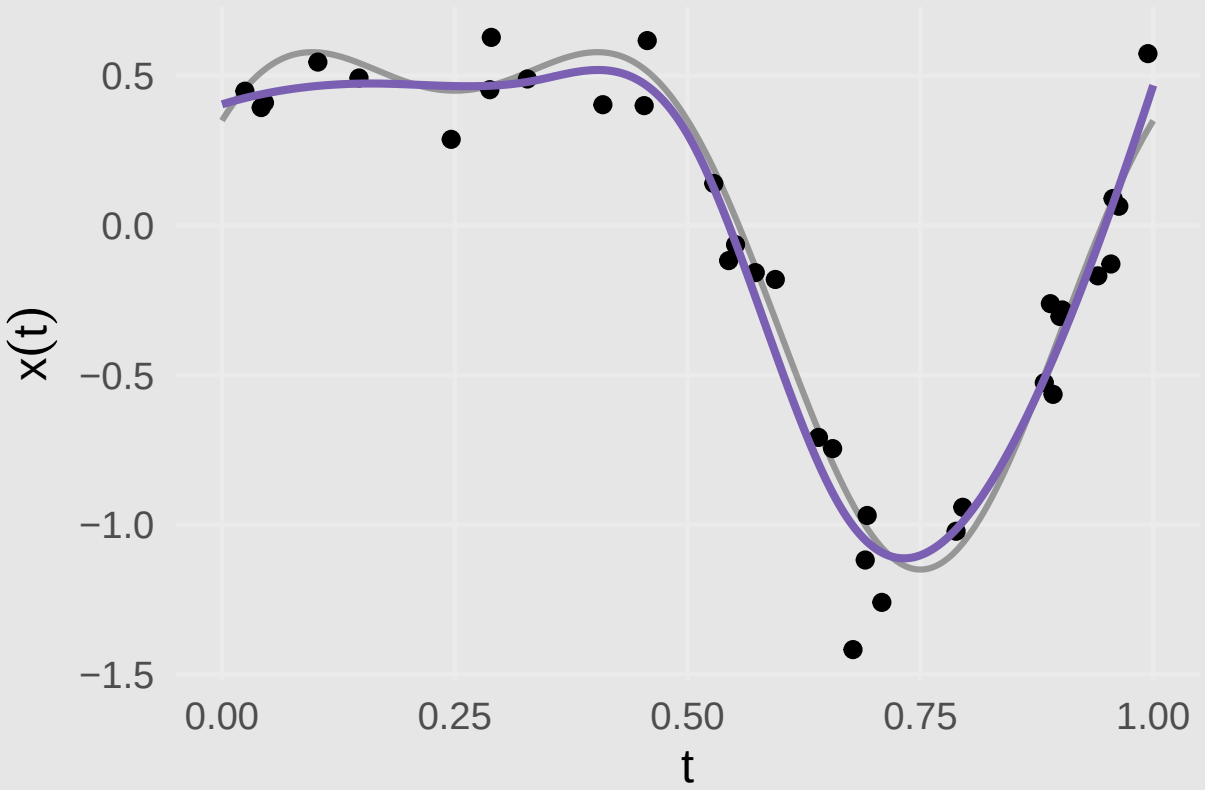

Recovered smooth functional representation.

# Example: B-Spline Basis




B-spline basis functions.

# Example: Smoothed Reconstruction




Recovered smooth functional representation.

# An Additional Ingredient: Penalization

So far, we have represented functions using flexible basis expansions:

$$\tilde{x}(t) = \sum_{k=1}^{K} c_k B_k(t).$$

Using many basis functions provides flexibility, but may also lead to overly rough representations.

An additional ingredient is therefore needed:

*penalization*.

# Smoothing Splines

A classical approach consists of penalizing the roughness of the estimated function directly.

Smoothing splines estimate $\tilde{x}(t)$ by minimizing

$$\sum_{i=1}^{n} [x_i - \tilde{x}(t_i)]^2 + \lambda \int [\tilde{x}''(t)]^2 \, dt.$$

The first term encourages fidelity to the observed data, while the second penalizes curvature.

The smoothing parameter $\lambda$ controls the trade-off between fidelity and smoothness.

# P-Splines

P-splines replace derivative penalties with penalties on neighboring spline coefficients.

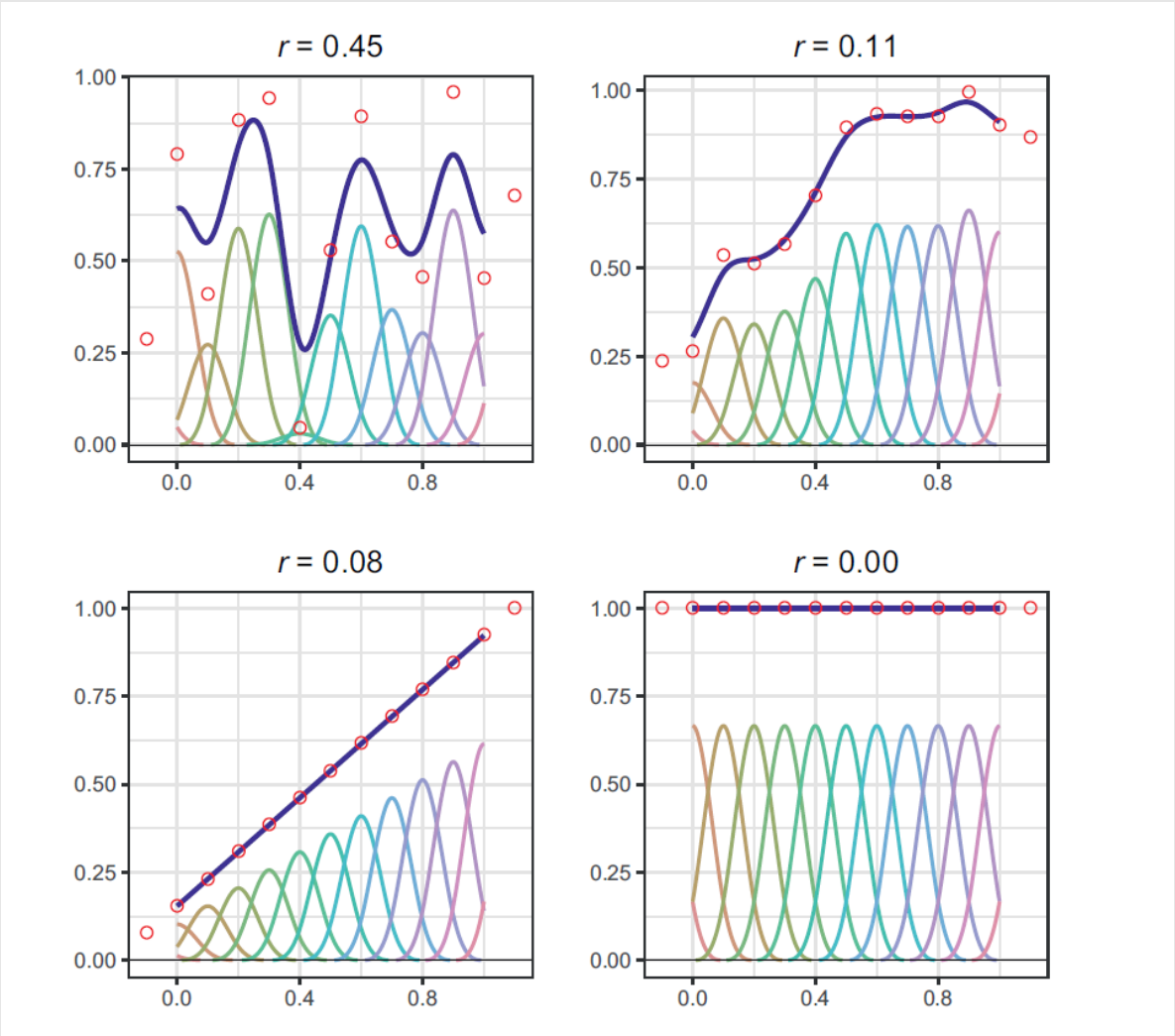
The coefficients are estimated by minimizing

$$\sum_{i=1}^{n} [x_i - \tilde{x}(t_i)]^2 + \lambda \sum_{k=2}^{K-1} (c_{k-1} - 2c_k + c_{k+1})^2.$$

The penalty discourages large local changes between neighboring coefficients.

Smooth functions therefore arise from smooth coefficient sequences.

# P-Splines Illustration




Adapted from Eilers & Marx (2021).

# P-Spline Matrix Formulation

The second-order difference penalty may be written compactly using a finite-difference matrix $D$:

$$\|x - Bc\|^2 + \lambda \|Dc\|^2.$$

For example,

$$D = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix}.$$

The matrix formulation is computationally convenient and naturally extends to tensor-product surfaces.

# Functional Data Analysis

Once represented as smooth functions, observations can be analyzed using extensions of classical statistical tools.

For instance, functional data can serve as predictors in a functional regression setting:

$$y_i = \alpha + \int_T \beta(t) x_i(t)\, dt + \varepsilon_i.$$

# Functional Data Analysis

Functional data can also be used as responses:

$$y_i(t) = \mu(t) + \beta(t)x_i + \varepsilon_i(t).$$

Models have also been developed to study function-on-function regression problems:

$$y_i(t) = \alpha(t) + \int_T \beta(s,t)x_i(s)\,ds + \varepsilon_i(t).$$

# Functional Data Analysis

To circle back to the earlier example of butterfly shape analysis, we can perform functional principal component analysis (FPCA).

FPCA projects functions onto a lower-dimensional space and identifies regions ($t$) of high variability within the dataset.

# Functional Principal Component Analysis

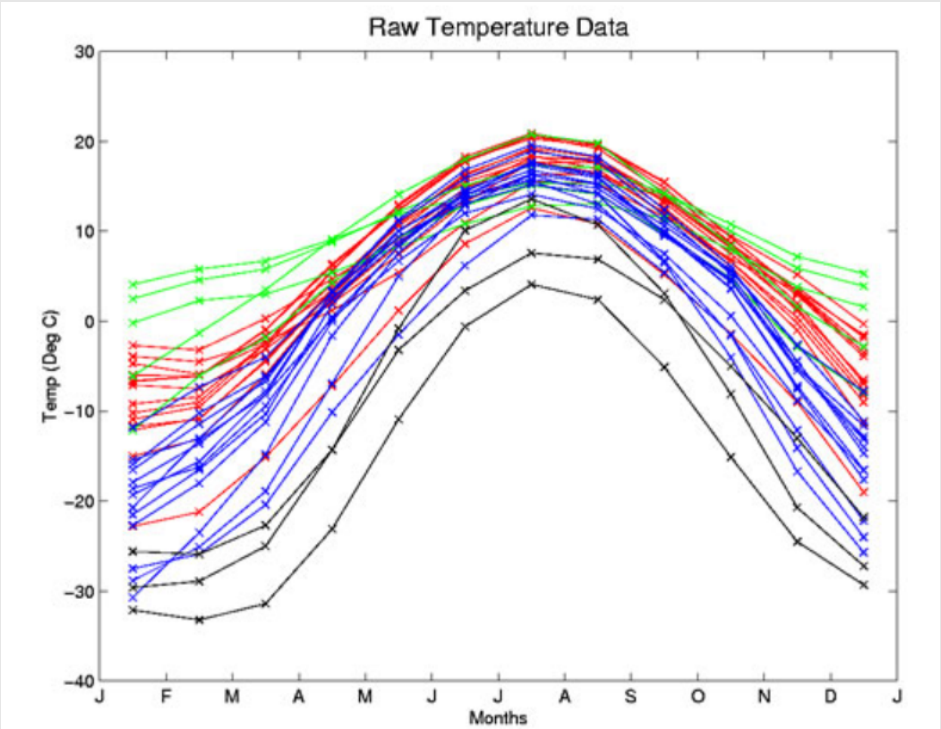



Figure: Daily temperature curves recorded across weather stations.

# Functional Principal Component Analysis

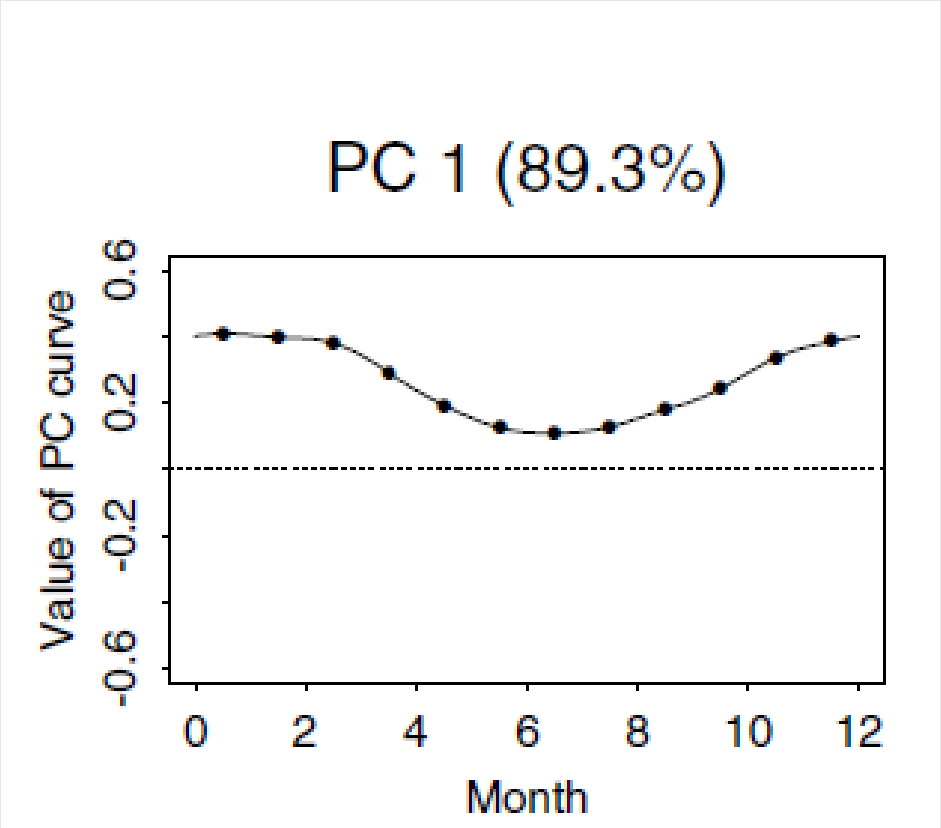



Figure: First functional principal component.

# Image Data

# Image Data

Images have been natively captured and stored in matrix format since cameras became digital.

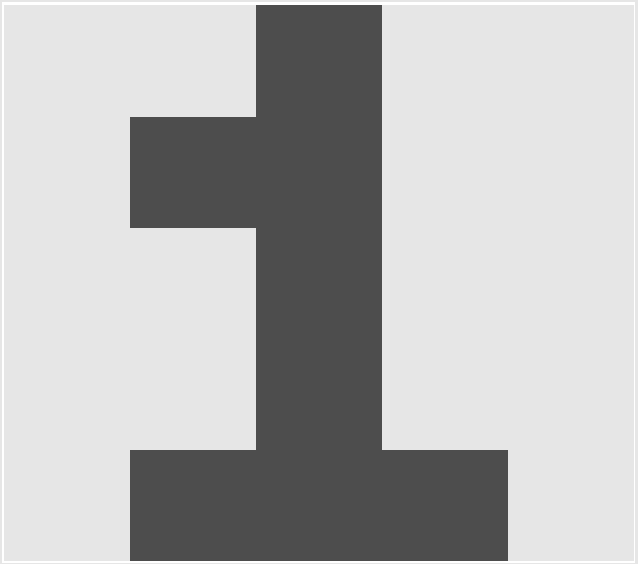
The element $(i, j)$ represents the color intensity at pixel $[i, j]$.

For black-and-white or grayscale images, the color intensity is an integer in the range $[0, 255]$.

A color image is typically represented using three matrices with integer entries in $[0, 255]$.

# Image Data

For black-and-white or grayscale images, the pixel intensity is an integer in the range $[0, 255]$.



*Black-and-white image of the digit '1'*

$$\begin{bmatrix} 255 & 255 & 0 & 255 & 255 \\ 255 & 0 & 0 & 255 & 255 \\ 255 & 255 & 0 & 255 & 255 \\ 255 & 255 & 0 & 255 & 255 \\ 255 & 0 & 0 & 0 & 255 \end{bmatrix}$$

*Matrix representation (pixel intensities)*

# Images as Surfaces

We can also think of an image as a surface, where the height at location $(i, j)$ corresponds to the color intensity.

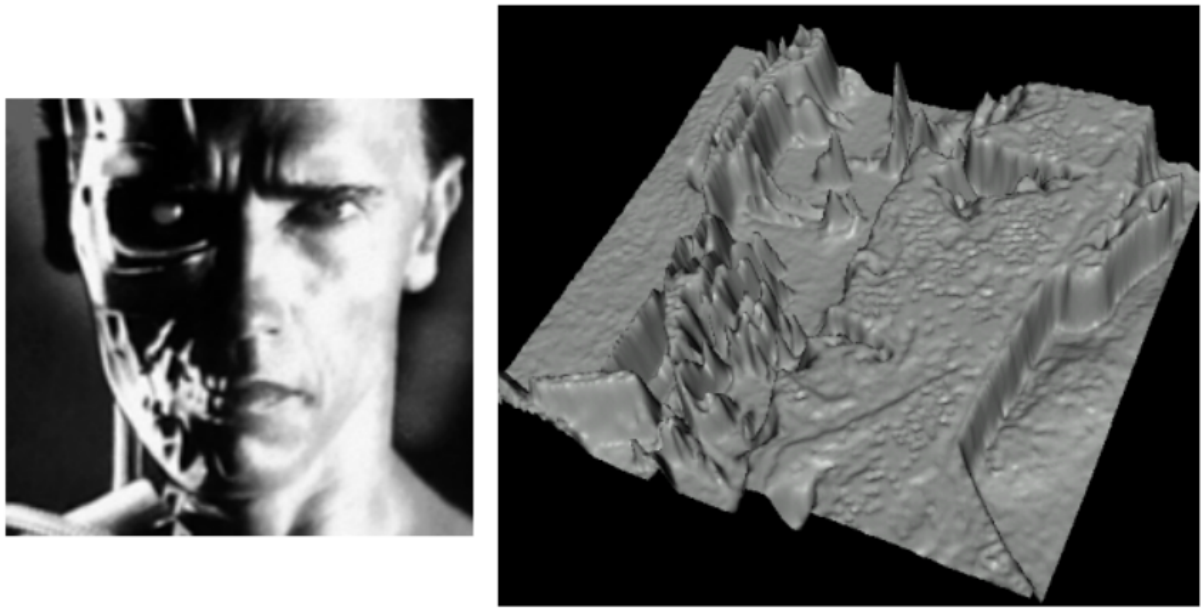


Figure: An image on the left and its associated surface representation on the right.

# Image Data Analysis

Typical approaches for image analysis are designed to analyze matrices.

Filtering and convolution are matrix operations that perform linear combinations of neighboring pixels.

Powerful predictive models can then be built by learning convolution weights within broader machine learning frameworks.

More recently, there has also been interest in developing transformer models for image analysis.

# Image Data Analysis

Even though these approaches can perform extremely well in predictive tasks, pixel-based approaches still present several challenges.

- Difficult interpretation,
- Large data objects (high-resolution images and videos),
- Generalization issues (sensitivity to resolution and acquisition technology).

# Our Solution

We want to stop looking at images as simple collections of pixels.

Instead, images can be analyzed as collections of objects defined by their shapes, textures, and colors.

Over the last few years, we developed a framework for the analysis of shapes extracted from images.

# Edge Detection and Shape Analysis

# Edge Detection and Shape Analysis

The first step in analyzing shapes in images is to extract them.

For this, we use contour detection techniques.

Edge and contour detection can provide us with flood-filled images (or masks).

This is not an easy task.

# Edge and Contour Detection

Assume we have an image containing an object of interest and a mask obtained through edge detection.

Photo of a bat.

*Mask image.*

# Planar Closed Curves

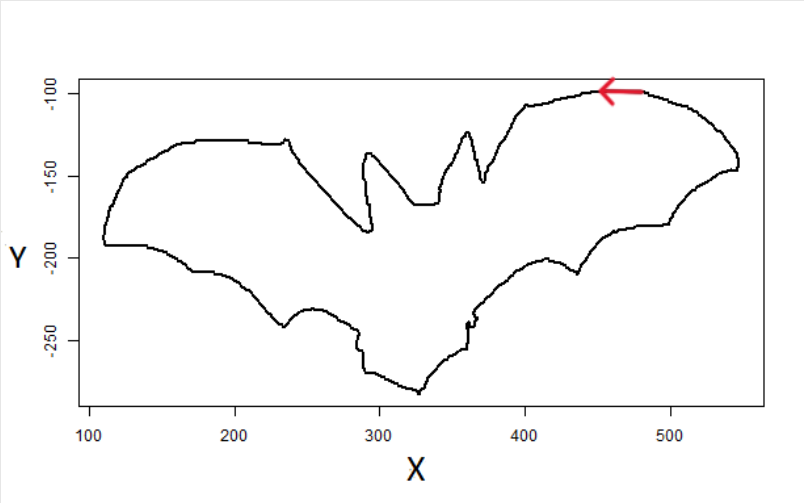
We extract a functional representation for the contour (a planar closed curve), represented through coordinates:

$$C(t) = (X(t), Y(t)),$$

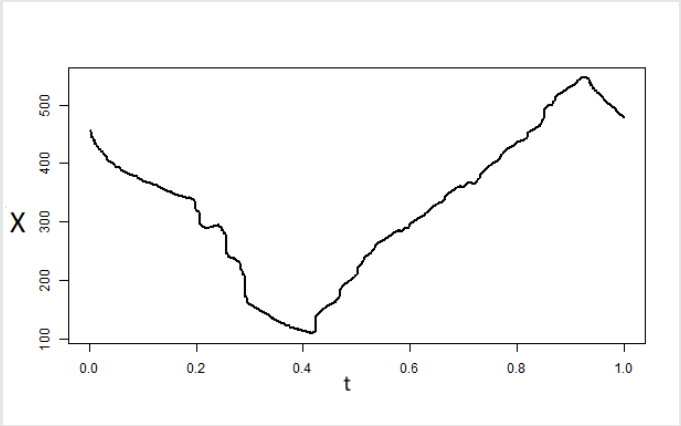
where $t \in [0, 1]$ represents the proportion of the curve traveled from the start ($t = 0$) to the end ($t = 1$).

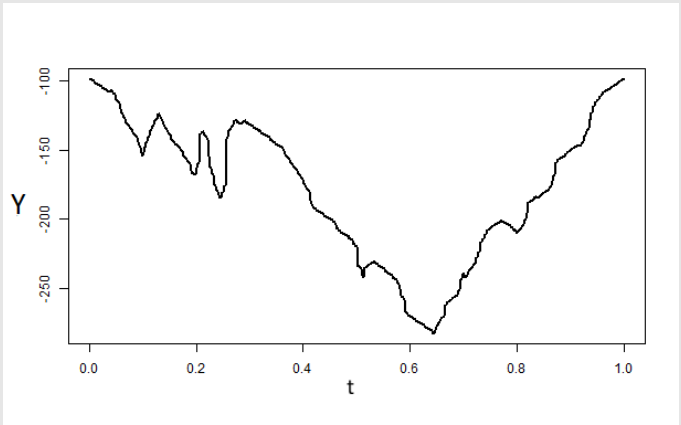
For closed curves, $C(0) = C(1)$.

# Traveling Along the Contour




Contour of the bat.



# Shape as Functional Data

We model closed curves as bivariate functional data.

Smoothing the data solves the problem of images and objects having different resolutions (irregular sampling).

Fourier bases provide a way to solve the complex alignment problem using transfer matrices.

Once the contours are aligned, we can analyze shapes as functional data.

# Alignment on Real Data

*MPEG-7* database:

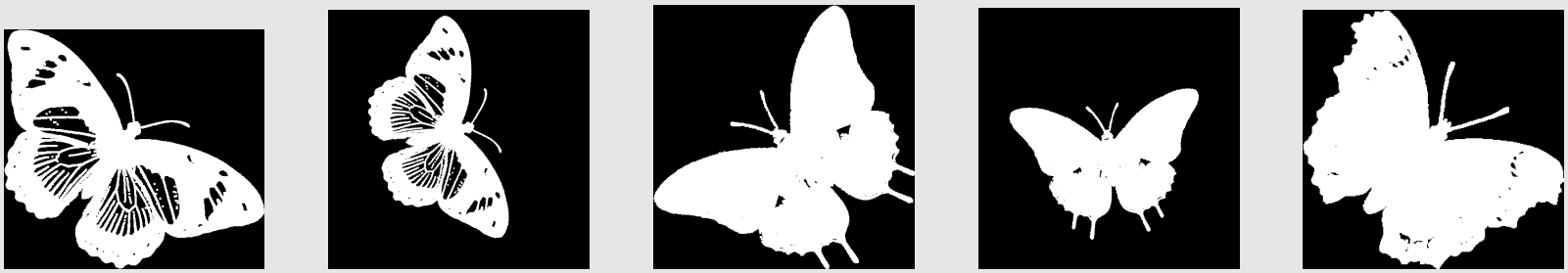
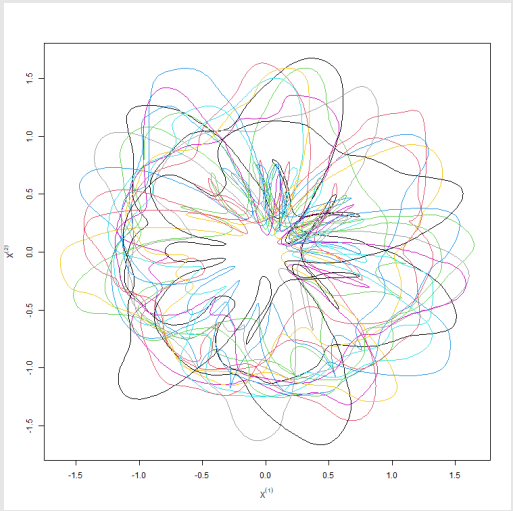
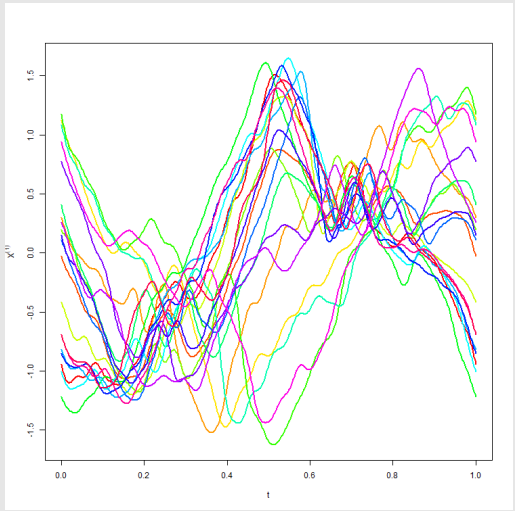
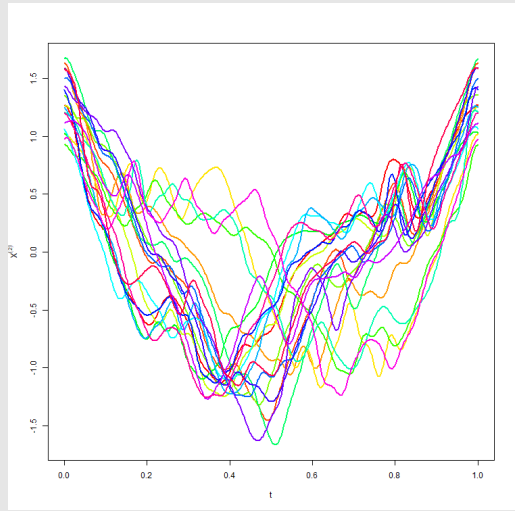
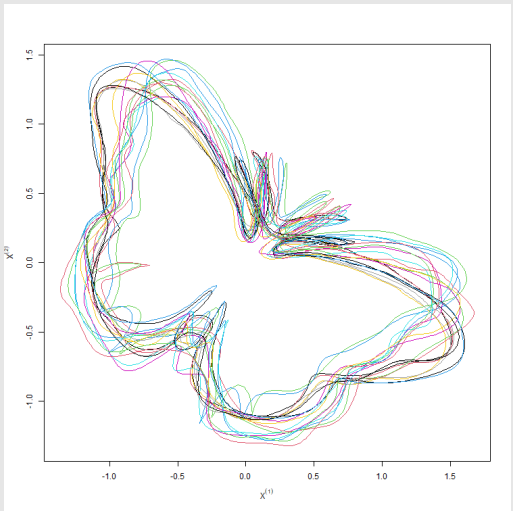
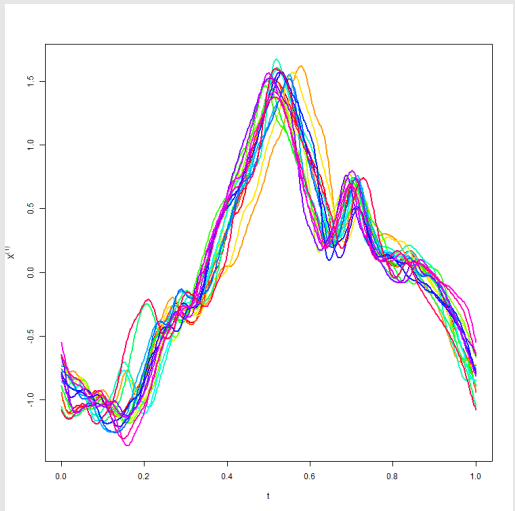
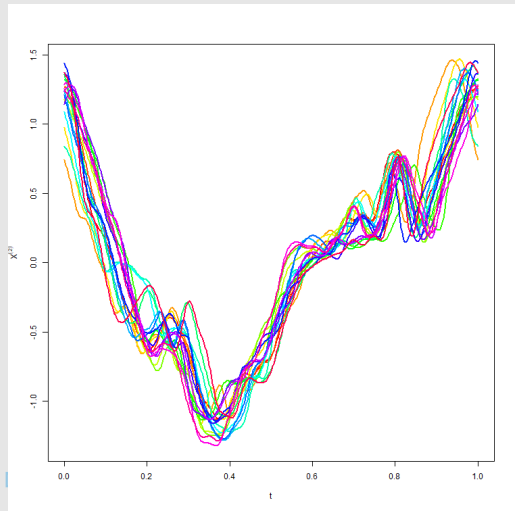


Figure: Examples of butterfly images from the database.

# Alignment on Real Data

# Functional Principal Components

PC1 (57%) PC2 (22%) PC3 (9%)

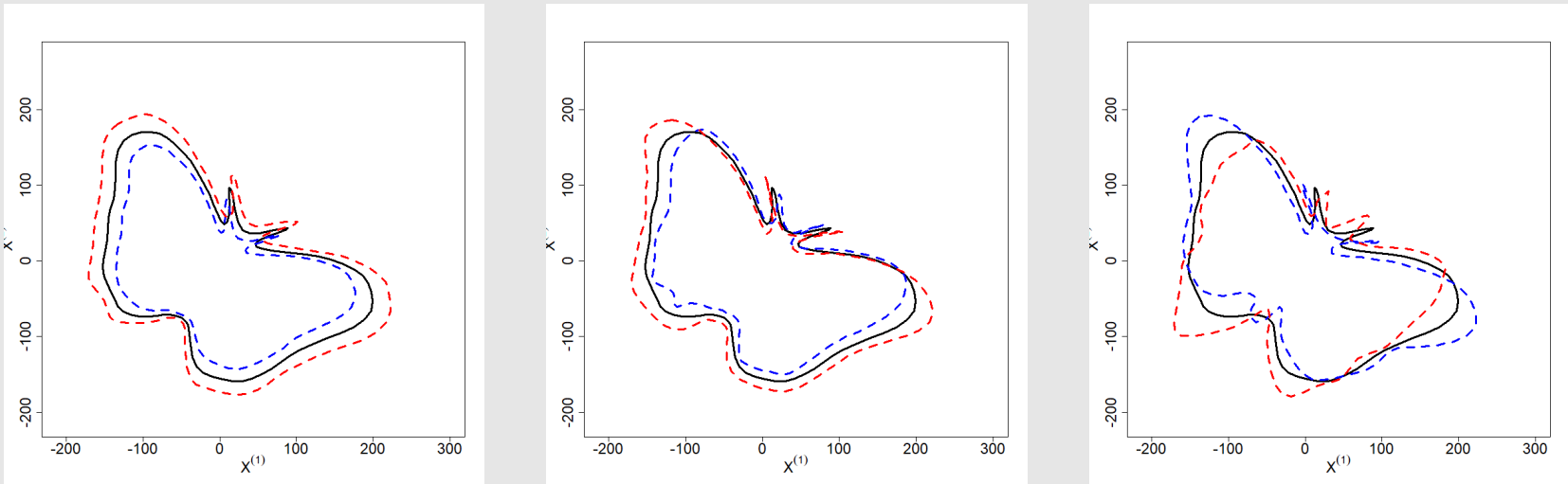



Figure: Plots of the estimated mean function $\bar{\mathbf{z}}$ in black, $\bar{\mathbf{z}} - 20\hat{\phi}_k$ in blue, and $\bar{\mathbf{z}} + 20\hat{\phi}_k$ in red for $k = 1, 2, 3$.

# Motivation for Surface Representations of Images

We represent contours as bivariate functions for statistical shape analysis.

Such approaches require extracting contours from images through image segmentation techniques.

These techniques identify large changes in images by computing local image derivatives directly from the pixel grid using discrete operators.

This raises the following question: if we represent images as surfaces, could this provide more accurate derivatives?

# Bivariate Splines

# From Curves to Surfaces

In one dimension, a functional observation may be smoothed and represented as

$$\tilde{x}(t) = \sum_{k=1}^{K} c_k B_k(t).$$

The coefficients $c_1, \ldots, c_K$ are learned by minimizing some criterion.

These same ideas naturally extend to higher-dimensional functions such as surfaces

$$x(s, t).$$

# Tensor-Product B-Splines

A smooth surface can be represented as

$$\tilde{x}(s,t) = \sum_{k=1}^{K_s} \sum_{\ell=1}^{K_t} \theta_{k\ell} B_k(s) B_\ell(t).$$

Here, $B_k(s)$ and $B_\ell(t)$ are one-dimensional B-spline basis functions along the two spatial directions.

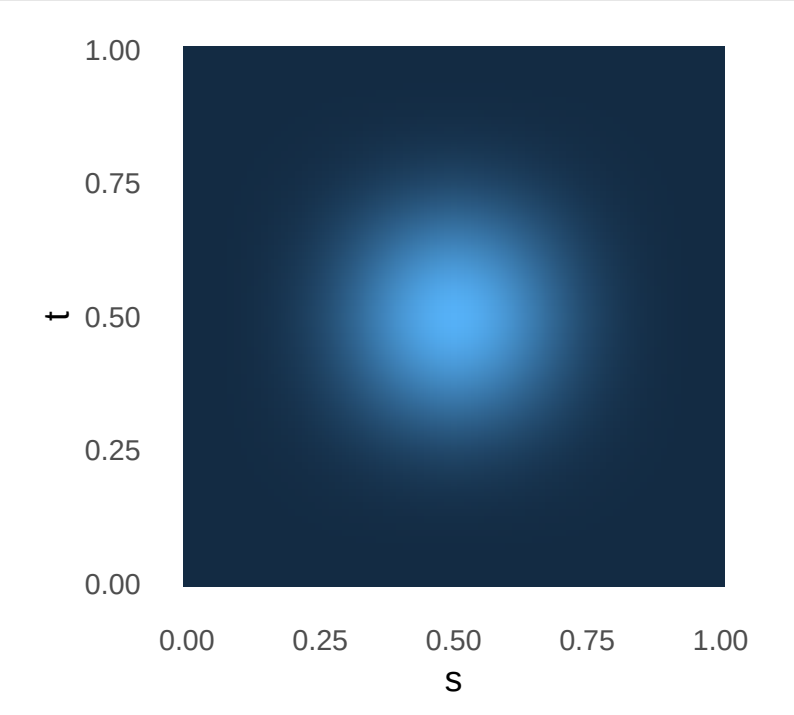
The coefficients $\theta_{k\ell}$ determine the shape of the smooth surface.

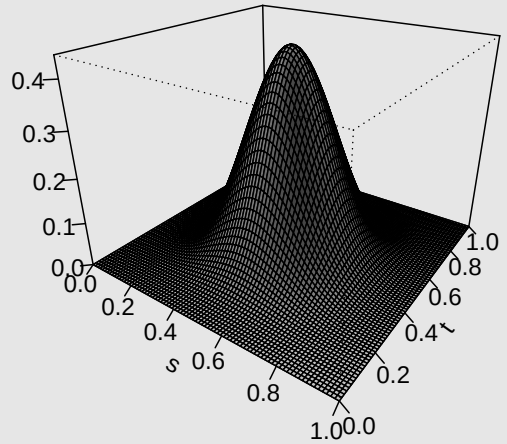
# A Single Tensor-Product Basis Function

A tensor-product B-spline basis function

$$B_k(s)B_\ell(t)$$

defines a localized smooth surface over the domain.



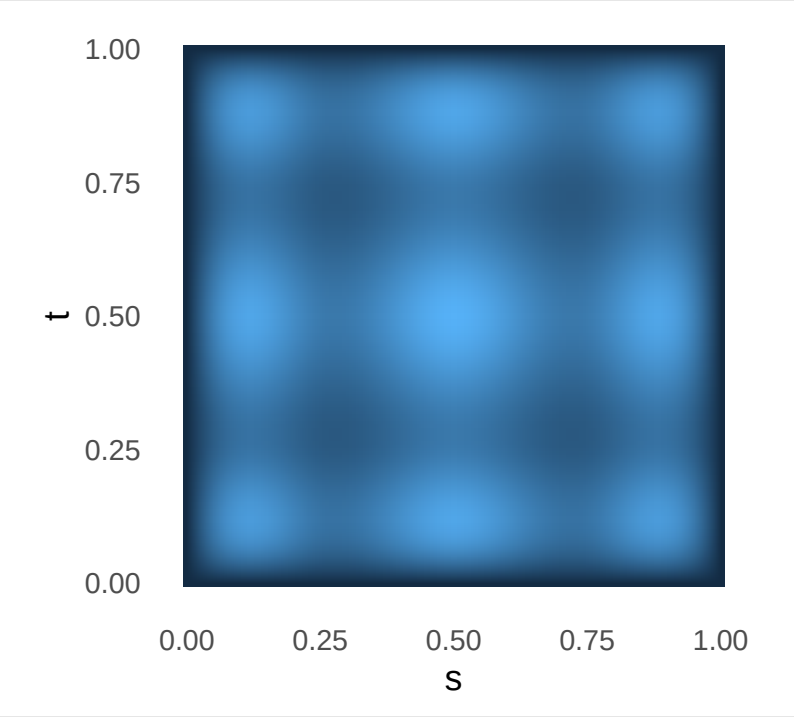



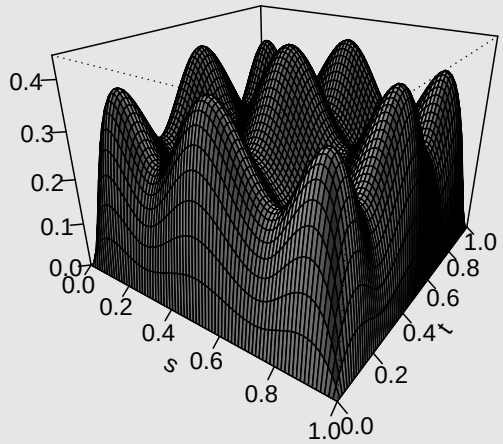

A single tensor-product basis function represented as a heatmap and as a

# Combining Tensor-Product Basis Functions

Smooth surfaces are obtained through linear combinations of many localized tensor-product basis functions.

The tensor-product structure naturally extends one-dimensional spline representations to two-dimensional domains.



# Tensor-Product Surface Representation

Suppose that the surface is evaluated on grids

$$s_1, \ldots, s_{n_s} \quad \text{and} \quad t_1, \ldots, t_{n_t}.$$

The smooth surface representation may then be written in matrix form as

$$\tilde{X} = B_1 \Theta B_2^T,$$

where:

- $B_1$ and $B_2$ are B-spline basis matrices,
- $\Theta$ is the tensor-product coefficient matrix.

# Penalized Tensor-Product Surfaces

For tensor-product surfaces,

$$\tilde{X} = B_1 \Theta B_2^T,$$

penalization can be applied independently along each spatial direction.

A common formulation is

$$\|\tilde{X} - X\|_F^2 + \lambda_s \|D_s \Theta\|_F^2 + \lambda_t \|\Theta D_t^T\|_F^2.$$

The two smoothing parameters independently control roughness along the horizontal and vertical directions.

This naturally leads to efficient tensor-product smoothers for image representations.

# Functional Surface Representations for Images

With these tools, we can fit a smooth surface representation of an image.

The pixel grid is viewed as a discrete observation of an underlying smooth surface.

The remaining challenge is therefore to estimate the coefficient matrix $\Theta$, typically by minimizing a mean squared reconstruction criterion over the observed pixels.

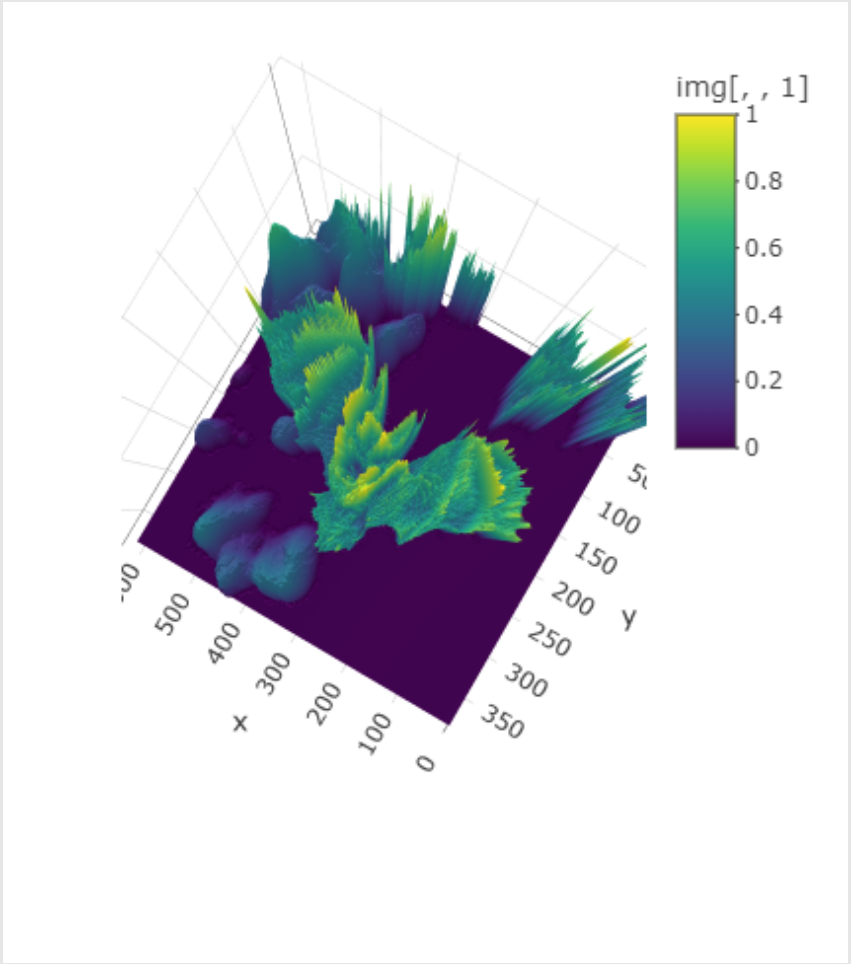
# Functional Surface Representations for Images

This surface representation offers several advantages:

- continuous image representations allowing instantaneous changes in resolution,
- **analytical spatial derivatives that may be useful for image segmentation and contour detection,**
- noise reduction through smoothing (if desired),
- resolution-independent representations,
- low-dimensional coefficient representations.

This motivates studying images as smooth surfaces rather than purely discrete pixel arrays.

# Functional Surface Representations for Images



# Applications and results

# Back to the Motivation

Edge detection:

Photo of a bat.

*Mask image.*

# A Classical Image Processing Pipeline

Many edge-detection methods rely on local image derivatives.

A typical pipeline consists of:

1. smoothing the image to reduce noise,
2. approximating spatial derivatives using discrete operators,
3. computing gradient magnitudes to identify sharp intensity changes.

For example, the Sobel operator approximates derivatives using local convolution masks:

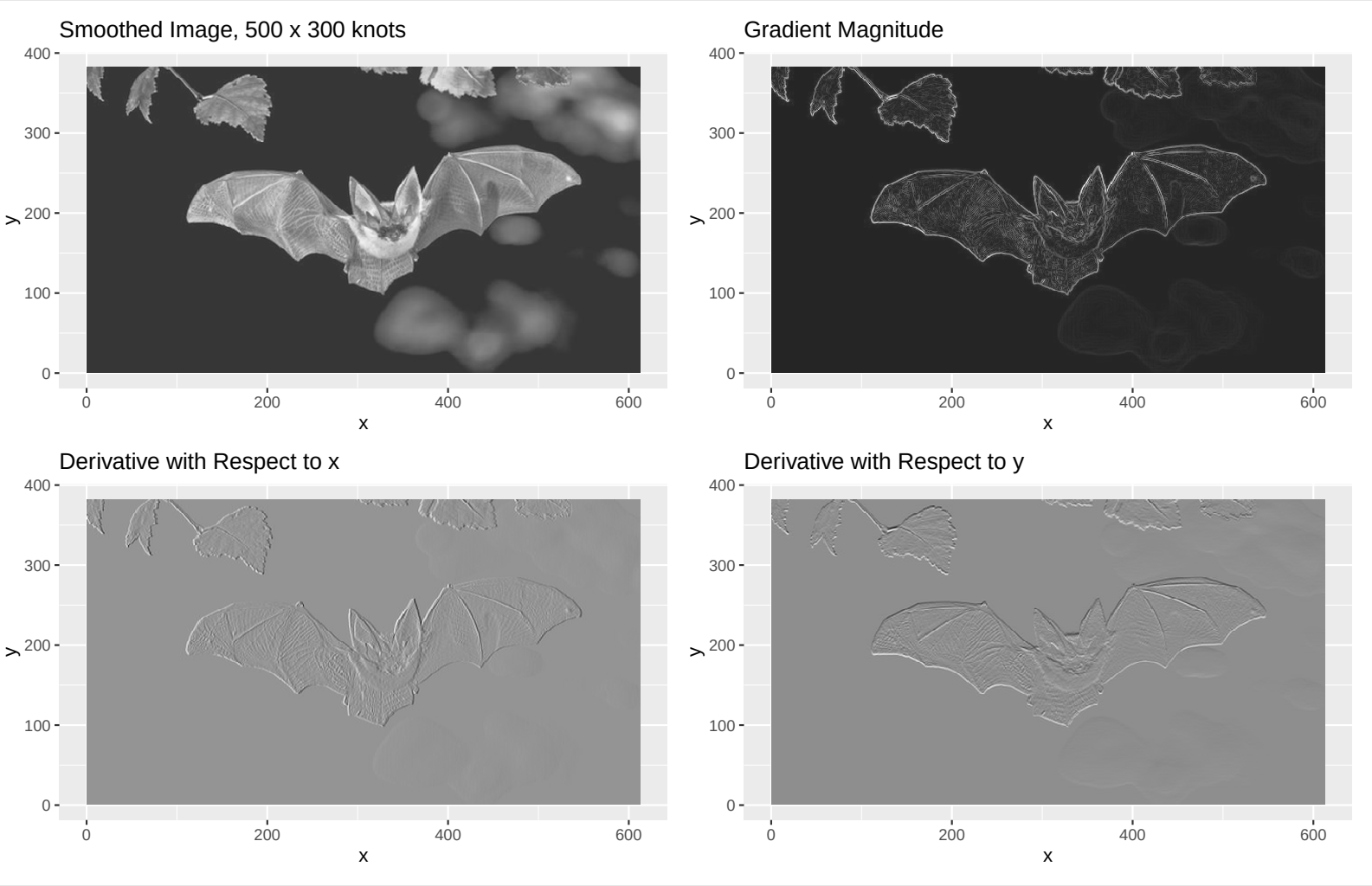
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

# Back to the Motivation

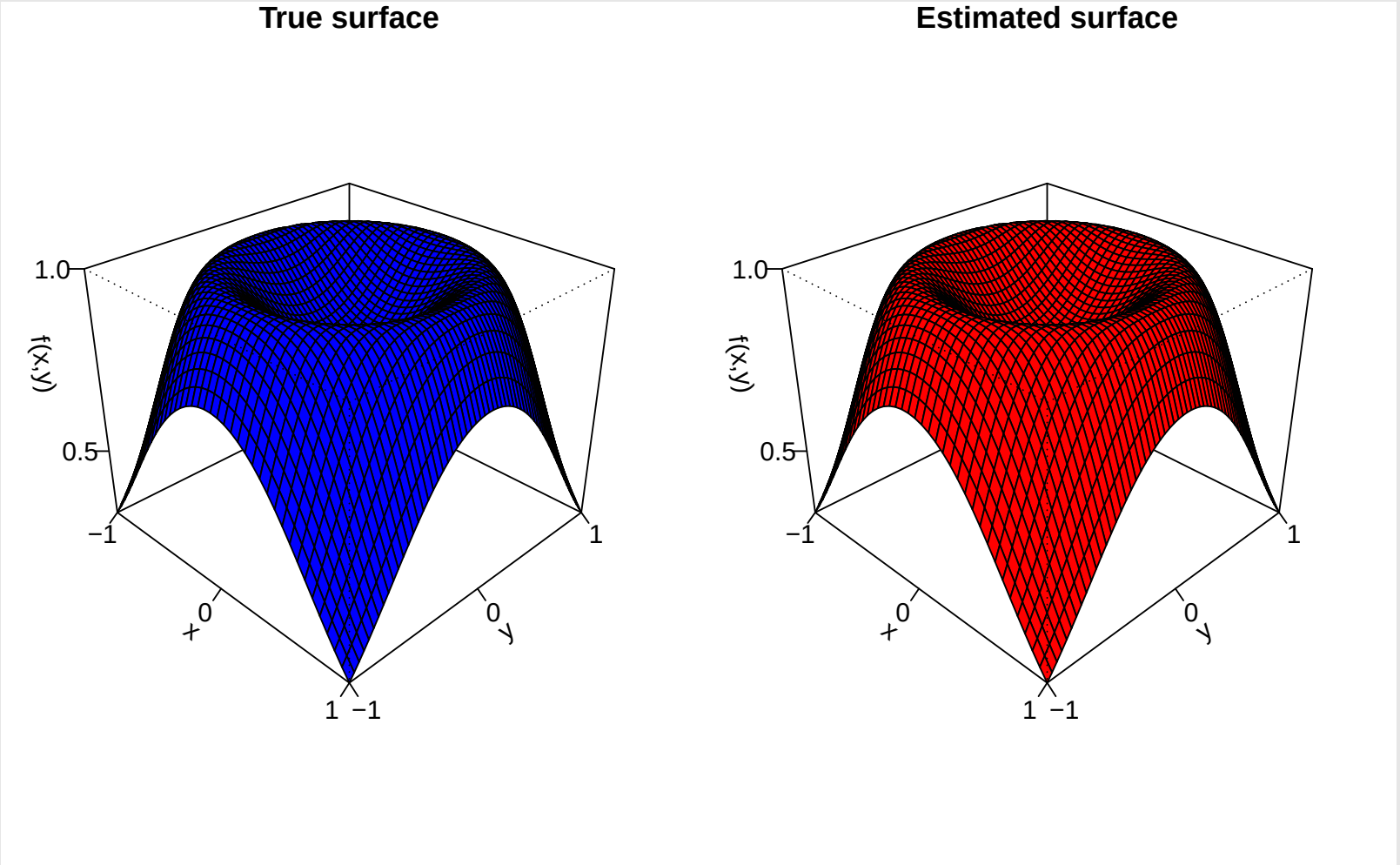
Assuming images can be viewed as surfaces, it feels natural to represent them as such and propose a similar pipeline:

1. smoothing the image to reduce noise (using a bivariate spline approach),
2. directly computing spatial derivatives analytically,
3. computing gradient magnitudes to identify sharp intensity changes.

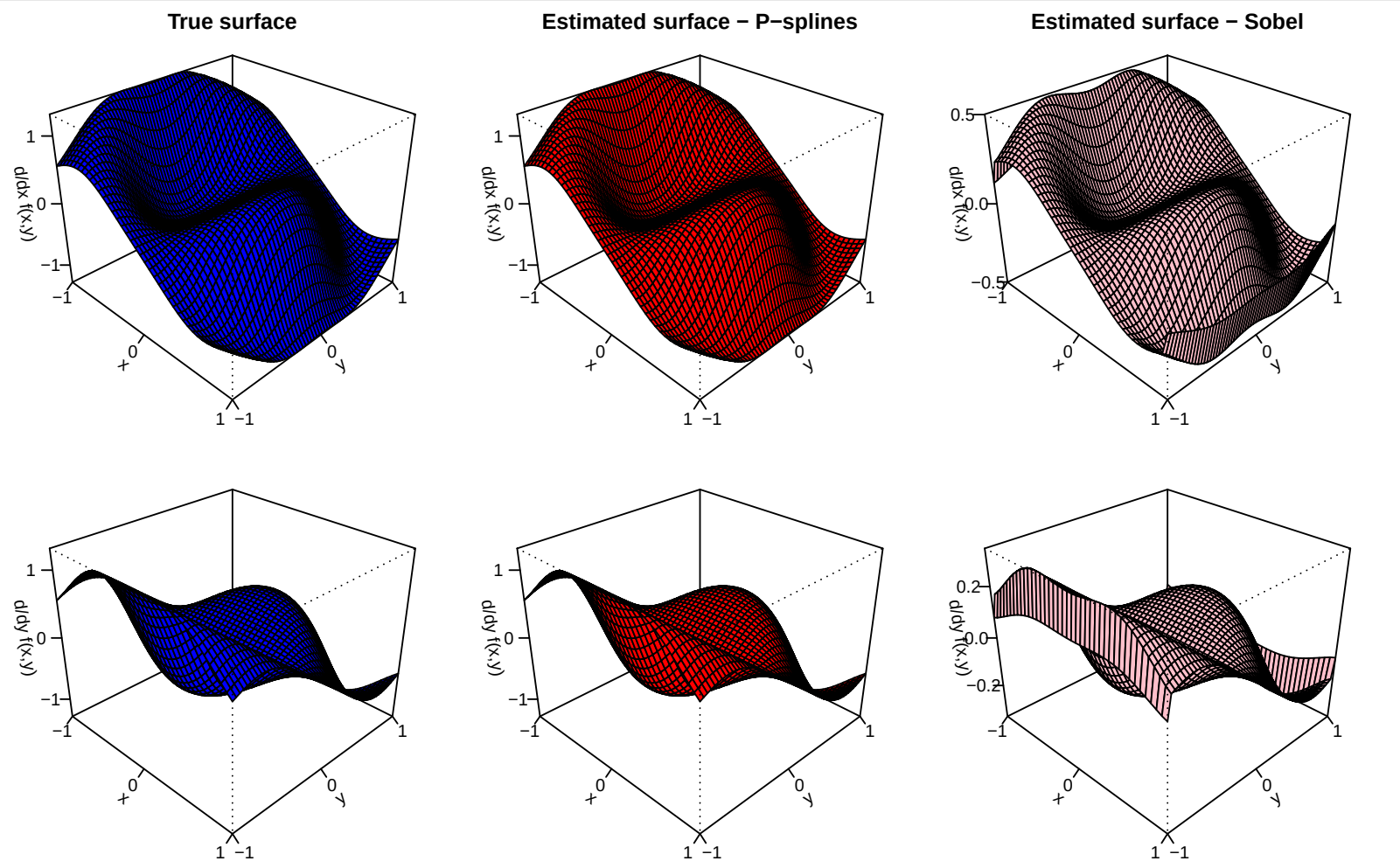
# Image Smoothing and Derivatives



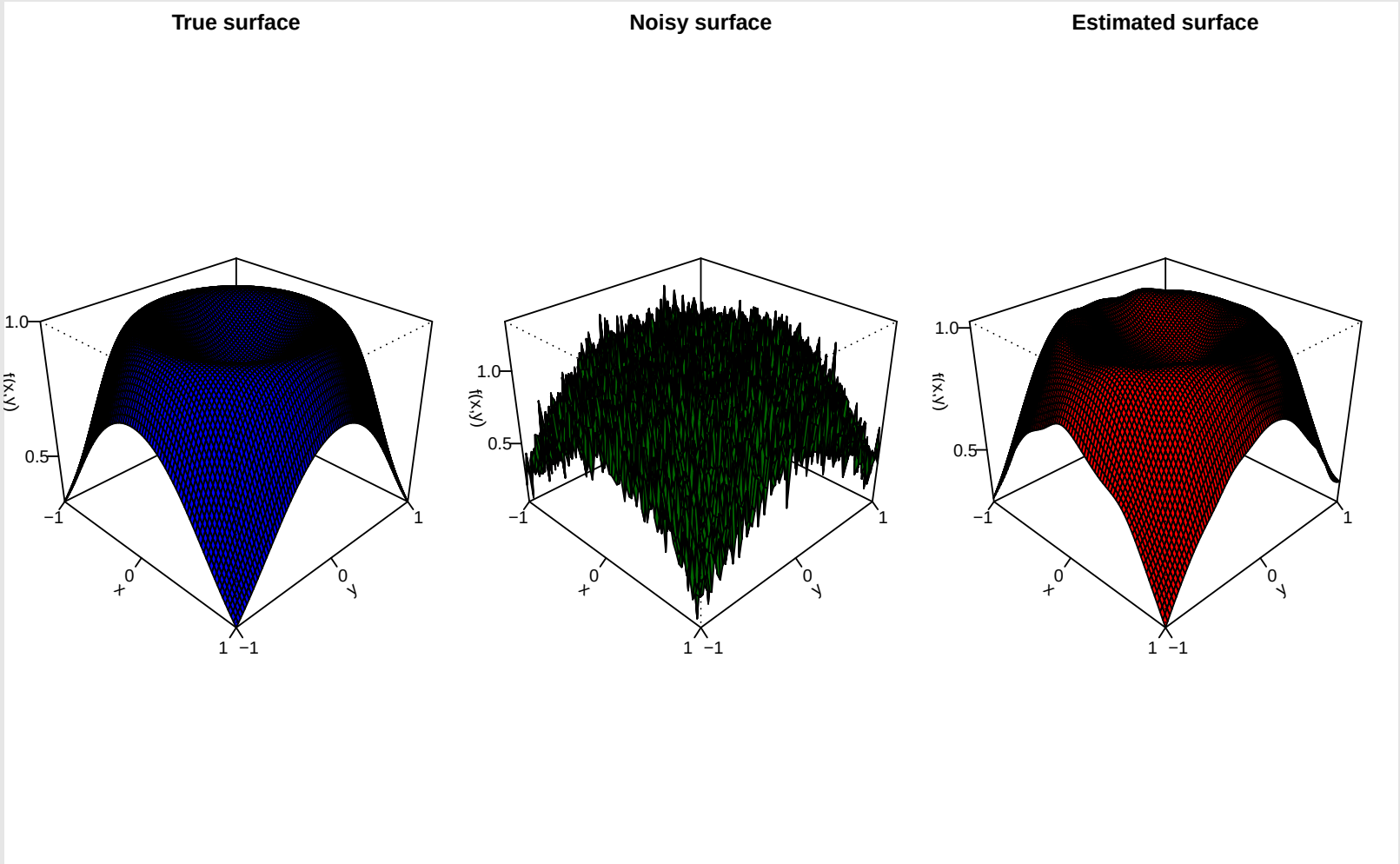
# Surface and Derivatives



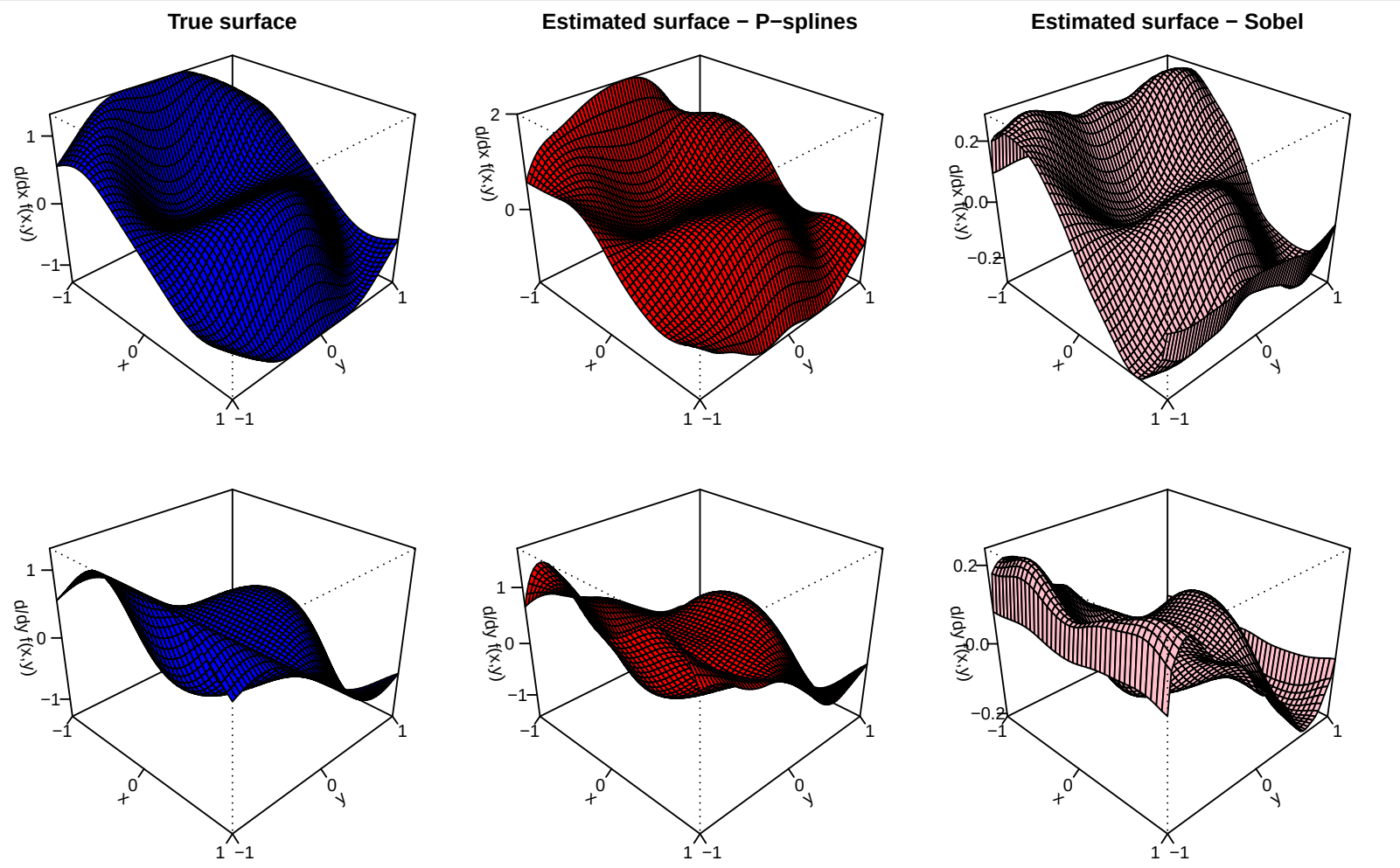
# Surface and Derivatives



# Surface and Derivatives



# Surface and Derivatives



# Results

The results are mitigated.

Splines are good smoothers, and it is not clear that images necessarily need to be smoothed.

Discrete approximations of derivatives are already quite robust.

# What Do We Do Now?

Are pixels measured with error? Do we need penalization or smoothing?

We are considering splines on triangulations or returning to wavelets and Fourier transforms.

Maybe using this surface representation is usefull for the analysis (regression) of images.

I would love to answer your questions.

J. O. Ramsay and B. W. Silverman. Functional Data Analysis (Second Edition). Springer, 2005.

P. H.C. Eilers, and B.D. Marx. Practical smoothing: The joys of P-splines. Cambridge University Press, 2021.

H. Mantz, K. Jacobs and K. Mecke, Utilizing Minkowski functionals for image analysis: a marching square algorithm, Journal of Statistical Mechanics: Theory and Experiment, 2008.

Xiao, Luo, Yingxing Li, and David Ruppert. "Fast bivariate P-splines: the sandwich smoother." Journal of the Royal Statistical Society Series B: Statistical Methodology 75, no. 3 (2013): 577-599.

# Vectorized Tensor-Product Representation

Using the vectorization operator,

$$\tilde{x} = \mathrm{vec}(\tilde{X}),$$

the tensor-product representation becomes

$$\tilde{x} = (B_2 \otimes B_1)\theta,$$

where

$$\theta = \mathrm{vec}(\Theta).$$

The Kronecker-product structure naturally arises from tensor-product basis expansions.

# Why P-Splines?

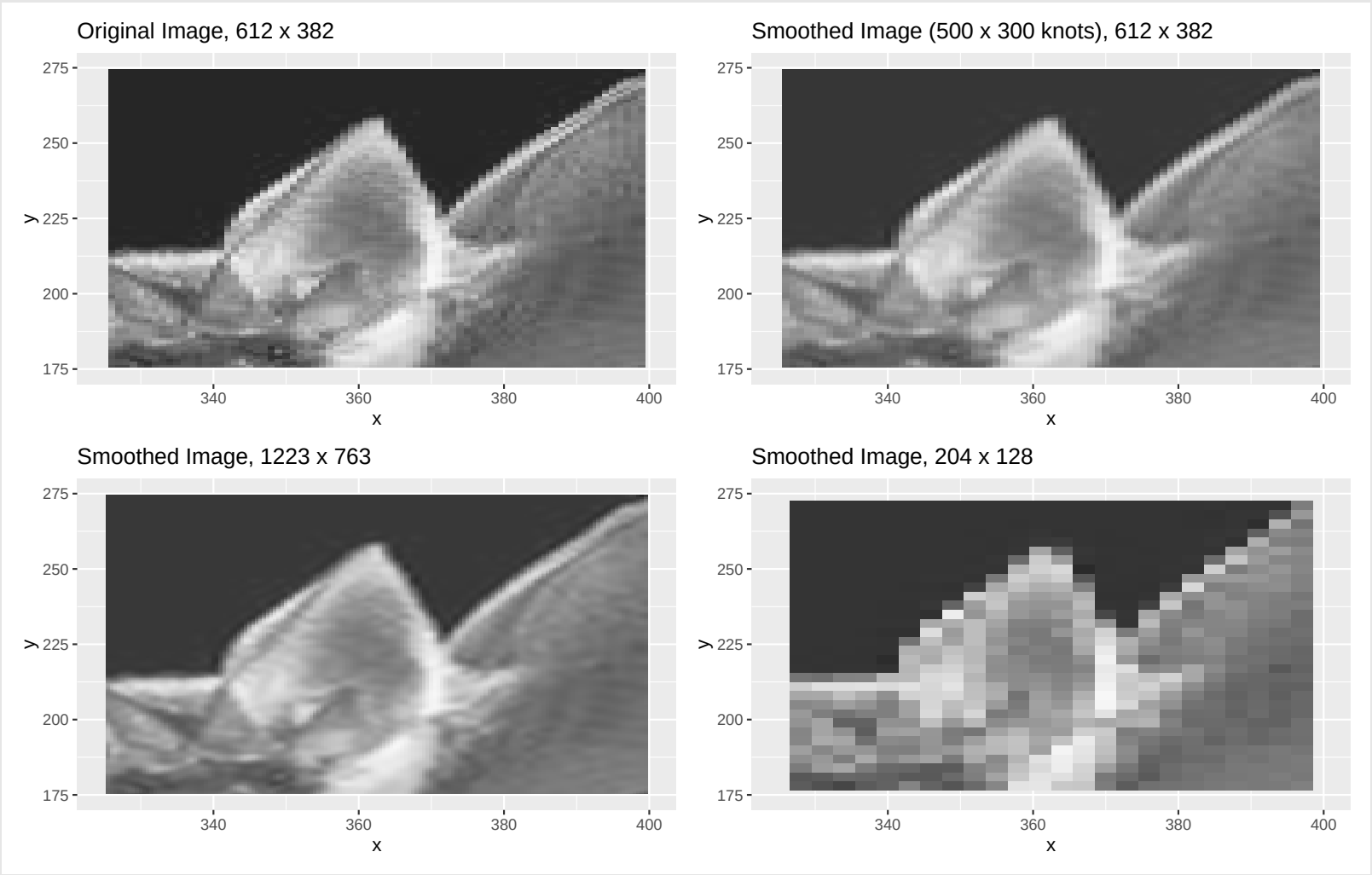
P-splines combine:

- the flexibility of rich B-spline bases,
- with computationally simple penalization.

Smooth functions therefore arise from smooth coefficient sequences.

Finite-difference penalties are also easy to generalize to higher-dimensional tensor-product surfaces.

# Image Smoothing and Resolution



# Functional Representation of the Contour