THESIS

Presented at
Université d'Auvergne

For the degree of
Doctor of the Université d'Auvergne

Speciality
Computer Vision

Defended by
Florent BRUNET

on 2010, November 30

Title

Contributions to Parametric Image Registration and 3D Surface Reconstruction

Jury

President	Laurent Sarry
Co-Directors	Adrien Bartoli
	Rémy Malgouyres
	Nassir Navab
Reviewers	Lourdes Agapito
	Joachim Hornegger
	Étienne Mémin

Abstract

This thesis deals with the modelling and the estimation of parametric functions in Computer Vision. It focuses on three main topics: range surface fitting, image registration, and 3D reconstruction of smooth surfaces. In addition to these three main topics, we consider other transversal elements. In particular, we focus on the general problem caused by the hyperparameters in parametric model estimation, which includes regularization problems. All these topics are related by a single objective which is nowadays one of the most important goals in Computer Vision: the reconstruction of an arbitrary surface from images taken in an arbitrarily deforming environment. This thesis can be divided into four main parts.

The first part deals with the basics. It includes background on optimization and on parameter estimation. The problems related to the hyperparameters are also explained and illustrated. The rest of the thesis is centred on our original contributions.

The second part of this thesis deals with the problem of fitting a surface to range data. This problem consists in finding a parametric smooth surface that approximates accurately a sparse set of 3D points. We consider two main problems. First, we propose methods to automatically tune the hyperparameters such as a regularization weight. Second, we show how heteroskedastic noise may be handled. Heteroskedastic noise is an important problem since it is typical of range sensors, for instance Time-of-Flight cameras.

The third part of this thesis is dedicated to the problem of image registration. We propose three contributions in this topic. First, we present a new warp (image deformation function) able to correctly model the effect of perspective projection. Second, we show how to solve an important problem that typically happens in direct image registration: the problem of the region of interest. Third, we propose a new framework to estimate in a reliable way the hyperparameters needed in feature-based image registration (threshold of an M-estimator, regularization weight, number of control points, *etc*).

The last part of this thesis deals with the problem of reconstructing an inextensible surface from a monocular sequence of images. We also use the hypothesis that a reference shape is known. Using only the motion cue, the problem is ill-posed but, nonetheless, satisfying and plausible results can be obtained. We propose two new formulations to reconstruct the surface: the first one reconstruct a sparse set of points using a second order cone program, and the second one reconstruct a smooth parametric surface using a least-squares minimization problem.

Résumé

Cette thèse traite de la modélisation et de l'estimation de fonctions paramétriques en vison par ordinateur. Notre travail se concentre sur trois axes principaux : l'ajustement de surfaces sur données de profondeur, le recalage d'images et la reconstruction de surfaces tridimensionnelles à partir d'images. En outre, nous abordons des sujets transversaux. En particulier, nous nous intéressons aux problèmes posés par les hyperparamètres (ce qui inclue les problèmes de régularisation). Tous ces aspects convergent vers un seul et unique but ultime : la reconstruction de surface quelconques à partir d'images prises en environnements déformables. Cette thèse peut être divisées en quatre parties.

La première partie traite des éléments fondamentaux comme les conventions et les notations. Les bases de l'optimisation et de l'estimation des problèmes paramétriques sont aussi expliquées. La notion d'hyperparamètre est expliquée et illustrée. Le reste de ce document est axé sur nos contributions originales.

La deuxième partie de cette thèse traite du problème de l'ajustement de surface sur des données de profondeur. Ce problème consiste à déterminer une surface paramétrique qui approxime de manière fidèle un

ensemble discret de points tridimensionnels. Nous étudions deux sous-problèmes. En premier lieu, nous proposons une méthode permettant de choisir automatiquement les hyperparamètres contrôlant l'importance de la régularisation. En second lieu, nous montrons comment un bruit hétéroskédastique peut être géré.

La troisième partie de cette thèse est dédiée au recalage d'images. Nous proposons trois contributions dans ce domaine. Premièrement, nous présentons un nouveau modèle paramétrique permettant de prendre en compte les effets d'une caméra perspective. Deuxièmement, nous montrons comment résoudre un problème important du recalage d'images par approche directe : le problème de la région d'intérêt. Troisièmement, nous proposons un nouveau cadre générique pour l'estimation automatique des hyperparamètres nécessaires au recalage d'images par approche basée primitive (ce qui inclut les seuils de M-estimateurs, le poids de la régularisation ou le nombre de points de contrôles d'un modèle de déformation).

Enfin, la quatrième partie de cette thèse se concentre sur le problème de la reconstruction d'une surface à partir d'une séquence monoculaire d'images. Nous faisons l'hypothèse que la surface à reconstruire est inextensible et qu'une forme de référence est disponible. Si l'on n'utilise que l'information produite par le déplacement de la surface, ce problème est mal posé. Néanmoins, des résultats satisfaisants et plausibles peuvent être atteints. Nous proposons deux nouvelles formulations permettant de reconstruire la surface : la première reconstruit un ensemble discret de points 3D par programmation conique du second ordre (*second order cone programming*) ; la seconde reconstruit une surface paramétrique lisse en utilisant une minimisation de moindres carrés.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital images and videos are nowadays ubiquitous. This stems from the rapid growth of cheap sensors such as webcams, digital cameras, camcorders, smartphones... A natural consequence of this omnipresence is a need for sophisticated algorithms to manipulate a massive amount of data. This is one of the reasons why Computer Vision has become a major research topic over the past few decades.

In a nutshell, the ultimate goal of Computer Vision would be to make computers able to understand the world into which they 'live'. Here, the word 'computer' must be taken in a broad sense since computing chips are now not only in classical computers but in many other devices such as smartphones, cars, *etc*. The verb 'to understand' must also be considered in a really broad sense. Despite all the efforts spent by the scientists during the past decades, we are still far from having intelligent computers. In a more realistic way, it would be probably more reasonable to talk about 'automatic extraction of information' of images instead of 'understanding the world'. The mechanical being does not exist yet but Computer Vision is nonetheless of broad interest with useful applications in domains such as multimedia, metrology, medical imaging, robotics, *etc*. Computer Vision has been present in the professional context for decades now, particularly in the field of medical imaging. However, the situation is evolving quite rapidly since the beginning of this millennium. Indeed, Computer Vision is now involved in mass products such as mobile phones, cars, and game consoles. This has opened brand new perspectives and developments for research in Computer Vision.

The magic triplet

This thesis deals with specific problems in Computer Vision. More precisely, we mainly consider the three following topics: surface fitting, image registration, and 3D reconstruction in deformable environments. In this thesis, we always consider parametric approaches. This is a common characteristic to all our contributions. Before giving general points about the specific problems treated here, let us have a quick look at what a parametric approach is. Most fundamentally, a parametric approach relies on what we call the 'magic triplet'[1]. This magic triplet is made of the three following elements:

Parametric models. A parametric model is a family of functions that can be described by a finite set of parameters[2]. The combination of a parametric model with a set of parameters allows one to model a phenomenon (such as a surface that fits a cloud of points or the deformation between two images). Many parametric models may be used in Computer Vision, ranging from very specific models (representing, for instance, an affine transformation) to models of general use (such as the well-known B-splines that allows one to model complex transformations such as the image deformation function). The choice of a 'good' parametric model, *i.e.* a model that can represent the phenomenon under study, is extremely important. In this thesis, we use many different existing parametric models and also propose new ones.

Parameter estimation. Parameter estimation is a central part of parametric approaches. It consists in finding an appropriate set of parameters that, combined with a fixed parametric model, 'explains' correctly a data set. This is achieved by modelling the problem under study. It usually results in a 'score function' (also known as criterion, cost function, loss function, or residual error). The minimization (or, sometimes, maximization) of this criterion is expected to give the right result. The modelling step is a mathematical formulation of the problem that takes into account various elements such as the nature of the data, the type of measurement noise, the presence of erroneous data, the prior knowledge one has about the solution, *etc*.

[1]This is a term that the reader will probably not found in the literature but that we nonetheless find quite appropriate.
[2]An infinite number of parameters could even be considered but is not treated in this work.

Hyperparameter estimation. Hyperparameter estimation is another important part of any classical parametric approach. What we call *hyperparameters* in this document are the additional parameters that typically arise in the optimization problems resulting of the modelling step. The number of control points of a B-spline or the strength given to a regularization prior are examples of hyperparameters. For reasons that will become clear along this manuscript, hyperparameters cannot be estimated the same way as natural parameters. Determining in an automatic way good hyperparameters is a challenging problem that is often neglected. Using appropriate hyperparameters is nonetheless crucial in order to get satisfactory results with a parametric approach. Part of our work is dedicated to this point.

Computer Vision problems addressed in this thesis

As said previously, the work presented in this thesis focuses on various specific topics in Computer Vision. We now give a brief overview of these topics. Of course, these elements will be further detailed in the corresponding chapters of this thesis.

Range surface fitting is the problem of finding an analytic expression of a smooth parametric surface that approximates a set of 3D data point. In this document, we consider 'range data'. This type of data is also known as 2.5D. It may be viewed as a set of 2D locations, each one of which being associated an altitude (or height, or depth). Range data is now of broad interest because there are some devices that allows one to get such data quite easily: Time-of-Flight cameras, laser range scanners, *etc*. The main challenges encountered in such problems is to cope with noise, large amounts of data, and discontinuities.

Image registration is the problem of determining the transformation between two (or more) images of the same scene. Various types of transformations may be considered: photometric, geometric. In this document, we are mainly interested in geometric transformations. Besides, we focus on deformable environments. It means that the position (or the shape) of the objects may vary between the images to register. This implies that complex parametric models must be used to model the deformations. Consequently the parameter estimation step also becomes quite difficult in general.

3D reconstruction of deformable surfaces. The last problem of Computer Vision addressed in this document is the reconstruction of a deforming 3D surface from a monocular video. Using the motion cue only makes it a fundamentally ill-posed problem since there exist an infinite number of 3D shapes that have the same reprojection in an image. We propose to overcome this problem by considering that the deformable surface is inextensible and that a reference shape is available for a template image. Although these assumptions are common, the way we enforce the underlying constraints is new: we model the reconstructed surface as a smooth surface (based on tensor-product B-splines) and impose that the surface be everywhere a local isometry.

Contributions

Various contributions related to the three main topics of the previous section are given in this thesis. We now give a brief overview of our contributions.

In surface fitting, we propose a new method that allows one to automatically tune the hyperparameters in an efficient way from a computational point of view. The results we get with our approach are similar to those obtained with state-of-the-art approaches such as Cross-Validation. We also propose an algorithm to fit a surface to data presenting heteroskedastic noise, *i.e.* noise with a variance which is not constant over the whole dataset. Moreover, our algorithm is extremely fast, which is interesting to process data coming from devices such as Time-of-Flight cameras.

In image registration, we propose a new parametric warp relying on NURBS (Non Uniform Rational B-Spline). We show that this model is particularly well-suited to perspective imaging conditions while, on the contrary, classical Free-Form Deformations relying on B-splines are more adapted to affine imaging conditions. Practical elements about the estimation of our NURBS-warps' parameters are also given in details.

In direct image registration, we propose a new modelling of the problem that allows one to discard the thorny problems caused by the so-called 'region-of-interest'. This new approach relies on the simple, yet successful, idea that the parts of the scene seen in one image but out of the field of view in the other images may be considered as outliers (as the outliers caused, for instance, by occlusions or specularities).

In feature-based image registration, we provide a new principle for automatically tuning the hyperparameters. This new principle relies on the well-known paradigm of dividing the data into a training set and a test set. We adapt this principle to the specific context of feature-based image registration by considering features as the training set and pixel colours as the test set. This approach has the advantage of using all the available image information (both features and colours). In a sense, this new approach may be considered as a way of combining the feature-based approach (used for the estimation of the natural parameters of the warp) and of the direct approach (used to automatically set the hyperparameters).

In 3D surface reconstruction, we propose a new algorithm that reconstructs inextensible surfaces from a monocular video. Two approaches are proposed. The first one reconstructs a sparse surface, *i.e.* a cloud of three-dimensional points. We manage to implement this first approach as a second order cone program. The second one reconstructs a smooth and parametric surface such as a tensor product B-spline. The founding idea is to say that the function (the parametric model) representing the reconstructed 3D surface must be locally and everywhere an isometry. This idea is implemented as a new term which is included in a cost function that also bundles a data term and a regularization term.

Outline of this thesis

Chapter 2 presents the general tools used in this thesis. In particular, we give all the convention and notation, the basic elements on optimization, and the most usual parametric models of functions. Chapter 3 also deals with basic elements but which are more related to the central topics of this thesis, *i.e.* basics on parameter and hyperparameter estimation. While these first two chapters only address general points, the other chapters are specific to our work. In particular, each one of the last chapters includes at least one of our contributions. Chapter 4 is dedicated to the problem of fitting a parametric smooth surface to range data. General points about range data are given in this chapter. In particular, some explanations on the standard devices that allows one to acquire range data are given. We also present our two contributions which are related to the problem of surface fitting with range data. Chapter 5 is concerned by a fundamental problem in Computer Vision: image registration. A review of the classical approaches to image registration is given. Then, two contributions related to parameter estimation in image registration are given in details. Chapter 6 is also dedicated to image registration. However, we have decided not to merge chapter 5 and chapter 6. Indeed, chapter 5 is about specific techniques that, given a parametric model, allows one to compute the parameters (and the hyperparameters). Chapter 6 is different of chapter 5 in the sense that we propose a new parametric model instead of giving methods to estimates the parameters. Chapter 7 deals with the reconstruction of deformable and inextensible surfaces from a monocular video. This chapter comes last since the tools and methods it proposes may be built upon the contributions brought in the previous chapters. For instance, the first step in the method we propose to reconstruct a 3D surface is to register successive images of the video. We conclude and comment our work and its possible future evolutions in chapter 8.

Chapter 2

General Tools

This chapter gives the basic tools required to read this thesis. We first define the notation. We then deal with the basic principles of optimization followed by a review of the most important optimization tools and algorithms. We finish this chapter by reviewing the parametric function models that we use the most in this thesis.

2.1 Notation, First Definitions

Although most of our notation is compliant with the international standard ISO 31-11 (Thompson and Taylor, 2008), we feel that it is appropriate to give the details of the notation most commonly used in this manuscript. We also give some basic definitions concerning, for instance, some standard operators or matrices.

2.1.1 Basic Notations

Scalars are denoted in italic Roman lowercase letters (*e.g.* x) or, sometimes, italic Greek lowercase (*e.g.* α). Vectors are written using bold fonts (*e.g.* $\mathbf{p}$). They are considered as column vectors. Sans-serif fonts are used for matrices (*e.g.* M). The elements of a vector are denoted using the same letter as the vector but in an italic Roman variant. The same remark holds for matrices. Full details on notation and tools for vector and matrices will be given in section 2.1.4.

2.1.2 Functions

A function is the indication of an input space I, an output space O, and a mapping between the elements of these spaces. A function is said to be monovariate when $\dim(I) = 1$ and multivariate when $\dim(I) > 1$. All the same way, a function is said to be scalar-valued when $\dim(O) = 1$ and vector-valued when $\dim(O) > 1$. We use lowercase italic letters for scalar-valued functions (*e.g.* f) and calligraphic fonts for vector-valued functions (*e.g.* $\mathcal{W}$). From time to time, other typographic conventions are used to denominate functions depending on the context. A function $\mathcal{F}$ is defined using this notation: $\mathcal{F} : I \to O$. The mapping between an element $\mathbf{x} \in I$ and its corresponding image $\mathcal{F}(\mathbf{x}) \in O$ is denoted $\mathbf{x} \mapsto \mathcal{F}(\mathbf{x})$. The vector $\mathbf{x}$ is called the free variable (and it can be replaced by any other notation). The complete definition of a function is written:

$$\begin{aligned} \mathcal{F} : I &\longrightarrow O \\ \mathbf{x} &\longmapsto \mathcal{F}(\mathbf{x}). \end{aligned} \tag{2.1}$$

Differential operators. Let $f : \mathbb{R}^m \to \mathbb{R}$ be a scalar-valued function and let $\mathbf{x} \in \mathbb{R}^m$ be the free variable. The partial derivative of f with respect to x_i is denoted by $\frac{\partial f}{\partial x_i}$. The gradient of f evaluated at $\mathbf{p}$ is denoted $\boldsymbol{\nabla}_{\mathbf{p}} f(\mathbf{p})$. It is considered as a column vector:

$$\boldsymbol{\nabla}_{\mathbf{p}} f(\mathbf{p}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{p}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{p}) \end{pmatrix}. \tag{2.2}$$

In practice, the dependency on $\mathbf{p}$ is omitted when it is obvious from the context. Consequently, $\boldsymbol{\nabla}_{\mathbf{p}} f(\mathbf{p})$ is often shortened to $\boldsymbol{\nabla} f(\mathbf{p})$ or even $\boldsymbol{\nabla} f$.

For a vector-valued function $\mathcal{F} : \mathbb{R}^m \to \mathbb{R}^n$, the counterpart of the gradient is the Jacobian matrix, denoted $\mathsf{J}_{\mathcal{F}}$. If the components of $\mathcal{F}$ are denoted $\{f_i\}_{i=1}^n$ then the Jacobian matrix is defined as:

$$\mathsf{J}_{\mathcal{F}}(\mathbf{p}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{p}) & \cdots & \frac{\partial f_1}{\partial x_m}(\mathbf{p}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{p}) & \cdots & \frac{\partial f_n}{\partial x_m}(\mathbf{p}) \end{pmatrix} \in \mathbb{R}^{n \times m}. \tag{2.3}$$

As for the gradient, the point where the Jacobian matrix is evaluated is omitted when it is clear from the context.

The Hessian matrix of a scalar-valued function $f : \mathbb{R}^m \to \mathbb{R}$ is the matrix of the partial derivatives of second order. This matrix is denoted H_f and it is defined by:

$$\mathsf{H}_f(\mathbf{p}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{p}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{p}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m}(\mathbf{p}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{p}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{p}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_m}(\mathbf{p}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1}(\mathbf{p}) & \frac{\partial^2 f}{\partial x_m \partial x_2}(\mathbf{p}) & \cdots & \frac{\partial^2 f}{\partial x_m^2}(\mathbf{p}) \end{pmatrix} \in \mathbb{R}^{m \times m}. \tag{2.4}$$

As for the gradient and the Jacobian matrix, we consider that the notation $\mathsf{H}_f(\mathbf{p})$ is equivalent to the notation H_f when the vector $\mathbf{p}$ is obvious from the context.

2.1.3 Sets and Collections

The sets are usually written using upper-case letters (*e.g.* S). The usual sets of numbers are denoted using the blackboard font: $\mathbb{N}$ for the natural numbers, $\mathbb{Z}$ for the integers, $\mathbb{Q}$ for the rational numbers, $\mathbb{R}$ for the real numbers, and $\mathbb{C}$ for the complex numbers[1]. The explicit definition of a set is denoted using the curly brackets (*e.g.* $A = \{1, 2, \dots, 10\} = \{i \mid i = 1, \dots, 10\} = \{i\}_{i=1}^{10}$). The vertical bar in a set definition is synonym of the expression 'such that' (often abbreviated 's.t.'). Following the Anglo-Saxon convention, we consider that $\mathbb{N} = \{1, 2, \dots\}$ and $\mathbb{N}^* = \{0, 1, 2, \dots\}$ while $\mathbb{R}$ is the set of all the real numbers and $\mathbb{R}^* = \mathbb{R} \setminus \{0\}$. The set of all the positive (respectively negative) real numbers is denoted $\mathbb{R}_+$ (respectively $\mathbb{R}_-$). The Cartesian product of two sets is designated using the $\times$ symbol, *i.e.* for two sets A and B, we have $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$. The notation A^n represents the Cartesian product of A with itself iterated n times. The symbols used for the intersection, the union, and the difference are respectively $\cap$, $\cup$, and $\setminus$.

Real intervals are denoted using brackets: $[a, b]$ is the set of all the real numbers x such that $a \leq x \leq b$. The scalars a and b are called the endpoints of the interval. We use outwards-pointing brackets to indicate the exclusion of an endpoint: for instance, $]a, b] = \{x \in \mathbb{R} \mid a < x \leq b\}$.

Integer intervals (also known as *discrete intervals*) are denoted using either the double bracket notation or the 'three dots' notation. For instance, the integer interval $\{-1, 0, 1, 2\}$ may be written $[\![-1, 2]\!]$ or $\{-1, \dots, 2\}$.

Collections, *i.e.* grouping of heterogeneous or 'complicated' elements such as point correspondences are denoted using fraktur fonts (*e.g.* $\mathfrak{D}$).

2.1.4 Matrices and Vectors

Matrices are denoted using sans serif font (*e.g.* M). Although a vector is a special matrix, we use bold symbols for them (*e.g.* $\mathbf{p}$ or $\boldsymbol{\beta}$). By default, vectors are considered as column vectors. The set of all the matrices defined over $\mathbb{R}$ and of size $m \times n$ is denoted $\mathbb{R}^{m \times n}$. The transpose, the inverse, and the pseudo-inverse of a matrix A are respectively denoted A^T, A^{-1}, and $\mathsf{A}^\dagger$. The pseudo-inverse is generally defined as $\mathsf{A}^\dagger = \left(\mathsf{A}^\mathsf{T}\mathsf{A}\right)^{-1} \mathsf{A}^\mathsf{T}$ (see section 2.2.2.6). The coefficient located at the intersection of the ith row and the jth column of the matrix A is denoted $a_{i,j}$. The coefficients of a vector are noted using the same letter but with the bold removed. For instance, the ith coefficient of the vector $\boldsymbol{\beta}$ is written β_i.

[1]Historically, these sets were typed using simple bold font but since it was difficult to make bold fonts on black boards, mathematicians created the 'double-bar' letters which were later adopted by typographers.

Parentheses and brackets. We use either the parenthesis or squared brackets when giving the explicit form of a matrix. Parenthesis are used when the elements are scalars, *e.g.* :

$$\mathsf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}. \tag{2.5}$$

The bracket notation is used when the matrix is defined with 'blocks', *i.e.* juxtaposition of matrices, vectors, and scalars. For instance:

$$\mathsf{B} = \begin{bmatrix} \mathsf{A} & 2\mathsf{A} \\ 3\mathsf{A} & 4\mathsf{A} \end{bmatrix}. \tag{2.6}$$

Common matrices. The identity matrix of size $n \times n$ is denoted $\mathbf{I}_n$:

$$\mathbf{I}_n = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}. \tag{2.7}$$

The matrix of size $m \times n$ filled with zeros is denoted $\mathbf{0}_{m \times n}$. The subscripts in the notation $\mathbf{I}_n$ and $\mathbf{0}_{m \times n}$ are often omitted when the size can be easily deduced from the context.

Common operators. The operator vect is used for the column-wise vectorization of a matrix. For instance, if $\mathsf{A} \in \mathbb{R}^{m \times n}$:

$$\mathrm{vect}(\mathsf{A}) = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{bmatrix}, \qquad \text{with } \mathsf{A} = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix}. \tag{2.8}$$

The operator diag deals with diagonal matrices. The effect of this operator is similar to the one of the `diag` function in Matlab. Applied to a vector $\mathbf{d} \in \mathbb{R}^n$, it builds a matrix $\mathsf{D} \in \mathbb{R}^{n \times n}$ such that:

$$\mathsf{D} = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & d_n \end{pmatrix}. \tag{2.9}$$

Conversely, when applied to a square matrix $\mathsf{M} \in \mathbb{R}^{n \times n}$, the operator diag builds a vector that contains the diagonal coefficients of M:

$$\mathrm{diag}(\mathsf{M}) = \begin{pmatrix} m_{1,1} & \dots & m_{n,n} \end{pmatrix}^\mathsf{T}. \tag{2.10}$$

The Hadamard product. The *Hadamard product* of two matrices, also known as the *element-wise product*, is denoted with the $\odot$ symbol. The Hadamard product of the matrices A and B is the matrix $\mathsf{C} = \mathsf{A} \odot \mathsf{B}$ such that $c_{i,j} \overset{\text{def}}{=} a_{i,j} b_{i,j}$. The matrices A, B, and C all have the same size.

The Kronecker product. The *Kronecker product*, denoted with the symbol $\otimes$, is a binary operation on two matrices of arbitrary sizes. Let $\mathsf{A} \in R^{m_a \times n_a}$ and $\mathsf{B} \in R^{m_b \times n_b}$ be two matrices. The Kronecker product of A

and B is defined as follows:

$$A \otimes B = \begin{bmatrix} a_{11}\mathsf{B} & \cdots & a_{1n_a}\mathsf{B} \\ \vdots & & \vdots \\ a_{m_a1}\mathsf{B} & \cdots & a_{m_an_a}\mathsf{B} \end{bmatrix}. \tag{2.11}$$

Vector norms. The *p-norm* of a vector $\mathbf{v} \in \mathbb{R}^n$ is denoted $\|\mathbf{v}\|_p$. It is defined for $p \geq 1$ by:

$$\|\mathbf{v}\|_p \overset{\text{def}}{=} \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}. \tag{2.12}$$

Note that the 1-norm is also known as the *taxicab norm* or the *Manhattan norm*. The 2-norm corresponds to the Euclidean norm. In this case, we prefer the notation $\|\mathbf{v}\|$ instead of the notation $\|\mathbf{v}\|_2$:

$$\|\mathbf{v}\| \overset{\text{def}}{=} \sqrt{\sum_{i=1}^{n} v_i^2}. \tag{2.13}$$

The *maximum norm*, also known as the *infinity norm* or *uniform norm*, is denoted $\|\mathbf{v}\|_\infty$. It is defined as:

$$\|\mathbf{v}\|_\infty \overset{\text{def}}{=} \max_{i \in [\![1,n]\!]} |v_i|. \tag{2.14}$$

Note that the maximum norm corresponds to the p-norm when $p \to \infty$.

The Frobenius norm. The Frobenius norm of a matrix $\mathsf{A} \in \mathbb{R}^{m \times n}$ is denoted $\|\mathsf{A}\|_{\mathcal{F}}$. It is defined as:

$$\|\mathsf{A}\|_{\mathcal{F}} \overset{\text{def}}{=} \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j}^2}. \tag{2.15}$$

The Frobenius norm of a matrix is related to the Euclidean norm of a vector in the sense that they are both defined as the square root of the sum of the squared coefficients. In fact, we have the following equality:

$$\|\mathsf{A}\|_{\mathcal{F}} = \|\text{vect}(\mathsf{A})\|. \tag{2.16}$$

2.1.5 Other Common Notation

Logic. The symbol $\forall$ means 'for all' and the symbol $\exists$ means 'there exists'.

Limits. The symbol $\to$ is synonym of 'tends to' (*e.g.* $x \to \infty$ means that x tends to the infinite).

Others. Some other notation will surely come during the next few weeks.

2.2 Basics on Continuous Optimization

Most of the problems solved in this thesis can be formulated as optimization problems. In this section, we give some general points on optimization. We also give details on the most used optimization algorithms in this document. The statistical grounds of certain type of cost function will be explained in section 3.1.

2.2.1 Generalities on Optimization

Unconstrained optimization. In its general form, an unconstrained minimization problem has the following form (Björck, 1996; Boyd and Vandenberghe, 2004; Culioli, 1994; Nocedal and Wright, 1999; Press et al., 1992):

$$\min_{\mathbf{x}} f(\mathbf{x}). \tag{2.17}$$

The function f is a scalar-valued function named the *cost function* or the *criterion*. It is assumed that the cost function is defined on $\mathbb{R}^n$. In some cases later explained, f can be a vector-valued function instead of a scalar-valued one. Note that we only consider the case of the minimization of the cost function since the problem of maximization can easily be turned into a minimization problem (by taking the negative of the cost function).

Constrained optimization. A constrained minimization problem is similar to an unconstrained minimization problem except that supplemental conditions on the solution must be satisfied. Such a problem has the following form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in C, \end{aligned} \tag{2.18}$$

where C is a set defining the constraints. These constraints can take many forms. For instance, it can be inequality constraints such as $x_i \geq 0$, linear equality constraints such as $\mathsf{A}\mathbf{x} = \mathbf{b}$ for some matrix A and some vector $\mathbf{b}$. It can also be more complex constraints such as $g(\mathbf{x}) \leq 0$ where g is some function. These are just a few examples.

In this section we focus on unconstrained minimization problem since it is the main tool used in this thesis.

Optimization algorithms. An *optimization algorithm* is an algorithm that provides a solution to an optimization problem. There exists a vast amount of optimization algorithms. The algorithm used to solve an optimization problem depends on the properties of the cost function and of the constraints. For instance, the optimization algorithm depends on the differentiability of the cost function. It can also exploit a particular form of the cost function (for instance, a cost function defined as a sum of squares).

Global optimization. The ultimate goal of a minimization problem is to find a *global minimum* of the cost function, *i.e.* a vector $\mathbf{x}^\star$ such that:

$$\forall \mathbf{x} \in \Omega, f(\mathbf{x}^\star) \leq f(\mathbf{x}). \tag{2.19}$$

The solutions proposed by optimization algorithms are only guaranteed to be local minima. A vector $\mathbf{x}$ is said to be a local minimizer of the function f if the following statement is true:

$$\exists r \in \mathbb{R}_+^\star : \forall \mathbf{y} : \|\mathbf{x} - \mathbf{y}\| \leq r : f(\mathbf{x}) \leq f(\mathbf{y}). \tag{2.20}$$

When the function f is differentiable, a necessary condition for $\mathbf{x}$ to be a local minimizer is:

$$\boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{0}. \tag{2.21}$$

A local minimum of the function f is not necessarily a global minimum of the function. Indeed, a cost function may have several local minima. Deciding whether a local minima is a global minimum or not is an undecidable problem.

Name	Abb.	Cost function	Sec
Downhill simplex	$\times$	Arbitrary	2.2.2.2
Gradient descent	GD	Once differentiable	2.2.2.3
Newton	$\times$	Twice differentiable	2.2.2.4
Gauss-Newton	GN	Sum of squared functions $\rightarrow$ non-linear least-squares (NLS)	2.2.2.5
Levenberg-Marquardt	LM	NLS	2.2.2.7
Cholesky factorization	$\times$	Linear system of equations $\mathsf{A}\mathbf{x} = \mathbf{b}$ (with A positive definite) and, coincidently, sum of squared linear functions $\rightarrow$ linear least-squares (LLS)	2.2.2.8
QR factorization	$\times$	LLS	2.2.2.9
Singular Value Decomposition	SVD	LLS	2.2.2.10
Iteratively Reweighted Least-Squares	IRLS	Weighted sum of squared functions with variable weights	2.2.2.11
Golden section search	$\times$	Mono-variate, scalar-valued, uni-modal	2.2.2.12

Table 2.1: Optimization algorithms reviewed in section 2.2.2. They are ordered from the more generic to the less generic. 'Abb.' in the second column stands for 'Abbreviation'.

Convex functions and global optimization. Convexity is a particularly interesting property for a function. Indeed, if a cost function is convex then it has only one minimum which is therefore a global minimum of the function. Note that the converse is not true; there exist non-convex functions which have only one (global) minimum.

Convex function. Let $f : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$ be a scalar-valued function. Let $\mathbf{x} \in \Omega$ and $\mathbf{y} \in \Omega$. A scalar function f is said to be convex if the following condition holds for all $\alpha \in [0, 1]$:

$$f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y}). \tag{2.22}$$

Alternatively, if the function f is twice differentiable, then f is convex if the Hessian matrix $\mathbf{H}_f$ is positive definite.

A *strictly convex* function is a function that satisfies equation (2.22) with the $\leq$ sign replaced by $<$. Note that the Hessian matrix of a twice differentiable strictly convex function is not necessarily strictly positive definite: it might only be positive definite.

2.2.2 Optimization Algorithms

In this section, we present the unconstrained minimization algorithms mostly used in this thesis. The choice of an optimization algorithm depends on the properties of the cost function to be minimized. These algorithms can be classified according to several criterion. Table 2.1 summarizes the optimization algorithm reviewed in this section and the properties that must be satisfied by the cost function.

This section is organized as follows: we go from the less restrictive algorithms (in terms of conditions on the cost function) to most restrictive ones (*i.e.* the more specific). Two sections not dedicated to a specific optimization algorithm are intercalated in this section: section 2.2.2.1 gives some general points on iterative optimization algorithms, and section 2.2.2.6 talks about the normal equations.

2.2.2.1 Iterative Optimization Algorithms

Before talking about the details, let us start with some generalities on the optimization algorithms. Most of these algorithms are iterative algorithms. This means that they start from an initial value $\mathbf{x}^{(0)}$ which is then iteratively updated. Therefore, an iterative optimization algorithm builds a sequence $\{\mathbf{x}^{(i)}\}_{i=1}^{i\star}$ that may converge towards a local minimum of the cost function, *i.e.* $\mathbf{x}^{(i^\star)}$ may be close to a local minimum of the solution. With a cost function that has several minima, the initial value $\mathbf{x}^{(0)}$ determines which minimum is found. This initial value is thus extremely important.

An important aspect of iterative optimization algorithm is the stopping criterion. For the algorithm to be valid, the stopping criterion must be a condition that will be satisfied in a finite and reasonable amount of time. A stopping criterion is usually the combination of several conditions. For instance, one can decide to stop the algorithm when the change in the solution becomes very small, *i.e.* when $\|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\| < \varepsilon$ with ε a small fixed constant (for instance, $\varepsilon < 10^{-6}$). Of course, this approach does not account for the scale of the solution. One may replace this condition using, for instance, a relative change, *i.e.* $\|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\|/\|\mathbf{x}^{(i)}\| < \varepsilon$. Another standard stopping criterion is the change in the cost function value: $|f(\mathbf{x}^{(i+1)}) - f(\mathbf{x}^{(i)})| \leq \varepsilon$. A maximal number of iterations must also be determined to guarantee that the optimization algorithm will finish in a finite amount of time. Indeed, the previous two stopping criterion may never be satisfied. In the rest of this section, we will denote STOP-CRIT the stopping criterion. In addition to what was just presented, the content of STOP-CRIT will be further detailed depending on the optimization algorithm.

2.2.2.2 Downhill Simplex

The downhill simplex method is an optimization algorithm due to (Nelder and Mead, 1965). It is a heuristic that does not make any assumption on the cost function to minimize. In particular, the cost function must not satisfy any condition of differentiability. It relies on the use of simplices, *i.e.* polytopes of dimension $n + 1$. For instance, in two dimensions, a simplex is a polytope with 3 vertices, most commonly known as a triangle. In three dimensions, a simplex is tetrahedron.

We now explain the mechanisms of the downhill simplex method of (Nelder and Mead, 1965). Note that more sophisticated versions of this method have been proposed. This is the case, for instance, of the algorithm used by the fminsearch function of Matlab.

The downhill simplex method starts from an initial simplex. Each step of the method consists in an update of the current simplex. These updates are carried out using four operations: *reflection*, *expansion*, *contraction*, and *multiple contraction*. Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function to minimize and let $\{\mathbf{x}_0, \dots, \mathbf{x}_n\}$ be the current simplex ($\mathbf{x}_i \in \mathbb{R}^n$ for all $i \in [\![0, n]\!]$). Let $h \in [\![0, n]\!]$ be the index of the 'worst vertex', *i.e.* the value $h = \arg\max_i f(\mathbf{x}_i)$ and let $l \in [\![0, n]\!]$ be the index of the 'best vertex', *i.e.* the value $l = \arg\min_i f(\mathbf{x}_i)$. The downhill simplex method and the four fundamental operations are detailed in algorithm 1. Figure 2.1 illustrates the four fundamental operations on a 3-simplex. In figure 2.3, as an example, the downhill simplex method is used to optimize the Rosenbrock function (see below).

Stopping criterion. The stopping criterion used by (Nelder and Mead, 1965) is defined by:

$$\sqrt{\frac{1}{n+1} \sum_{i=0}^{n} \left(f(\mathbf{x}_i) - \overline{f(\mathbf{x}_i)} \right)^2} \leq \varepsilon, \tag{2.23}$$

with $\overline{f(\mathbf{x}_i)}$ the average of the values $\{f(\mathbf{x}_i)\}_{i=0}^{n}$, and ε a predefined constant. This criterion has the advantage of linking the size of the simplex with an approximation of the local curvature of the cost function. A very

Algorithm 1: Downhill simplex of (Nelder and Mead, 1965).

```
input : the cost function $f : \mathbb{R}^n \to \mathbb{R}$
        $\{\mathbf{x}_i\}_{i=0}^n$ an initial simplex
output: $\mathbf{x}^\star$, a local minimum of the cost function $f$.
begin
    $k \leftarrow 0$
    while STOP-CRIT and $(k < k_{max})$ do
        $h \leftarrow \arg\max_i f(\mathbf{x}_i)$
        $l \leftarrow \arg\min_i f(\mathbf{x}_i)$
        $\mathbf{x}' \leftarrow (1+\alpha)\bar{\mathbf{x}} - \alpha\mathbf{x}_h$
            where $\alpha > 0$ is the reflection coefficient
        if $f(\mathbf{x}') < f(\mathbf{x}_l)$ then
            $\mathbf{x}'' \leftarrow (1+\gamma)\mathbf{x}' - \gamma\bar{\mathbf{x}}$
                where $\gamma > 1$ is the expansion coefficient
            if $f(\mathbf{x}'') < f(\mathbf{x}_l)$ then
                $\mathbf{x}_h \leftarrow \mathbf{x}''$                                /* expansion */
            else
                $\mathbf{x}_h \leftarrow \mathbf{x}'$                                 /* reflection */
        else if $f(\mathbf{x}') > f(\mathbf{x}_i), \forall i \neq h$ then
            if $f(\mathbf{x}') \leq f(\mathbf{x}_h)$ then
                $\mathbf{x}_h \leftarrow \mathbf{x}'$                                 /* reflection */
            $\mathbf{x}'' \leftarrow \beta\mathbf{x}_h + (1-\beta)\bar{\mathbf{x}}$
                where $0 < \beta < 1$ is the contraction coefficient
            if $f(\mathbf{x}'') > f(\mathbf{x}_h)$ then
                $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i+\mathbf{x}_l}{2} \quad \forall i \in [\![0, n]\!]$     /* multiple contraction */
            else
                $\mathbf{x}_h \leftarrow \mathbf{x}''$                                /* contraction */
        else
            $\mathbf{x}_h \leftarrow \mathbf{x}'$                                     /* reflexion */
        $k \leftarrow k + 1$
    return $\mathbf{x}_l$
```

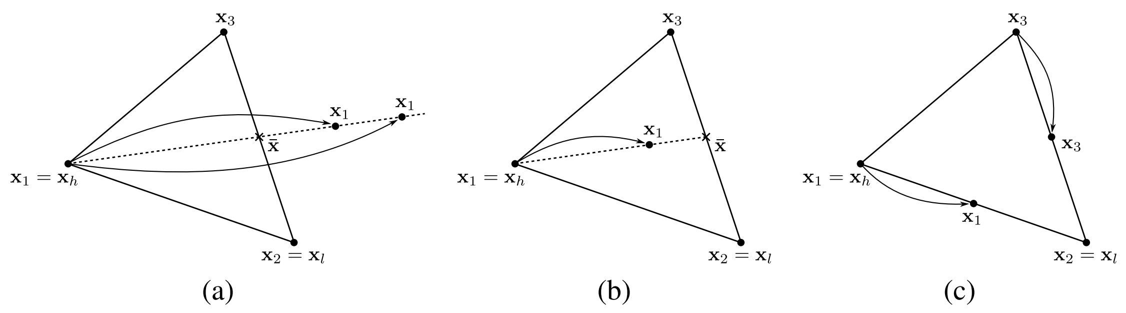


Figure 2.1: Illustration in 2 dimensions of the four fundamental operations applied to the current simplex by the downhill simplex method of (Nelder and Mead, 1965): (a) reflection and expansion, (b) contraction, and (c) multiple contraction.

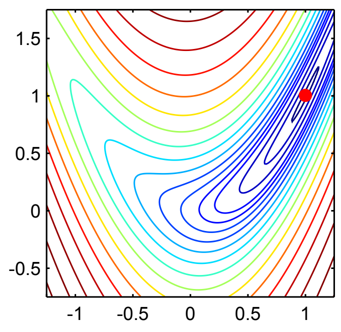


Figure 2.2: Contour plot of the Rosenbrock banana function.

accurate minimum is often obtained for a high curvature of the cost function. On the contrary, a minimum located in a flat valley of the cost function carries less information. Therefore, it does not make much sense to shrink an optimal simplex which would be almost flat.

Rosenbrock function. When it is possible, the algorithms presented in this section are illustrated on the Rosenbrock function. This function, also known as the *banana function*, has been a standard test case for optimization algorithms. It is a two-dimensional function defined as:

$$f(\mathbf{x}) = (1 - x_1)^2 + \alpha(x_2 - x_1^2)^2, \tag{2.24}$$

with α a positive constant. In our illustrations, we use $\alpha = 10$. The minimum is inside a long, narrow, parabolic flat valley. Finding the valley is simple but finding the actual minimum of the function is less trivial. The minimum of the Rosenbrock function is located at the point of coordinate $\mathbf{x}^\star = (1, 1)$, whatever the value given to α. Figure 2.2 gives an illustration of this function.

2.2.2.3 Gradient descent

The method of gradient descent (also known as the method of steepest descent or Cauchy's method) is one the most simple algorithm for continuous optimization (Culioli, 1994). It was first designed to solve linear system a long time ago (Cauchy, 1847). It is an iterative algorithm that relies on the fact that the value of the cost function decreases fastest in the direction of the negative gradient (for a differentiable cost function). The principle of the gradient descent algorithm is given in algorithm 2.

The exact line search procedure of line 5 in algorithm 2 can be replaced with an approximate line search procedure such that the value $\alpha^{(k)}$ satisfies the Wolfe conditions (Michot et al., 2009; Nocedal and Wright, 1999) (see below).

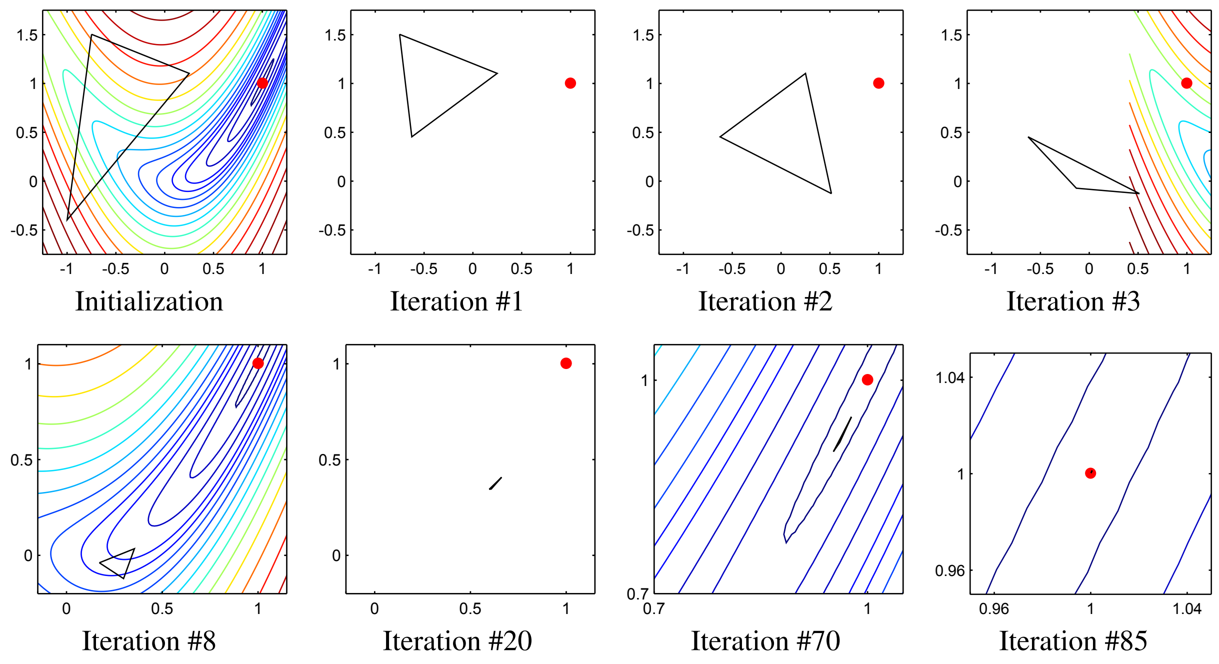


Figure 2.3: Illustration of the downhill simplex method of (Nelder and Mead, 1965) with the Rosenbrock function. The minimum of this function is at the point of coordinate (1,1), represented by the red point on the figures. Here, the convergence is reached in 85 iterations.

Algorithm 2: Gradient Descent

```
  input : $f : \mathbb{R}^n \rightarrow \mathbb{R}$ a differentiable function
          $\mathbf{x}^{(0)}$ an initial solution
  output: $\mathbf{x}^\star$, a local minimum of the cost function $f$.
1 begin
2     $k \leftarrow 0$
3     while STOP-CRIT and $(k < k_{max})$ do
4         $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \alpha^{(k)} \boldsymbol{\nabla} f(\mathbf{x})$
5         with $\alpha^{(k)} = \arg\min_{\alpha \in \mathbb{R}_+} f\left(\mathbf{x}^{(k)} - \alpha \boldsymbol{\nabla} f(\mathbf{x})\right)$
6         $k \leftarrow k + 1$
7     return $\mathbf{x}^{(k)}$
```

Properties. As a first remark, we can notice that the successive search directions used by the gradient descent method are always orthogonal to the level sets of the cost functions. Second, we may also notice that if an exact line search procedure is used, then the method of gradient descent is guaranteed to converge to a local minimum. Under some mild assumptions, it has also been proven to converge with an inexact line search procedure (see, for instance, (Burachik et al., 1996)). However, this convergence is usually very slow. This stems from the fact that, if an exact line search procedure is used, two successive directions are necessarily orthogonal. This is easy to see. Indeed, if $\alpha^{(k)}$ minimizes $f(\mathbf{x}^{(k)} - \alpha^{(k)}\boldsymbol{\nabla} f(\mathbf{x}^{(k)}))$ then we have:

$$\frac{\partial}{\partial \alpha} f(\mathbf{x}^{(k)} - \alpha^{(k)}\boldsymbol{\nabla} f(\mathbf{x}^{(k)})) = 0 \tag{2.25}$$

$$\Leftrightarrow \quad \boldsymbol{\nabla} f(\mathbf{x}^{(k)} - \alpha^{(k)}\boldsymbol{\nabla} f(\mathbf{x}^{(k)}))^\mathsf{T} \boldsymbol{\nabla} f(\mathbf{x}^{(k)}) = 0 \tag{2.26}$$

$$\Leftrightarrow \quad \boldsymbol{\nabla} f(\mathbf{x}^{(k+1)})^\mathsf{T} \boldsymbol{\nabla} f(\mathbf{x}^{(k)}) = 0, \tag{2.27}$$

which proves that the search directions at iteration (k) and $(k+1)$ are orthogonal. Therefore, the path formed by the values $\{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^\star\}$ is shaped like a stairs. Figure 2.4 gives an illustration on the Rosenbrock function of this typical behaviour of the gradient descent method.

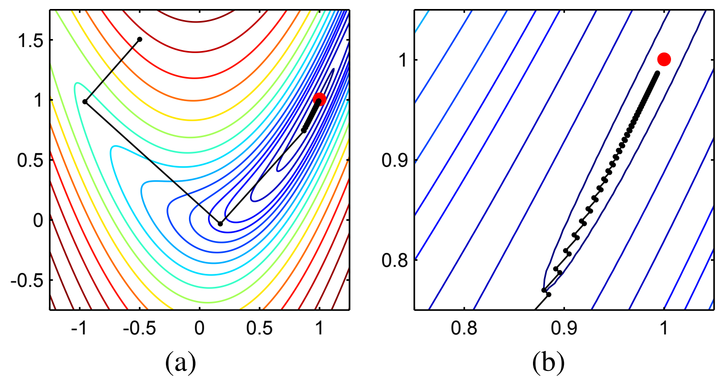


Figure 2.4: Illustration of the successive steps taken by the method of the gradient descent for optimizing the Rosenbrock functions. Notice that the successive search directions are orthogonal. In (b), we see the typical stair effect of this method. It makes the convergence very slow. Indeed, after 100 iterations, convergence has not yet been reached.

Approximate line search and Wolfe conditions. Given a search direction $\boldsymbol{\delta}$, an exact line search procedure amounts to solve the following mono-dimensional minimization problem:

$$\min_\alpha f(\mathbf{x} + \alpha\boldsymbol{\delta}). \tag{2.28}$$

Problem (2.28) can be solved using an algorithm such as the golden section search that will be presented in section 2.2.2.12. Depending on the function f, problem (2.28) can be heavy to compute. Another approach consists in doing an approximate line search. It consists in starting from an initial estimate of α and then refine it so that the Wolfe conditions are satisfied. The Wolfe conditions gives some criteria to decide whether a step α is satisfactory or not. In its basic form, there are two Wolfe conditions:

1. $f(\mathbf{x} + \alpha\boldsymbol{\delta}) \leq f(\mathbf{x}) + \omega_1 \alpha \boldsymbol{\delta}^\mathsf{T} \boldsymbol{\nabla} f(\mathbf{x})$

2. $\boldsymbol{\delta}^\mathsf{T} \boldsymbol{\nabla} f(\mathbf{x} + \alpha\boldsymbol{\delta}) \geq \omega_2 \boldsymbol{\delta}^\mathsf{T} \boldsymbol{\nabla} f(\mathbf{x})$

with ω_1 and ω_2 two constants such that $0 < \omega_1 < \omega_2 < 1$. ω_1 is usually chosen as a small constant and ω_2 as a constant (much) bigger than ω_1. The first condition (also known as the Armijo condition) avoids long steps that would not decrease much the cost function. The second condition (also known as the curvature condition) forces the step to minimize the curvature of the cost function.

2.2.2.4 Newton's Method

Newton's method is another iterative optimization algorithm that minimizes a twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. It relies on the second order Taylor expansion of the function f around the point $\mathbf{x}$:

$$f(\mathbf{x}+\boldsymbol{\delta}) = f(\mathbf{x}) + \boldsymbol{\nabla} f(\mathbf{x})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\mathsf{T}\mathsf{H}_f(\mathbf{x})\boldsymbol{\delta} + O(\|\boldsymbol{\delta}\|^2). \tag{2.29}$$

Each step of the Newton's method consists in finding the minimum of the quadratic approximation of the function f around the current point $\mathbf{x}$. This principle can be stated using the second order Taylor expansion of f:

$$\min_{\boldsymbol{\delta}} f(\mathbf{x}) + \boldsymbol{\nabla} f(\mathbf{x})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\mathsf{T}\mathsf{H}_f(\mathbf{x})\boldsymbol{\delta}. \tag{2.30}$$

A necessary condition for problem (2.30) is obtained by setting to zero the derivative (with respect to $\boldsymbol{\delta}$) of the cost function in equation (2.30). This amounts to solve the following linear system of equations:

$$\mathsf{H}_f(\mathbf{x})\boldsymbol{\delta} = -\boldsymbol{\nabla} f(\mathbf{x}). \tag{2.31}$$

The search direction $\boldsymbol{\delta}$ for the Newton method is thus given by:

$$\boldsymbol{\delta} = -\big(\mathsf{H}_f(\mathbf{x})\big)^{-1}\boldsymbol{\nabla} f(\mathbf{x}). \tag{2.32}$$

Note that, in practice, the inverse Hessian matrix in equation (2.32) does not need to be explicitly calculated ; it is possible to compute $\boldsymbol{\delta}$ from equation (2.31) using an efficient solver of linear systems.

The complete Newton's method is summarized in algorithm 3.

Algorithm 3: Newton's Method

```
input : $f : \mathbb{R}^n \to \mathbb{R}$ a twice-differentiable function
        $\mathbf{x}^{(0)}$ an initial solution
output: $\mathbf{x}^\star$, a local minimum of the cost function $f$.
1 begin
2     $k \leftarrow 0$
3     while STOP-CRIT and ($k < k_{max}$) do
4         $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \boldsymbol{\delta}^{(k)}$
5         with $\boldsymbol{\delta}^{(k)} = -\big(\mathsf{H}_f(\mathbf{x}^{(k)})\big)^{-1}\boldsymbol{\nabla} f(\mathbf{x}^{(k)})$
6         $k \leftarrow k+1$
7     return $\mathbf{x}^{(k)}$
```

Algorithm 3 is the *constant step* version of Newton's algorithm. The update $\mathbf{x} = \mathbf{x} + \boldsymbol{\delta}$ is often replaced by the update $\mathbf{x} = \mathbf{x} + \gamma\boldsymbol{\delta}$ where γ is a positive value smaller than 1. This is done to insure that the Wolfe conditions are satisfied for each step of the algorithm.

Convergence. Every local minimum of the function f has a neighbourhood N in which the convergence of Newton's method with constant step is quadratic (Culioli, 1994). In other words, if we initialize the Newton's method with $\mathbf{x}^{(0)} \in N$, the convergence is quadratic. Outside of the neighbourhoods of the local minima of the cost function, there are no guarantee that Newton's method will converge. Indeed the condition of equation (2.31) is just a necessary condition for having a local minimum: it is *not* a sufficient condition. The condition of equation (2.31) is also satisfied for a local maximum or a saddle point of the cost function. Therefore, Newton's method can converge towards such points.

Illustration. An illustration of Newton's method on the Rosenbrock function is given in figure 2.5. Two variants are presented in this figure: the constant step length variant and the optimal step length determined with a line search strategy.

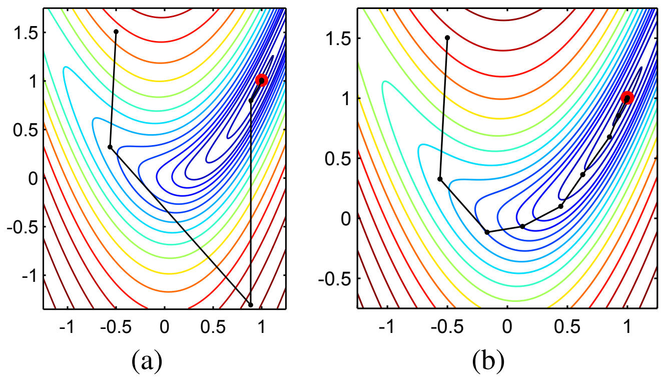


Figure 2.5: Illustration of the successive steps taken by Newton's method for optimizing the Rosenbrock functions. (a) Constant step length variant. (b) Step length determined with an optimal line search strategy. On the Rosenbrock function, the Newton method is usually faster to converge than the gradient descent strategy. In this particular example, convergence is reached within 6 iterations for the constant step length variant and 10 iterations with the optimal line search strategy.

Quasi-Newton algorithms. Newton's method is great since it usually converges quickly. This means that it does not take a lot of iterations to reach the minimum. However, each iteration of this algorithm can be costly since it requires one to compute the Hessian matrix. The goal of quasi-Newton approach is to replace the Hessian matrix by some good approximation, less heavy to compute. There exists several formula to make these approximations iteratively: DFP, SR1, BFGS (Culioli, 1994). We will not detail these formula here but we will give the general principle. As for Newton's method, quasi-Newton relies on the second order Taylor expansion of the function to optimize, except that the Hessian matrix is replaced with an approximation A:

$$f(\mathbf{x}+\boldsymbol{\delta}) \approx f(\mathbf{x}) + \boldsymbol{\nabla} f(\mathbf{x})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\mathsf{T}\mathsf{A}\boldsymbol{\delta}. \tag{2.33}$$

The gradient of this approximation with respect to $\boldsymbol{\delta}$ is given by:

$$\boldsymbol{\nabla} f(\mathbf{x}+\boldsymbol{\delta}) \approx \boldsymbol{\nabla} f(\mathbf{x}) + \mathsf{A}\boldsymbol{\delta}. \tag{2.34}$$

The general principle of a quasi-Newton approach is to choose the matrix A such that:

$$\boldsymbol{\nabla} f(\mathbf{x}+\boldsymbol{\delta}) = \boldsymbol{\nabla} f(\mathbf{x}) + \mathsf{A}\boldsymbol{\delta}. \tag{2.35}$$

The difference between the different formula are the properties that the matrix A satisfies at each iteration of the algorithm. For instance, the BFGS method guarantees that the matrix A will always be symmetric and positive definite. In this case, the approximation of the cost function is convex and, therefore, the search direction is guaranteed to be a descent direction.

2.2.2.5 Gauss-Newton Algorithm

The Gauss-Newton method is an optimization algorithm used to minimize a cost function that can be written as a sum of squares, *i.e.* if $f : \mathbb{R}^n \to \mathbb{R}$:

$$f(\mathbf{x}) = \sum_{i=1}^{m} (f_i(\mathbf{x}))^2 , \tag{2.36}$$

where each f_i for $i \in [\![1, m]\!]$ is a function of the form $f_i : \mathbb{R} \to \mathbb{R}$. The only hypothesis that must be satisfied for the Gauss-Newton algorithm is that the functions f_i must all be differentiable. Equation (2.36) is the very definition of a least-squares problem. For reasons that will become clear in the next section, least-squares cost functions often arises when estimating the parameters of a parametric model. The derivation of the Gauss-Newton algorithm is more conveniently done if we consider the minimization of a vector-valued function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$:

$$\min_{\mathbf{x}} \|\mathcal{F}(\mathbf{x})\|^2, \tag{2.37}$$

where $\mathcal{F}$ is defined as:

$$\mathcal{F}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix}. \tag{2.38}$$

Note that solving problem (2.37) is completely equivalent to minimizing the function f as defined in equation (2.36). The Gauss-Newton algorithm is an iterative algorithm where each step consists in minimizing the first-order approximation of the function $\mathcal{F}$ around the current solution. The first-order approximation of $\mathcal{F}$ is given by:

$$\mathcal{F}(\mathbf{x} + \boldsymbol{\delta}) = \mathcal{F}(\mathbf{x}) + \mathsf{J}_{\mathcal{F}}(\mathbf{x})\boldsymbol{\delta}. \tag{2.39}$$

Therefore, each step of the Gauss-Newton algorithm consists in determining the step $\boldsymbol{\delta}$ by solving the following minimization problem:

$$\min_{\boldsymbol{\delta}} \|\mathcal{F}(\mathbf{x}) + \mathsf{J}_{\mathcal{F}}(\mathbf{x})\boldsymbol{\delta}\|^2. \tag{2.40}$$

Problem (2.40) is a linear least-squares minimization problem. As it will be seen later, such a problem can easily be solved. Algorithm 4 gives the complete principle of the Gauss-Newton method. As for the others algorithms presented so far, a line-search procedure can be combined to the step.

Algorithm 4: Method of Gauss-Newton

```
input  : $f : \mathbb{R}^n \to \mathbb{R}$ a function such that $f(\mathbf{x}) = \sum_{i=1}^m (f_i(\mathbf{x}))^2$ where all the $f_i$ are differentiable
         functions from $\mathbb{R}^n$ to $\mathbb{R}$
         $\mathbf{x}^{(0)}$ an initial solution
output: $\mathbf{x}^\star$, a local minimum of the cost function $f$.
1 begin
2     $k \leftarrow 0$
3     while STOP-CRIT and $(k < k_{max})$ do
4         $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \boldsymbol{\delta}^{(k)}$
5         with $\boldsymbol{\delta}^{(k)} = \arg\min_{\boldsymbol{\delta}} \|\mathcal{F}(\mathbf{x}^{(k)}) + \mathsf{J}_{\mathcal{F}}(\mathbf{x}^{(k)})\boldsymbol{\delta}\|^2$
6         $k \leftarrow k + 1$
7     return $\mathbf{x}^{(k)}$
```

Properties. It can be shown that the step δ is a descent direction (Björck, 1996). If the algorithm converges then the limit is a stationary point of the function f (but not necessarily a minimum). However, with no assumptions on the initial solution, there is no guarantee that the algorithm will converge, even locally.

With a good starting point and a 'nice' function f (*i.e.* a mildly nonlinear function), the convergence speed of the Gauss-Newton method is almost quadratic. However, it can be worse than quadratic if the starting point is far from the minimum or if the matrix $\mathbf{J}_{\mathcal{F}}^{\mathsf{T}}\mathbf{J}_{\mathcal{F}}$ is ill-conditioned.

Illustration. The Rosenbrock function can be written as a sum of squares if we define the functions f_1 and f_2 in the following way:

$$f_1(\mathbf{x}) = 1 - x_1 \tag{2.41}$$

$$f_2(\mathbf{x}) = \sqrt{\alpha}(x_2 - x_1^2) \tag{2.42}$$

Figure 2.6 shows the step taken by the Gauss-Newton algorithm for minimizing the Rosenbrock function.

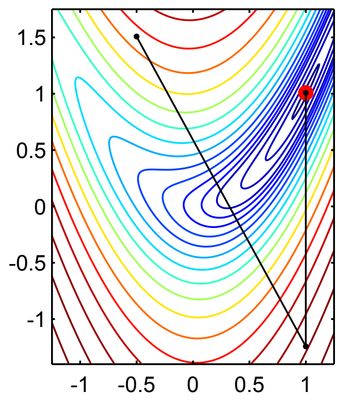


Figure 2.6: Illustration of the Gauss-Newton method on the Rosenbrock function. For this simple example, the optimal solution is found in 2 iterations.

2.2.2.6 The Normal Equations

We broke a little bit our classification of the optimization algorithms and present now a basic tool for linear least-squares problems: the normal equations. We do so because, as it was just seen, the iterations of the Gauss-Newton algorithm involves the solution of a linear least-squares minimization problem. Besides, a part of the algorithm we present next, namely the Levenberg-Marquardt algorithm, uses the so-called normal equations.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function to minimize. Let us suppose that f can be written as a sum of squares of the form $f(\mathbf{x}) = \sum_{i=1}^{m}(f_i(\mathbf{x}) - y_i)^2$, where the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are linear with respect to $\mathbf{x}$ (*i.e.* $\exists \mathbf{f}_i \in \mathbb{R}^n$ such that $f_i(\mathbf{x}) = \mathbf{f}_i^{\mathsf{T}}\mathbf{x}$, for all $i \in [\![1, m]\!]$) and where $y_i \in \mathbb{R}$ for all $i \in [\![1, m]\!]$. Minimizing the function f can be stated in matrix form:

$$\min_{\mathbf{x}} \|\mathsf{F}\mathbf{x} - \mathbf{y}\|^2, \tag{2.43}$$

where the matrix $\mathsf{F} \in \mathbb{R}^{m\times n}$ and the vector $\mathbf{y} \in \mathbb{R}^m$ are defined by:

$$\mathsf{F}^{\mathsf{T}} = \begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_m \end{bmatrix}, \qquad \mathbf{y}^{\mathsf{T}} = \begin{pmatrix} y_1 & \cdots & y_m \end{pmatrix}. \tag{2.44}$$

If the matrix F has full column rank (Björck, 1996), problem (2.43) can be solved using the normal equations:

$$\mathsf{F}^{\mathsf{T}}\mathsf{F}\mathbf{x} = \mathsf{F}^{\mathsf{T}}\mathbf{y}. \tag{2.45}$$

The hypothesis that F has full column rank allows us to say that the square matrix $\mathsf{F}^\mathsf{T}\mathsf{F}$ is invertible. The solution of problem (2.45) (and therefore of problem (2.43)) is given by:

$$\mathbf{x} = \left(\mathsf{F}^\mathsf{T}\mathsf{F}\right)^{-1}\mathsf{F}^\mathsf{T}\mathbf{y}. \tag{2.46}$$

Since the matrix $\left(\mathsf{F}^\mathsf{T}\mathsf{F}\right)^{-1}\mathsf{F}^\mathsf{T}$ has full column rank, it is the Moore-Penrose pseudo-inverse matrix of F and is denoted $\mathsf{F}^\dagger$. It is generally a bad practice to explicitly compute the pseudo-inverse matrix (Golub and Van Loan, 1996a). We will see later in this document several methods designed to efficiently solve the linear system of equation (2.45). Note that F being full column rank necessary implies that F must have more rows than columns, *i.e.* $m \geq n$. There exists several ways to derive the normal equations. For instance, one can get them by writing the optimality conditions of problem (2.43), *i.e.* :

$$\frac{\partial}{\partial \mathbf{x}}\|\mathsf{F}\mathbf{x} - \mathbf{y}\|^2 = 0. \tag{2.47}$$

The normal equations are trivially derived by noticing that $\|\mathsf{F}\mathbf{x} - \mathbf{y}\|^2 = (\mathsf{F}\mathbf{x} - \mathbf{y})^\mathsf{T}(\mathsf{F}\mathbf{x} - \mathbf{y})$. Another way to obtain the normal equations is to consider the over-determined linear system of equations:

$$\mathsf{F}\mathbf{x} = \mathbf{y}. \tag{2.48}$$

Since this linear system of equations is over-determined, it does not have necessarily an exact solution. In this case, we can seek for an approximate solution by finding the point of the column space of F which is as close as possible to the point $\mathbf{y}$. The shortest distance between a point and an hyperplane is the distance between the point and its orthogonal projection into the hyperplane. This is illustrated by figure 2.7. In other words, the vector $\mathbf{y} - \mathsf{F}\mathbf{x}$ must lie in the left nullspace of F, which can be written:

$$\mathsf{F}^\mathsf{T}(\mathbf{y} - \mathsf{F}\mathbf{x}) = 0. \tag{2.49}$$

Equation (2.49) are exactly equivalent to the normal equations.

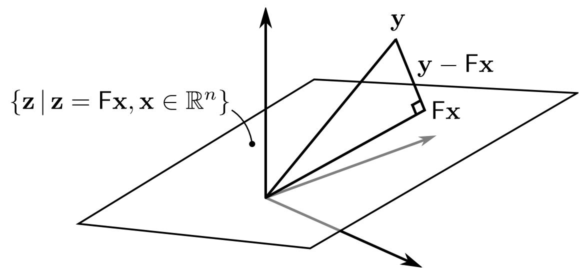

$\mathbf{y}$
$\mathbf{y} - \mathsf{F}\mathbf{x}$
$\mathsf{F}\mathbf{x}$
$\{\mathbf{z} \,|\, \mathbf{z} = \mathsf{F}\mathbf{x}, \mathbf{x} \in \mathbb{R}^n\}$

Figure 2.7: Graphical interpretation of the normal equations.

2.2.2.7 Levenberg-Marquardt

The Levenberg algorithm is another method to solve a least-squares minimization problem proposed by (Levenberg, 1944). (Marquardt, 1963) proposed a slight variation of the initial method of Levenberg known as the Levenberg-Marquardt algorithm. The content and the presentation of this section is inspired by (Madsen et al., 2004). We use the same notation than in the section dedicated to the Gauss-Newton algorithm. The normal equations can be used for the step in the Gauss-Newton algorithm. This step, denoted $\boldsymbol{\delta}_{gn}$ in this section, can thus be written $\boldsymbol{\delta}_{gn} = (\mathbf{J}^\mathsf{T}\mathbf{J})^{-1}\mathbf{J}\mathbf{f}$, where $\mathbf{J}$ is the Jacobian matrix of the function $\mathcal{F}$ evaluated at $\mathbf{x}$, and

$\mathbf{f}^\mathsf{T} = \begin{pmatrix} f_1(\mathbf{x}) & \dots & f_m(\mathbf{x}) \end{pmatrix}$. The Levenberg and the Levenberg-Marquardt algorithms are damped versions of the Gauss-Newton method. The step for the Levenberg algorithm, denoted $\boldsymbol{\delta}_l$, is defined as:

$$(\mathsf{J}^\mathsf{T}\mathsf{J} + \lambda \mathbf{I})\boldsymbol{\delta}_l = \mathsf{J}\mathbf{f}, \tag{2.50}$$

while the step for the Levenberg-Marquardt algorithm, denoted $\boldsymbol{\delta}_{lm}$, is defined as:

$$(\mathsf{J}^\mathsf{T}\mathsf{J} + \lambda \mathrm{diag}(\mathsf{J}^\mathsf{T}\mathsf{J}))\boldsymbol{\delta}_{lm} = \mathsf{J}\mathbf{f}. \tag{2.51}$$

From now on, we only consider the Levenberg-Marquardt algorithm[2]. The linear system of equation (2.51) is called the *augmented normal equations*. The value λ is a positive value named the damping parameter. It plays several roles. First, the matrix $\mathsf{J}^\mathsf{T}\mathsf{J} + \lambda \mathrm{diag}(\mathsf{J}^\mathsf{T}\mathsf{J})$ is positive definite. Therefore, $\boldsymbol{\delta}_{lm}$ is necessarily a descent direction. For large values of λ, $\boldsymbol{\delta}_{lm} \approx -\frac{1}{m}\boldsymbol{\nabla} f$. In this case, the Levenberg-Marquardt algorithm is almost a gradient descent method (with a short step). This is a good strategy when the current solution is far from the minimum. On the contrary, if λ is a small value, then the Levenberg-Marquardt step $\boldsymbol{\delta}_{lm}$ is almost identical to the Gauss-Newton step $\boldsymbol{\delta}_{gn}$. This is a desired behaviour for the final steps of the algorithm since, near the minimum, the convergence of the Gauss-Newton method can be almost quadratic.

Both the length and the direction of the step are influenced by the damping parameter. Consequently, there is no need for a line-search procedure in the iterations of this algorithm. The value of the damping parameter λ is updated along with the iterations according to the following strategy. If the current λ results in an improvement of the cost function, then the step is applied and λ is divided by a constant ν (with, typically, $\nu = 2$). On the contrary, if the step resulting of the current λ increases of the function, the step is discarded and λ is multiplied by ν. This is the most basic stratagem for updating the damping parameter. There exists more evolved approach, such as the one presented in (Madsen et al., 2004). The whole Levenberg-Marquardt method is given in algorithm 5. Figure 2.8 gives an illustration of the Levenberg-Marquardt algorithm on the Rosenbrock function.

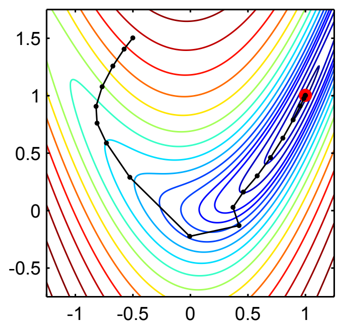


Figure 2.8: Illustration of the Levenberg-Marquardt algorithm for optimizing the Rosenbrock function. In this particular example, this algorithm took 25 iterations to reach the minimum.

2.2.2.8 Cholesky Factorization

A word about linear systems of equations. We now provides some tools for solving efficiently a linear system of equations. This is of great interest in optimization since we have seen that solving a linear least-squares minimization problem is equivalent to finding a solution to the linear system formed by the normal equations. Besides, other optimization problems (not necessarily written as linear least-squares problem) involves a linear

[2]Note that in the recent literature, the Levenberg-Marquardt algorithm is often confused with the Levenberg algorithm.

Algorithm 5: Levenberg-Marquardt algorithm

```
input : $f : \mathbb{R}^n \to \mathbb{R}$ a function such that $f(\mathbf{x}) = \sum_{i=1}^{m} (f_i(\mathbf{x}))^2$ where all the $f_i$ are differentiable
        functions from $\mathbb{R}^n$ to $\mathbb{R}$
        $\mathbf{x}^{(0)}$ an initial solution
output: $\mathbf{x}^\star$, a local minimum of the cost function $f$.
1  begin
2      $k \leftarrow 0$
3      $\lambda \leftarrow \max \operatorname{diag}(\mathsf{J}^\mathsf{T}\mathsf{J})$
4      $\mathbf{x} \leftarrow \mathbf{x}^{(0)}$
5      while STOP-CRIT and $(k < k_{max})$ do
6          Find $\boldsymbol{\delta}$ such that $(\mathsf{J}^\mathsf{T}\mathsf{J} + \lambda \operatorname{diag}(\mathsf{J}^\mathsf{T}\mathsf{J}))\boldsymbol{\delta} = \mathsf{J}^\mathsf{T}\mathbf{f}$
7          $\mathbf{x}' \leftarrow \mathbf{x} + \boldsymbol{\delta}$
8          if $f(\mathbf{x}') < f(\mathbf{x})$ then
9              $\mathbf{x} \leftarrow \mathbf{x}'$
10             $\lambda \leftarrow \frac{\lambda}{\nu}$
11         else
12             $\lambda \leftarrow \nu\lambda$
13         $k \leftarrow k + 1$
14     return $\mathbf{x}$
```

least-squares problem in their iterations. Let us consider the following linear system of equations:

$$\mathsf{A}\mathbf{x} = \mathbf{b}, \tag{2.52}$$

with $\mathsf{A} \in \mathbb{R}^{n \times n}$ a square matrix, and $\mathbf{b} \in \mathbb{R}^n$. The problem equation (2.52) admits a single solution if the determinant of the matrix A is not null. In this case, the solution is theoretically given with the inverse matrix of A, *i.e.* $\mathbf{x} = \mathsf{A}^{-1}\mathbf{b}$. However, explicitly forming the matrix A^{-1} is generally not the best way of solving a linear system of equations (Golub and Van Loan, 1996a). This stems from the fact that a direct computation of the inverse matrix is numerically unstable. Besides, it is not possible to exploit sparse properties of a matrix when explicitly forming the inverse matrix. A very basic tool that can be used to solve a linear system is the Gauss elimination algorithm (Golub and Van Loan, 1996a; Press et al., 1992). The main advantage of this algorithm is that it does not need strong assumption on the matrix of the system (except that it is square and invertible, of course). Here, we will not give the details of the Gauss elimination algorithm. Instead, we will present other algorithms more suited to the intended purpose, *i.e.* optimization algorithms.

The Cholesky factorization. The Cholesky factorization is a method for solving a linear system of equations. If the matrix A is positive definite, then the Cholesky factorization of A is:

$$\mathsf{A} = \mathsf{R}^\mathsf{T}\mathsf{R}, \tag{2.53}$$

with $\mathsf{R} \in \mathbb{R}^{n \times n}$ an upper triangular matrix with strictly positive diagonal entries. Note that, for example, the left-hand side of the augmented normal equations used in the iterations of the Levenberg-Marquardt algorithm is a positive definite matrix. If we substitute the matrix A in equation (2.52) by its Cholesky factorization, we obtain:

$$\mathsf{R}^\mathsf{T}\mathsf{R}\mathbf{y} = \mathbf{b}. \tag{2.54}$$

If we note $\mathbf{y}' = \mathsf{R}\mathbf{y}$ then solving equation (2.52) is equivalent to successively solving the two linear systems $\mathsf{R}^\mathsf{T}\mathbf{y}' = \mathbf{b}$ and $\mathsf{R}\mathbf{x} = \mathbf{y}'$. This is easily and efficiently done using a back-substitution process because of the triangularity of the matrices R and R^T.

Although efficient, this approach works only if the problem is well-conditioned. If not, some of the diagonal entries of the matrix R are close to zero and the back-substitution process becomes numerically unstable. In practice, the Cholesky factorization is particularly interesting for sparse matrices. For instance, the package cholmod of (Davis and Hager, 1999), dedicated to sparse matrices, heavily uses Cholesky factorization.

2.2.2.9 QR Factorization

We now give a method based on the QR factorization for solving an over-determined linear least-squares minimization problem of the form:

$$\min_{\mathbf{x}} \|\mathsf{F}\mathbf{x} - \mathbf{y}\|^2, \tag{2.55}$$

with $\mathsf{F} \in \mathbb{R}^{m\times n}$, $m \geq n$, and $\mathbf{y} \in \mathbb{R}^m$. As we have seen in section 2.2.2.6, solving problem (2.55) amounts to solve the normal equations, namely:

$$\mathsf{F}^\mathsf{T}\mathsf{F}\mathbf{x} = \mathsf{F}^\mathsf{T}\mathbf{y}. \tag{2.56}$$

The QR factorization of the matrix F is:

$$\mathsf{F} = \mathsf{QR} = \begin{bmatrix} \mathsf{Q}_1 & \mathsf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathsf{R}_1 \\ \mathbf{0} \end{bmatrix}, \tag{2.57}$$

with $\mathsf{Q} \in \mathbb{R}^{m\times m}$ an orthonormal matrix, $\mathsf{Q}_1 \in \mathbb{R}^{m\times n}$, $\mathsf{Q}_2 \in \mathbb{R}^{m\times(m-n)}$, and $\mathsf{R}_1 \in \mathbb{R}^{n\times n}$ an upper triangular matrix. If the matrix F has full column rank, then the columns of the matrix Q_1 form an orthonormal basis for the range of F (Golub and Van Loan, 1996b). If we replace the matrix F by its QR factorization into the normal equations, we obtain:

$$\begin{bmatrix} \mathsf{R}_1^\mathsf{T} & \mathbf{0} \end{bmatrix} \mathsf{Q}^\mathsf{T}\mathsf{Q} \begin{bmatrix} \mathsf{R}_1 \\ \mathbf{0} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathsf{R}_1^\mathsf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathsf{Q}_1 \\ \mathsf{Q}_2 \end{bmatrix} \mathbf{y}. \tag{2.58}$$

This last expression can be simplified using the fact that the matrix Q_1 is orthonormal:

$$\mathsf{R}_1\mathbf{x} = \mathsf{Q}_1^\mathsf{T}\mathbf{y}. \tag{2.59}$$

As for the approach based on the Cholesky factorization, the linear system of equation (2.59) can be easily solved with a back-substitution algorithm.

2.2.2.10 Singular Value Decomposition

The singular value decomposition is a tool that can be used to solve certain linear least-squares minimization problems. The singular value decomposition of a real matrix $\mathsf{A} \in \mathbb{R}^{m\times n}$ is given by:

$$\mathsf{A} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}, \tag{2.60}$$

with $\mathsf{U} \in \mathbb{R}^{m\times n}$ a unitary matrix, $\Sigma \in \mathbb{R}^{n\times n}$ a diagonal matrix with non-negative elements on its diagonal, and $\mathsf{V} \in \mathbb{R}^{n\times n}$ a unitary matrix. The diagonal entries of Σ, named the singular values, are generally sorted from the largest to the smallest. If the matrix A has full column rank then the diagonal entries of Σ are all strictly positive. In fact, the number of strictly positive elements on the diagonal of Σ is the rank of the matrix A.

Non-homogeneous linear least-squares. Once again, we consider the problem of minimizing the over-determined linear least-squares function $\|\mathsf{F}\mathbf{x} - \mathbf{y}\|^2$. This least-squares problem is called non-homogeneous because there exists a non-null right-hand side $\mathbf{y}$. The singular value decomposition can be used to solve this problem. Indeed, if we replace the matrix F in the normal equations by its singular value decomposition $\mathsf{F} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$, we obtain:

$$\mathsf{V}\Sigma\mathsf{U}^\mathsf{T}\mathsf{U}\Sigma\mathsf{V}^\mathsf{T}\mathbf{x} = \mathsf{V}\Sigma\mathsf{U}^\mathsf{T}\mathbf{y}. \tag{2.61}$$

Using the fact that U is a unitary matrix and that Σ is a diagonal matrix, we can write that:

$$\mathbf{x} = \mathsf{V}\Sigma^{-1}\mathsf{U}^\mathsf{T}\mathbf{y}. \tag{2.62}$$

In other words, we have that $\mathsf{F}^\dagger = \mathsf{V}\Sigma^{-1}\mathsf{U}^\mathsf{T}$. As for the other algorithms, the computational complexity of this method is in $\mathcal{O}(n^3)$. The main advantage of the SVD approach for linear least-squares problem is when the matrix F is almost rank deficient. In this case, the smallest singular values in Σ are very close to zero. Therefore, a simple computation of Σ^{-1} would lead to unsatisfactory results (the inverse of a diagonal matrix being obtained by taking the reciprocals of the diagonal entries of Σ). In this case, one can replace the inverse matrix Σ^{-1} in equation (2.62) by the pseudo-inverse matrix $\Sigma^\dagger$. The pseudo-inverse of a diagonal matrix $\Sigma = \operatorname{diag}\begin{pmatrix}\sigma_1 & \cdots & \sigma_{n'} & \mathbf{0}\end{pmatrix} \in \mathbb{R}^{n\times n}$ (with $n' \leq n$) is the matrix $\Sigma^\dagger = \operatorname{diag}\begin{pmatrix}\frac{1}{\sigma_1} & \cdots & \frac{1}{\sigma_{n'}} & \mathbf{0}\end{pmatrix} \in \mathbb{R}^{n\times n}$.

Homogeneous linear least-squares. We now consider the case of the homogeneous linear least-squares minimization problem. It corresponds to the following minimization problem:

$$\min_{\mathbf{x}} \sum_{i=1}^{m} (f_i(\mathbf{x}))^2, \tag{2.63}$$

where the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are linear with respect to $\mathbf{x}$, *i.e.* $\exists \mathbf{f}_i \in \mathbb{R}^n$ such that $f_i(\mathbf{x}) = \mathbf{f}_i^\mathsf{T}\mathbf{x}$ for all $i \in [\![1, m]\!]$. This minimization problem can be written in matrix form:

$$\min_{\mathbf{x}} \|\mathsf{F}\mathbf{x}\|^2, \tag{2.64}$$

with $\mathsf{F} = \begin{bmatrix}\mathbf{f}_1 & \ldots & \mathbf{f}_m\end{bmatrix}^\mathsf{T} \in \mathbb{R}^{m\times n}$. An obvious solution to problem (2.64) is $\mathbf{x} = \mathbf{0}$. Of course, this dummy solution is generally not interesting. Instead of solving problem (2.64), one replaces it with the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathsf{F}\mathbf{x}\|^2 \\ \text{subject to} \quad & \|\mathbf{x}\| = 1. \end{aligned} \tag{2.65}$$

Problem (2.65) is easily solved using the singular value decomposition. Indeed, let $\mathsf{F} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$ be the singular value decomposition of F. If we write $\mathbf{x}' = \mathsf{V}^\mathsf{T}\mathbf{x}$, then problem (2.65) is equivalent to:

$$\begin{aligned} \min_{\mathbf{x}'} \quad & \|\Sigma\mathbf{x}'\|^2 \\ \text{subject to} \quad & \|\mathbf{x}'\| = 1. \end{aligned} \tag{2.66}$$

This stems from the fact that the matrices U and V are unitary and, consequently, we have that $\|\mathsf{U}\Sigma\mathsf{V}^\mathsf{T}\mathbf{x}\| = \|\Sigma\mathsf{V}^\mathsf{T}\mathbf{x}\|$ and $\|\mathbf{x}'\| = \|\mathsf{V}^\mathsf{T}\mathbf{x}\| = \|\mathbf{x}\|$. Remind the fact that for a full column rank matrix the diagonal elements of Σ are $\sigma_1 \geq \ldots \geq \sigma_n > 0$. Therefore, an evident minimizer of problem (2.66) is $\mathbf{x}' = \begin{pmatrix}0 & \cdots & 0 & 1\end{pmatrix}^\mathsf{T}$. Finally, since we have that $\mathbf{x} = \mathsf{V}\mathbf{x}'$, the solution of our initial homogeneous linear least-squares problem is given by the last column of the matrix V. If the matrix F has not full column rank, then the same principle can be applied except that one may take the column of V corresponding the last *non-zero* singular value.

2.2.2.11 Iteratively Reweighed Least Squares

The method of Iteratively Reweighed Least Squares (IRLS) is an optimization algorithm designed to minimize a function $f : \mathbb{R}^n \to \mathbb{R}$ of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x}) \|f_i(\mathbf{x}) - y_i\|^2, \tag{2.67}$$

where w is a function from $\mathbb{R}^n$ to $\mathbb{R}$. For instance, such a cost function arises when M-estimators are involved. IRLS is an iterative method for which each step involves the resolution of the following weighted least squares problem:

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \sum_{i=1}^{n} w(\mathbf{x}^{(k)}) \|f_i(\mathbf{x}) - y_i\|^2. \tag{2.68}$$

2.2.2.12 Golden Section Search

We now arrive at the last optimization algorithm presented in this document: the golden section search algorithm. This algorithm allows one to minimize a monovariate function $f : \mathbb{R} \to \mathbb{R}$. The minimization necessarily takes place over a given interval. The function f has to be continuous and unimodal on this interval for the golden section search algorithm to work. It can be used as the line search procedure for the previously presented algorithm of this section. It was first proposed in (Kiefer, 1953) and the refined in (Avriel and Wilde, 1966). The general principle of the golden section search is to bracket the location of the minimum of f. This is achieved by updating and refining a set of 3 locations $x_1 < x_2 < x_3$ with the assumptions that $f(x_2) \leq f(x_1)$ and $f(x_2) \leq f(x_3)$. These locations are then updated according to the value taken by f at a new point x_4 such that $x_2 < x_4 < x_3$. If $f(x_4) < f(x_2)$ then the 3 locations become $x_2 < x_4 < x_3$. Otherwise, *i.e.* if $f(x_4) \geq f(x_2)$, they become $x_1 < x_2 < x_4$. The principle is illustrated in figure 2.9.

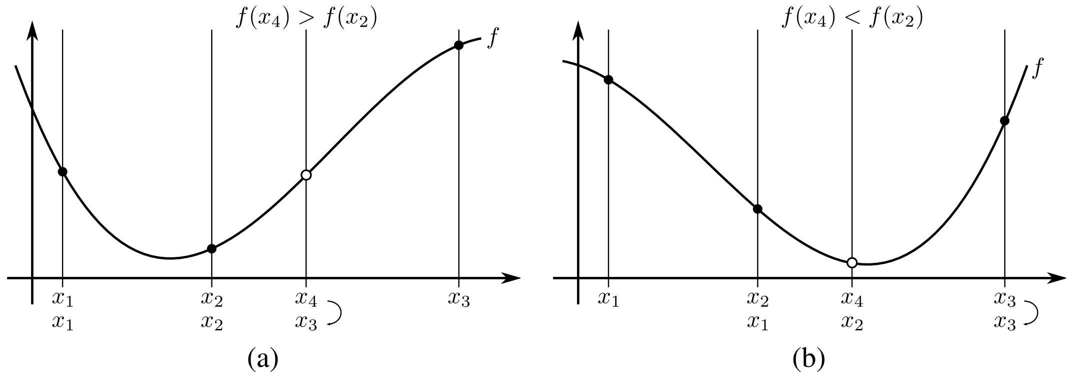


Figure 2.9: Illustration of an update of the golden section search algorithm. (a) If the function to optimize is uni-modal and if $f(x_4) > f(x_2)$ then the minimum of the function is necessarily located on the interval $[x_1, x_4]$. (b) On the contrary, if $f(x_4) < f(x_2)$ then the minimum of the function lies in the interval $[x_2, x_3]$.

As just described, the update of the 3 locations implies that the minimum of a function will be located either in the interval $[x_1, x_4]$ or $[x_2, x_3]$. The principle of the golden section search algorithm is to impose the length of these two intervals: they must be equal. It implies that we must have:

$$x_3 - x_4 = \varphi(x_4 - x_2), \tag{2.69}$$

where φ is the golden ratio, *i.e.* $\varphi = \frac{1+\sqrt{5}}{2}$.

2.3 Parametric Models of Function

As it will be seen in the rest of this document, the vast majority of the problems treated in this thesis may be cast into *parameter estimation problems*. The fundamental concepts underlying such problems will be detailed in chapter 3. For now, let us just say that parameter estimation problems consists in finding the parameters of a parametric model so that the resulting function is 'close to' a given set of data points. In a nutshell, a parametric model of function is a function whose exact behaviour is controlled by a set of parameters. For instance, a polynomial can be considered as a parametric model of function with the polynomial coefficients as parameters. A more precise definition of the parametric model of function will be given in section 3.1.1.1. In this section, we detail some particular but generic parametric models of function which are the ones mostly used in this document. Although we do not pretend to be fully exhaustive, we spend time in explaining the basics on splines and B-splines. The interested reader can deepen this topic with this readings (de Boor, 2001; Dierckx, 1993; Farin, 1997; Malgouyres, 2005). We also present other useful parametric models such as rational splines and radial basis functions.

2.3.1 Splines

The word *spline* is a generic term that designates a parametric model of function. This word has been used in many contexts (computer aided geometric design, approximation theory, computer vision) with a meaning that slightly varies from one domain to the other. In this section, we first fix the meaning of the word spline as we understand it in this document. We then give the basic mathematical definition of a mono-dimensional spline. We also give the fundamental properties of such objects.

Historical note. The term spline originally designated a physical object used to draw smooth curves. It was widely used by engineers and architects to craft blue prints. It was made of a flexible strip that was fixed at some points. An illustration of such splines is given in figure 2.10. Of course, in this document we will be more interested in the mathematical form of the splines than in their physical counterparts.

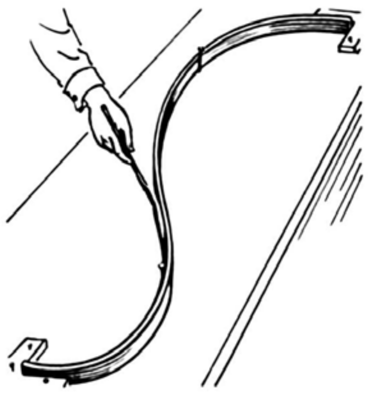


Figure 2.10: An historical spline (credits: Pearson Scott Foresman).

General definition. In its most general definition, a spline is a piecewise polynomial function. Note that this definition is independent of any particular form. For instance, the B-splines (see section 2.3.2) are just a particular representation of a spline. Note also that this basic definition does not make any assumption on the continuity of the function. In particular, the polynomial pieces of a spline may or may not satisfy continuity conditions at their junction. Examples of splines are given in figure 2.11.

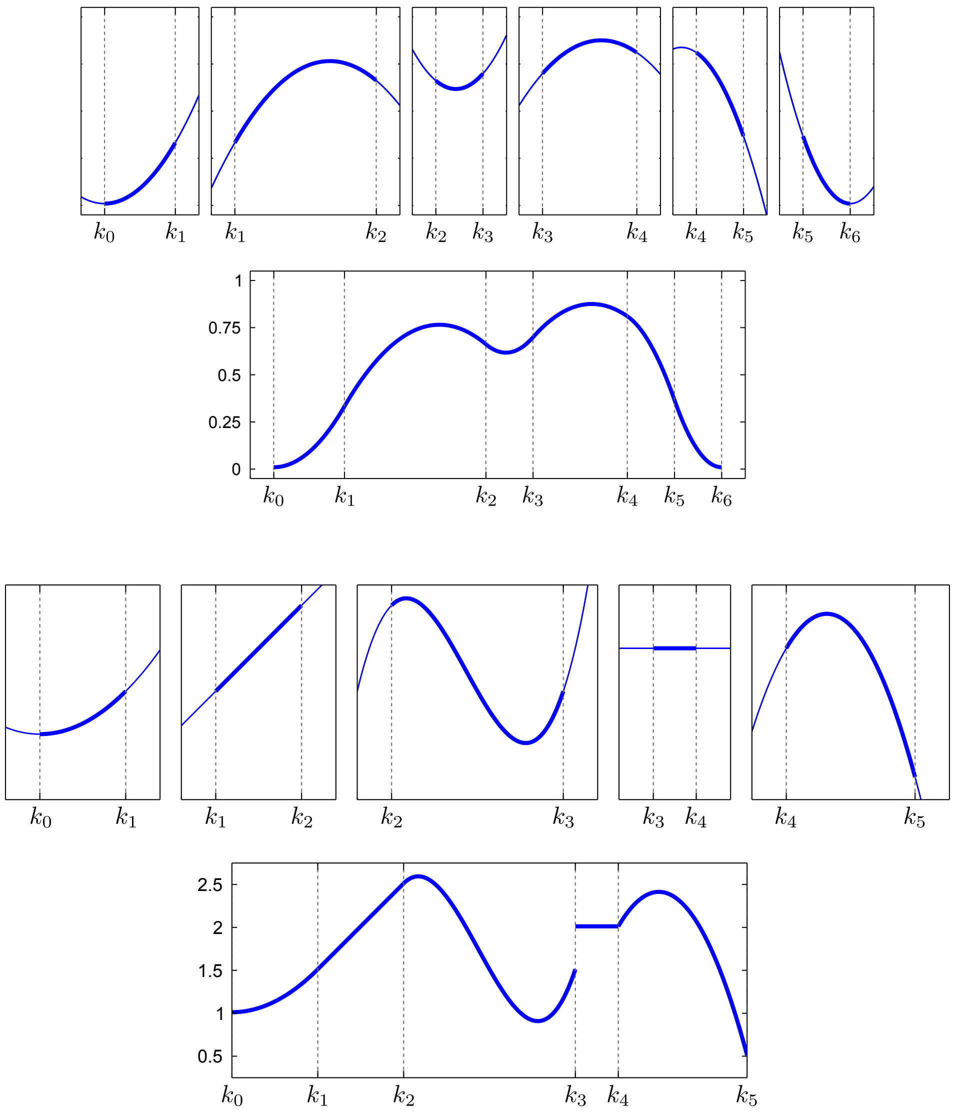


Figure 2.11: Examples of spline functions. A spline is a piecewise polynomial function. The width of each piece is not necessarily the same. In the most general definition of a spline, there may (top) or may not (bottom) be continuity conditions between the polynomial pieces. Top: a spline made of 6 polynomial pieces of degree 2 with $\mathcal{C}^1$ continuity over the whole domain $[k_0, k_6]$. Bottom: a spline made of 5 polynomials of various degrees with various continuity conditions at the knots.

Mathematical definition. Mathematically speaking, several elements are required to define a spline: a degree, a set of contiguous intervals, and, of course, polynomials on each of this intervals. The degree of a spline, denoted d in this section, is the maximal degree of the polynomial pieces. Equivalently, one may talk about the order of the spline, which is $d+1$ for a spline of degree d. The intervals are defined using a *knot sequence* $k_0 < \ldots < k_n$ which is a strictly increasing sequence of $n+1$ real numbers. The values $k_0, \ldots, k_n$ are called the *knots*. These knots defines n *knot intervals* $[k_i, k_{i+1}]$ for $i \in [\![0, n-1]\!]$. On each knot interval, a spline of degree d is defined by a polynomial of degree at most d. The *natural definition domain* of a spline is the interval $[k_0, k_n]$. For instance, a spline $s : [k_0, k_n] \to \mathbb{R}$ may be defined as:

$$s(x) = \sum_{i=0}^{d} p_{ij}(x-k_j)^i \qquad \begin{cases} \text{if } x \in [k_j, k_{j+1}[\text{ and } j \in [\![0, n-2]\!] \\ \text{if } x \in [k_{n-1}, k_n] \end{cases} \tag{2.70}$$

where the values p_{ij} for $i \in [\![0, d]\!]$ are the coefficients of the j-th polynomial piece ($j \in [\![0, n-1]\!]$). Note that equation (2.70) is just a particular way of expressing a spline. It is the most basic form for expressing the fact that s is a piecewise polynomial function. Other more interesting representations will be given later in this chapter.

In addition to these basic elements, the polynomial pieces may be stitched to each other using continuity conditions. These conditions are written:

$$\lim_{\substack{k \to k_i \\ k < k_i}} \frac{\partial^l s}{\partial x^l}(k) = \lim_{\substack{k \to k_i \\ k > k_i}} \frac{\partial^l s}{\partial x^l}(k) \qquad i \in [\![1, n-1]\!], l \in [\![0, c_i - 1]\!], \tag{2.71}$$

where $c_i \in [\![0, d-1]\!]$ is the required class of continuity at the knot k_i and where the convention $\frac{\partial^0 s}{\partial x^0} = s$ is used. A common choice is $c_1 = \ldots = c_{n-1} = d-1$. We name this choice the *full continuity constraints*. In this case, the spline s belongs to $\mathcal{C}^{d-1}([k_0, k_n])$, which is the maximal continuity class for a piecewise polynomial function.

Number of degrees of freedom. Let us note $\mathcal{S}_d(k_0, \ldots, k_n)$ the vector space of the splines of degree d with knots $k_0, \ldots, k_n$. Let s be a spline of $\mathcal{S}_d(k_0, \ldots, k_n)$. The number of degrees of freedom of the spline s is the dimension of the vector space $\mathcal{S}_d(k_0, \ldots, k_n)$. It is the total number of polynomial coefficients minus the number of continuity conditions. In the full continuity case, the number of degrees of freedom of a spline is $n(d+1) - (n-1)d = n + d$.

Truncated power functions. A particular way to expressing the splines relies on the truncated power functions. A truncated power function is denoted with the symbol $+$. It is defined as:

$$(x-c)_+^k = \begin{cases} (x-c)^k & \text{if } x \geq c \\ 0 & \text{otherwise.} \end{cases} \tag{2.72}$$

It can be proved (Dierckx, 1993) that any spline s of $\mathcal{S}_d(k_0, \ldots, k_n)$ can be uniquely written as a linear combination of the canonical monomial x^i and of the truncated power functions of degree d:

$$s(x) = \sum_{i=0}^{d} \alpha_i x^i + \sum_{i=1}^{n-1} \beta_j (x - k_i)_+^d, \tag{2.73}$$

where $\{\alpha_i\}_{i=0}^d$ and $\{\beta_i\}_{i=1}^{n-1}$ are $n+d$ real values. The $n+d$ polynomials $1, x, \ldots, x^d, (x-k_1)_+^d, \ldots, (x-k_{n-1})_+^d$ form a basis of the vector space $\mathcal{S}_d(k_0, \ldots, k_n)$. However, this family of functions constitutes an ill-conditioned basis which makes the expression of equation (2.73) numerically unstable. Therefore, equation (2.73) is not suited for computations. In the next section, we explore another representation of splines, the so-called B-splines, which is more convenient for practical use.

2.3.2 The B-spline Representation

In this section, we give the definitions and the properties of the parametric model mostly used in this thesis: the splines expressed as a linear combination of B-splines. The B-splines are a set of piecewise polynomial functions that defines a suitable basis for the vector space of the splines (the *B* in *B*-spline stands for *B*asis). As remarked by (de Boor, 1972), they were first introduced in (Curry and Schoenberg, 1947; Schoenberg, 1946). We start this section with the construction and the definitions of the basic building block, *i.e.* the B-spline functions. We then give the details on the general representation of splines with the B-splines basis functions. We end this section with a special case of particular interest: the uniform cubic B-splines.

2.3.2.1 The B-Splines Functions

The B-spline $N_{i,d+1}$ of degree d (order $d + 1$) with knots $k_i < \ldots < k_{i+d+1}$ is defined recursively with the following relation:

$$N_{i,d+1}(x) = \frac{x - k_i}{k_{i+d} - k_i} N_{i,d}(x) + \frac{k_{i+d+1} - x}{k_{i+d+1} - k_{i+1}} N_{i+1,d}(x), \tag{2.74}$$

$$N_{i,1}(x) = \begin{cases} 1 & \text{if } x \in [k_i, k_{i+1}[\\ 0 & \text{otherwise.} \end{cases} \tag{2.75}$$

These formula are only one way of defining the B-splines. They are called the *Cox de Boor recursion formula*. One may remark from equation (2.74) and equation (2.75) that B-splines are themselves splines.

Properties. Here comes a list of the most basic properties of the B-spline functions. These properties were extensively studied in (de Boor, 2001; Schumaker, 1981).

Positivity. The B-splines are everywhere positive or null:

$$N_{i,d+1}(x) \geq 0 \qquad \forall x \in \mathbb{R}. \tag{2.76}$$

Local support. The support of the B-splines is bounded, *i.e.* they are non-zero only over a their natural definition domain:

$$N_{i,d+1}(x) = 0 \text{ if } x \notin [k_i, k_{i+d+1}] \tag{2.77}$$

Boundary values.

$$N_{i,d+1}(k_i) = N_{i,d+1}(k_{i+d+1}) = 0 \tag{2.78}$$

Continuity. For a given degree d, the B-splines belongs to the highest possible class of continuity for a piecewise polynomial function:

$$N_{i,d+1} \in \mathcal{C}^{d-1}([k_i, k_{i+d+1}]) \tag{2.79}$$

Derivatives of a B-spline. The derivative of a B-spline of degree d is a linear combination of B-splines of degree $d - 1$:

$$N'_{i,d+1}(x) = d \left(\frac{N_{i,d}(x)}{k_{i+d} - k_i} - \frac{N_{i+1,d}(x)}{k_{i+d+1} - k_{i+1}} \right) \tag{2.80}$$

Coincident knots. So far, we made the assumptions that the knots were all different. The definition of the B-splines may be extended to coincident knots. This allows one to weaken the continuity conditions at the position of these coincident knots. Let r knots in the set $\{k_i, \ldots, k_{i+d+1}\}$ be coincident at the point c (with $2 \leq r \leq d + 1$). The B-spline have continuous derivatives up to order $d - r$ at c and is discontinuous at c if $r = d + 1$. Figure 2.12 illustrates the influence of coincident knots on the B-spline basis functions.

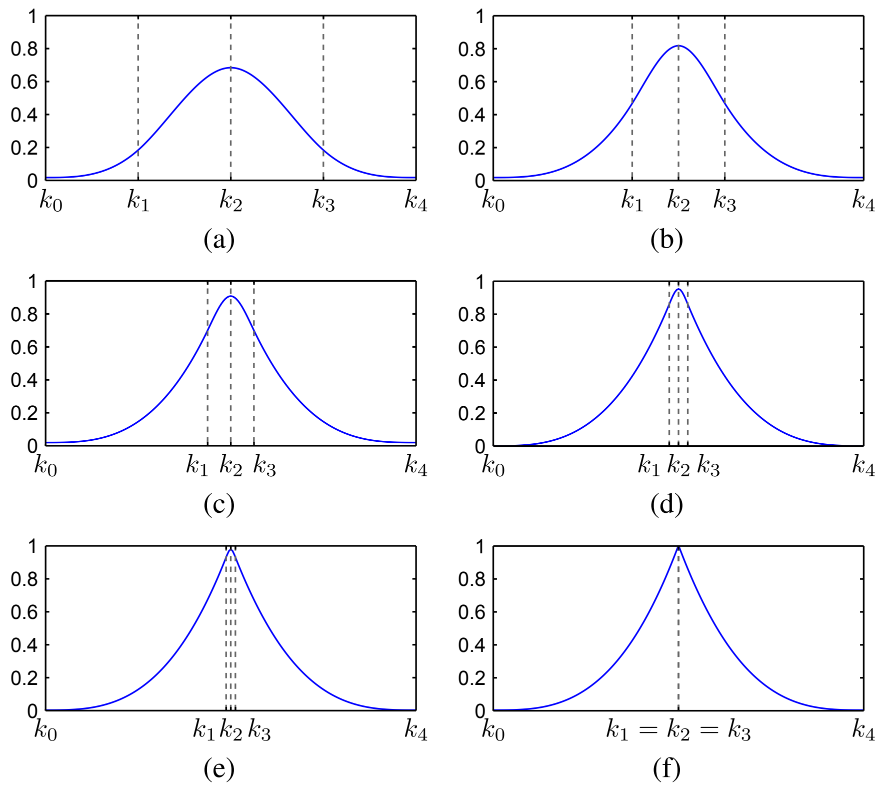


Figure 2.12: Influence of coincident knots on a B-spline basis function. Here we consider a B-spline of degree $d = 3$. The knots $k_0, \dots, k_4$ form a strictly increasing sequence in the sub-figures (a) to (e). In sub-figure (f), k_1, k_2, and k_3 are coincident. In (a-e), the B-spline belongs to $\mathcal{C}^3$. In (f), it is only $\mathcal{C}^{d-3} = \mathcal{C}^0$ at the position of the knot $k_1 = k_2 = k_3$.

Multiple knots, breaks. Let $k_0 \leq \dots \leq k_n$ be a knot sequence with coincidence allowed. The break sequence $b_0 < \dots < b_m$ (with $m \leq n$) is the *strictly* increasing sequence of real numbers build by removing the redundancies from the knot sequence:

$$b_0 = k_0 < \dots < b_i = k_{i_1} = \dots = k_{i_1+r_1} < \dots < b_j = k_{i_2} = \dots = k_{i_2+r_2} < \dots < b_m = k_n. \tag{2.81}$$

The *multiplicity* of the breaks b_i is the number of knots that corresponds to the break b_i. It is denoted with the operator mult. For instance, in equation (2.81), we have that $\mathrm{mult}(b_j) = r_2 + 1$.

2.3.2.2 Splines as Linear Combinations of B-Splines

Splines can be written as a linear combination of B-splines basis functions. Such splines are often referred to as B-splines. This may be a bit confusing but it is shorter than ‘splines as a linear combination of B-splines’. We will use this language shortcut in the rest of this document. The expression ‘B-spline basis function’ will be utilised to distinguish between the splines and the basis functions. Let us take back the knot sequence we used when we introduced the spline functions (see section 2.3.1): $\mathbf{k} = k_0 \leq \dots \leq k_n$[3]. Since a single B-spline of degree d spans $d+1$ knot intervals, $n-d$ independent B-splines can be defined using $\mathbf{k}$. The vector space $\mathcal{S}_d(k_0, \dots, k_n)$ has $n+d$ dimensions. Therefore, we need $2d$ supplemental B-splines to form a basis of $\mathcal{S}_d(k_0, \dots, k_n)$. These B-splines may be defined by adding $2d$ extra knots to the initial knot sequence satisfying the following conditions:

$$\begin{aligned} k_{-d} \leq \dots \leq k_{-1} \leq k_0, \\ \text{and } k_n \leq k_{n+1} \leq \dots \leq k_{n+d}. \end{aligned} \tag{2.82}$$

[3]From now on, multiple knots are allowed.

These additional knots are named the *boundary knots*. They must satisfy the conditions of equation (2.82) but may be otherwise arbitrary. There exists some common choice for defining these boundary knots, later reviewed in this document.

A B-spline s is uniquely written as a linear combination of the $n+d$ basis functions $N_{i,d+1}$ for $i \in [\![-d, n-1]\!]$:

$$s(x) = \sum_{i=-d}^{n-1} w_i N_{i,d+1}(x), \tag{2.83}$$

where the real values w_i ($i \in [\![-d, n-1]\!]$) are named the *weights* (or the coefficients) of the B-spline. For vector-valued splines (see after), the entities that corresponds to the weights will be called the *control points*. Note that equation (2.83) will often be written in vector notation:

$$s(x) = \mathbf{n}_x^\mathsf{T} \mathbf{w} = \mathbf{w}^\mathsf{T} \mathbf{n}_x, \tag{2.84}$$

where $\mathbf{w}^\mathsf{T} = \begin{pmatrix} w_{-d} & \dots & w_{n-1} \end{pmatrix}$ and $\mathbf{n}_x^\mathsf{T} = \begin{pmatrix} N_{-d,d+1}(x) & \dots & N_{n-1,d+1}(x) \end{pmatrix}$.

Coincident knots. The dimension of the vector $\mathcal{S}_d(k_0, \dots, k_n)$ is reduced with coincident knots. Indeed, it corresponds to the number m ($m \leq n$) of break intervals multiplied by the number of polynomial coefficients on each break interval ($d+1$) minus the number of continuity conditions. This is consistent with the fact that the order of continuity of a spline is reduced at the location of coincident knots. Indeed, for a break b_i with multiplicity r_i, we have $d - r_i$ continuity constraints instead of d:

$$\frac{\partial^l s}{\partial x^l}(k_i^-) = \frac{\partial^l s}{\partial x^l}(k_i^+) \qquad l \in [\![0, d - r_i]\!]. \tag{2.85}$$

Properties. We now list some basic properties of the B-splines.

Definition domain. From equation (2.82) and equation (2.83), we see that a B-spline can take non-zero values over the interval $]k_{-d}, k_{n+d}[$. Therefore, the interval $[k_{-d}, k_{n+d}]$, named the *full definition domain*, seems to be a good candidate for the definition domain of the spline. This is not always a good choice. Indeed, if there are no coincident knots, the boundary values of the splines will always be zero. Besides, for arbitrary knots, some interesting properties of the B-splines will not be satisfied on $[k_{-d}, k_0[$ and $]k_n, k_{n+d}]$. For instance, it is the case for the property of the partition of unity (see below). Besides, on the interval $[k_0, k_n]$, a B-spline is always the combination of exactly $d+1$ non-zero basis functions (except maybe at the knot position or if some weights are zero). This is not true on $[k_{-d}, k_0[$ and $]k_n, k_{n+d}]$. Consequently, the definition domain $[k_0, k_n]$ is often a more natural choice than $[k_{-d}, k_{n+d}]$. We name this interval the *natural definition domain*. Besides, it is more consistent with the general definition of a spline, as explained in section 2.3.1.

Partition of unity. The B-spline basis functions have the interesting property of forming a partition of unity. This means that we have:

$$\sum_{i=-d}^{n-1} N_{i,d+1}(x) = 1 \qquad \forall x \in [k_0, k_n]. \tag{2.86}$$

This property is the reason why the B-spline basis functions are often qualified of *normalized*. Figure 2.13 illustrates the partition of unity property.

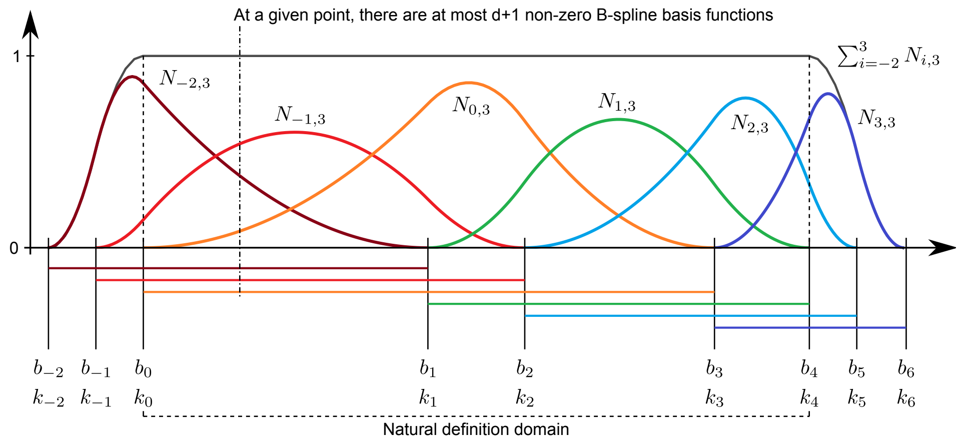


Figure 2.13: Some interesting properties of the B-splines. On the natural definition domain of the B-spline ($[k_0, k_4]$ on this figure), the B-spline basis functions sum up to one (partition of unity). In this example, we use B-splines of degree 2. The horizontal segment below the abscissa axis represents the domain of influence of the B-splines basis function, *i.e.* the interval on which they are not null. At a given point, there are at most $d + 1$ non-zero B-spline basis functions (compact support).

Limited influence of the weights. Another interesting property of the B-splines is the fact that the influence of the weights is spatially limited. This comes from the fact that the B-spline basis functions have a limited support. Consequently, for a given point x, the value $s(x)$ is always the linear combination of at most $d + 1$ non-zero basis functions. This is illustrated on figure 2.13. For reasons that will become clear later in this document, this property allows one to design efficient procedures for parameter estimation with splines. For now, let us just say that it is obviously faster to compute the sum of $d + 1$ numbers than the sum of $d + n$ numbers (especially when n is much more bigger than d, which is often the case in practice).

Representational power. By definition, it is clear that a B-spline is a spline. Reciprocally, it can also be shown that *any* spline of $\mathcal{S}_d(k_0, \dots, k_n)$ can be represented by a B-spline. This property is sometimes considered as the fundamental theorem of the B-splines (de Boor, 2001).

Position of the weights. At this point, the reader who has a minimal background on splines and B-splines might wonder where are the 'control points' of the B-spline. This is generally a confusing point. As we mentioned earlier, the term 'control point' is more conveniently used for vector-valued splines (for instance, a parametric curve embedded in a 2D or 3D space). In this case, talking about the position of the control points has a meaning (and, besides, some interesting properties can be linked to these positions). In the mono-dimensional case, and more generally in the scalar-valued case, speaking about the position of the weights does not make much sense. Indeed, the weight w_i is just a scalar that multiplies the i-th basis function. It is thus linked to the whole interval on which the basis function $N_{i,d+1}$ is non null ($[k_i, k_{i+d}]$). In other words, the weight w_i may be seen as an ordinate without an abscissa. If one really wants to associate a spatial information to the weight w_i, it might be an horizontal segment of line ranging from k_i to k_{i+d} for the abscissa and of ordinate w_i. This is illustrated in figure 2.14.

Linearity. Although the expression $s(x)$ is not linear with respect to the abscissa x, it is linear with respect to the weights. This is of particular interest in parameter estimation since in such problems we are interested in estimating the weights, not the abscissa. A linear relation generally leads to easier computations.

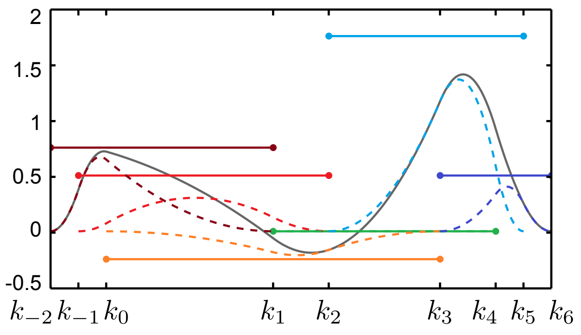


Figure 2.14: A possible graphical representation of the weights of a mono-dimensional B-spline. In this illustration, we take the same knots and the same degree than in figure 2.13 (in other words, the same B-spline basis functions). The horizontal segments represents the weights applied to each one of the basis functions (the heights are equal to the weights). The dashed curves are the weighted basis functions, *i.e.* the basis functions multiplied by their corresponding weight. The solid line is the sum of the weighted basis functions, *i.e.* the final B-spline.

Derivatives. Let us temporarily write explicitly the dependency of s in the weights $\mathbf{w}^\mathsf{T} = \begin{pmatrix} w_{-d} & \dots & w_{n-1} \end{pmatrix}$, *i.e.* $s(x) = s(x; \mathbf{w})$. The derivatives of the B-spline s with respect to x is easily computed:

$$\frac{\partial s}{\partial x}(x; \mathbf{w}) = \sum_{i=-d}^{n-1} w_i \frac{\partial N_{i,d+1}}{\partial x}(x). \tag{2.87}$$

Since the derivative of a B-spline basis function of degree d is a linear combination of B-spline basis functions of degree $d - 1$ (see equation (2.80)), the derivative of a B-spline is a B-spline with one less degree than the original.

The 'derivative', or more precisely the gradient, of s with respect to the weights $\mathbf{w}$ is given by:

$$\boldsymbol{\nabla}^\mathsf{T} s(x; \mathbf{w}) = \begin{pmatrix} N_{-d,d+1}(x) & \dots & N_{n-1,d+1}(x) \end{pmatrix}. \tag{2.88}$$

The gradient of a spline with respect to the parameters $\mathbf{w}$ plays an important role in parameter estimation since it is used by optimization algorithms.

Coincident boundary knots. The *coincident boundary knots* are a common choice for the supplemental boundary knots $k_{-d}, \dots, k_{-1}$ and $k_{n+1}, \dots, k_{n+d}$. It is defined by:

$$\begin{aligned} k_{-d} &= \dots = k_0, \\ \text{and } k_n &= \dots = k_{n+d}. \end{aligned} \tag{2.89}$$

In this case, the full definition domain of the B-spline $[k_{-d}, k_{n+d}]$ coincides with the natural definition domain $[k_0, k_n]$. The boundary values of the splines on the full definition domain are not 0 anymore but 1. Therefore, the partition of unity property is satisfied over the entire definition domain. This particular choice of boundary knots is illustrated in figure 2.15. Note that there exists other choices for the boundary knots. For instance, one may use arbitrary knots at his convenience or *periodic boundary knots* (Dierckx, 1993). This later choice is particularly well suited for approximating periodic functions with B-splines. We do not detail this choice here because we do not use it in this document.

2.3.2.3 Uniform Cubic B-Splines

The *Uniform Cubic B-Splines (UCBS)* are a special case of B-splines which are particularly interesting for their simplicity and efficiency. As the name indicates, they are B-splines of degree 3. This degree is a good

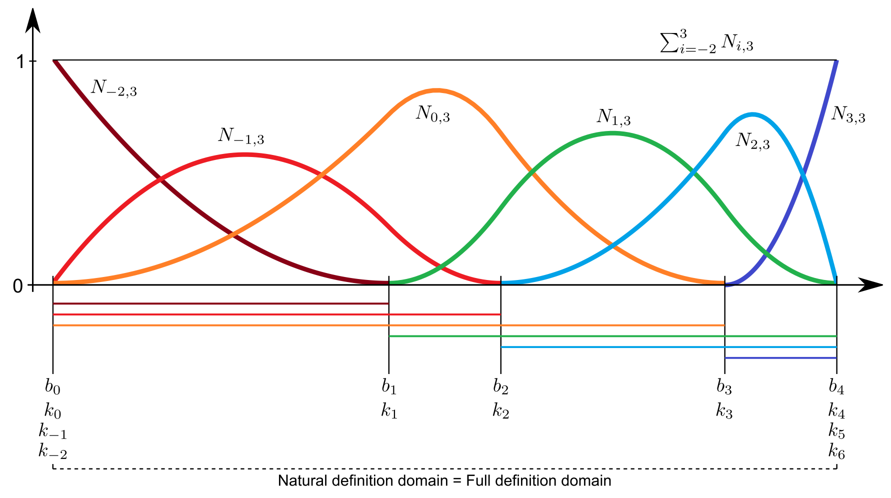


Figure 2.15: B-spline with coincident boundary knots. In this case, the partition of unity property is satisfied over all the full definition domain which is the same as the natural definition domain.

compromise between flexibility and simplicity of the induced computations. The word *uniform* means that all the knots are equally spaced. In this case, the B-splines basis functions are just shifted copies of each others:

$$N_{j,d+1}(x) = N_{i,d+1}\left(x - (j+i)s\right) \qquad \forall (i,j) \in [\![-d, n-1]\!]^2, \tag{2.90}$$

where s is the width of a knot interval (*i.e.* $s = k_{i+1} - k_i$ for all $i \in [\![-d, n-2]\!]$). Figure 2.16 shows a set of B-spline basis functions in the UCBS case. For the sake of simplicity, we drop the index that indicates the order in the notation of the B-spline basis functions, *i.e.* $N_i(x) = N_{i,4}(x)$.

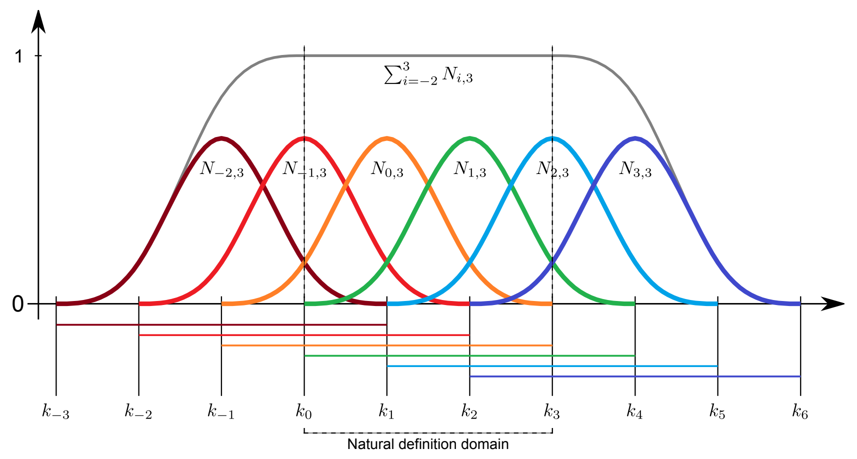


Figure 2.16: The B-spline basis functions for UCBS are shifted copies of each others. On the natural definition domain ($[k_0, k_3]$ on this figure), there are always 4 non-zero basis functions.

Notation, definitions

Normalized abscissa. When dealing with UCBS, it is often more convenient to work with *normalized abscissa* than with original abscissa. The normalized abscissa consists in scaling and translating the abscissa so that the knot k_0 coincides with 0 and so that the width of the knot intervals equals 1. Let $\mathbf{k} = \{k_{-3}, \ldots, k_0, \ldots, k_n, \ldots k_{n+3}\}$ be the knot sequence. The normalized abscissa of the point x, denoted $\nu(x)$, is defined by:

$$\nu(x) = \frac{x - k_0}{s}, \tag{2.91}$$

where s is the original width of a knot interval, *i.e.* $s = k_{i+1} - k_i$ ($i \in [\![-3, n+2]\!]$).

Knot interval. The knot interval in which a point x lies is denoted $\iota(x)$. ι is a function from $[k_{-3}, k_{n+3}]$ to $[\![-3, n+2]\!]$ defined as follows:

$$\iota(x) = \begin{cases} \lfloor \nu(x) \rfloor & \text{if } x \in [k_{-3}, k_{n+3}[\\ n+2 & \text{if } x = k_{n+3}. \end{cases} \tag{2.92}$$

Normalized 0-based abscissa. The normalized 0-based abscissa of x is the number $o(x)$ such that:

$$o(x) = \nu(x) - \iota(x). \tag{2.93}$$

It corresponds to the abscissa of the point x in a local coordinate frame so that the lower bound of the knot interval in which the point x lies coincides with 0 and of width 1.

UCBS basis functions. A closed form expression of the UCBS basis functions can be computed by unrolling the Cox-de Boor recursive relations that defines the B-spline basis functions (equation (2.74)). Without loss of generality, we consider that the knot sequence is normalized and 0-based. If it was not the case, one would just have to replace x by $o(x)$ in the right-hand side of the following equations. The i-th basis function of the UCBS is expressed as:

$$N_i(x) = \begin{cases} b_3(x) = \frac{1}{6}x^3 & \text{if } x \in [k_i, k_{i+1}[\\ b_2(x) = \frac{1}{6}\left(-3x^3 + 3x^2 + 3x + 1\right) & \text{if } x \in [k_{i+1}, k_{i+2}[\\ b_1(x) = \frac{1}{6}\left(3x^3 - 6x^2 + 4\right) & \text{if } x \in [k_{i+2}, k_{i+3}[\\ b_0(x) = \frac{1}{6}\left(-x^3 + 3x^2 - 3x + 1\right) & \text{if } x \in [k_{i+3}, k_{i+4}[\\ 0 & \text{otherwise.} \end{cases} \tag{2.94}$$

Figure 2.17 gives an illustration of the B-spline basis function for the UCBS. The evaluation of a B-spline at a point x is then equivalent to a blending of the four weights closest to the point x. The blending functions are the four polynomials $b_0, \dots, b_3$ of degree 3 that constitutes the B-spline basis function. In other words, we have that:

$$s(x) = \sum_{i=0}^{3} w_{\iota(x)+i} b_i(o(x)) \tag{2.95}$$

Vector notation. As any B-spline, a UCBS can be written in vector notation:

$$s(x) = \mathbf{n}_x^\mathsf{T} \mathbf{w}, \tag{2.96}$$

$$\text{with } \mathbf{n}_x = \begin{pmatrix} \mathbf{0}_{\iota(x)} \\ b_0(o(x)) \\ \vdots \\ b_3(o(x)) \\ \mathbf{0}_{n-\iota(x)-4} \end{pmatrix} \quad \text{and } \mathbf{w} = \begin{pmatrix} w_{-d} \\ \vdots \\ w_{n-1} \end{pmatrix}. \tag{2.97}$$

Remark that the vector $\mathbf{n}_x$ has at most 4 non-zeros entries, whatever the size of the knot vector (*i.e.* the number of weights).

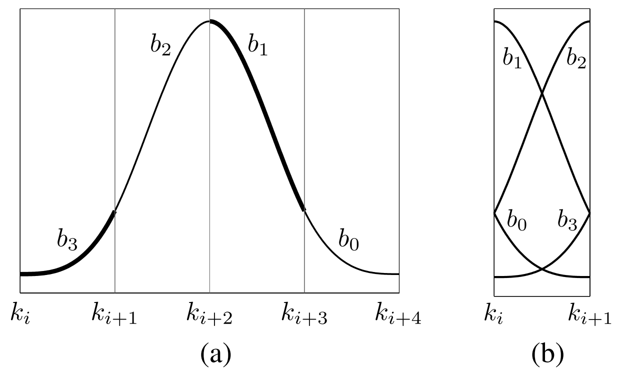


Figure 2.17: Anatomy of a B-spline basis function of UCBS. (a) A B-spline basis function of a UCBS is made of four pieces, each one of which being a polynomial of degree 3. (b) On a given knot interval, the value of a B-spline can be viewed as the blending of the four adjacent coefficients of the B-spline with weights given by the basis functions.

Matrix notation. The *matrix notation* is another common notation for the UCBS. It explicitly reveals that, on a given knot interval, a UCBS is a polynomial function of degree 3 with coefficients obtained by blending the weights of the 4 non-zero B-spline basis functions that corresponds to this knot interval. As we will see later in this manuscript, the matrix notation can make easy some computations that would have been more difficult otherwise. It is given by:

$$s(x) = \begin{pmatrix} o(x)^3 & o(x)^2 & o(x) & 1 \end{pmatrix} \mathsf{M}_G \begin{pmatrix} w_{\iota(x)} \\ w_{\iota(x)+1} \\ w_{\iota(x)+2} \\ w_{\iota(x)+3} \end{pmatrix}, \tag{2.98}$$

where M_G is known as the *geometric matrix* (Malgouyres, 2005) and is defined by:

$$\mathsf{M}_G = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}. \tag{2.99}$$

Although we do not use them in this document, other type of splines can be written in the form of equation (2.98) just by changing the geometric matrix. For instance, this is the case for the cubic Hermite splines (Farin, 1997).

2.3.2.4 Natural Splines

One of the most important application of the splines is to interpolate a sparse set of data point. Let $\{x_i \leftrightarrow y_i\}_{i=1}^n$ be the data set. The *natural spline* is the 'least bended' function that interpolates the data set. It is the solution to the following variational problem:

$$\begin{aligned} \min_{f \in \mathcal{C}^2} \quad & B[f] \\ \text{subject to} \quad & f(x_i) = y_i \quad \forall i \in [\![1, n]\!] \end{aligned} \tag{2.100}$$

where B is the functional that gives the bending energy of a function over its definition domain Ω (here, $\Omega = \left[\min_i x_i, \max_i x_i\right]$). It is defined as:

$$B[f] = \int_\Omega \left(\frac{\partial^2 f}{\partial x^2}(x) \right)^2 \mathrm{d}x \tag{2.101}$$

The solution to problem (2.100) is a B-spline of degree 3 with knots identical to the abscissa of the data points (de Boor, 2001).

2.3.2.5 B-Splines in Higher Dimensions

In this section, we extend the B-splines as presented in the previous section to higher dimensions. Note that for the sake of simplicity, we restrict our study to B-splines although most of the concept presented in this section could be applied to arbitrary splines. We first show how vector-valued B-splines can be defined. We will not spend much time on these aspects since they are barely used in the rest of this document. We then consider a case of greater importance in this thesis: the bivariate B-splines built using the tensor-product of univariate B-splines.

Vector-Valued B-Splines Vector-valued B-splines can easily be constructed from scalar-valued B-splines by replacing the weights with control points. A vector-valued B-splines $\mathcal{S} : \mathbb{R} \rightarrow \mathbb{R}^m$ is thus defined by:

$$\mathcal{S}(\mathbf{x}) = \sum_{i=-d}^{n-1} \mathbf{p}_i N_{i,k+1}(x), \tag{2.102}$$

where the $\mathbf{p}_i$ are the control points of the B-splines. They are vector of $\mathbb{R}^m$. With $m = 2$, equation (2.102) is the equation of a parametric curve embedded in a plane. With $m = 3$, equation (2.102) describes a parametric curve embedded in space. An illustration of vector-valued (with $m = 2$) is given in Figure 2.18.

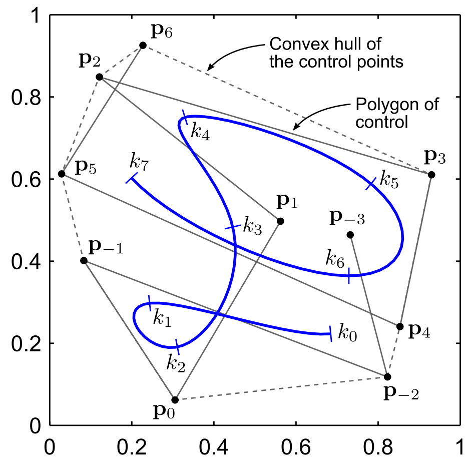


Figure 2.18: A vector-valued B-splines (blue thick curve) embedded in the 2D plane. We use a B-spline of degree 3 with a uniform knot sequence $\{k_{-3}, \ldots, k_{10}\}$. The blue ticks represent the position of the knots (note that we only consider the natural definition domain $[k_0, k_7]$). The black dots are the control points of the curve (from $\mathbf{p}_{-3}$ to $\mathbf{p}_6$). The solid grey line is the polygon of control. The dashed grey line is the convex hull of the control points.

Contrarily to the scalar-valued case, it makes sense to speak about the position of the control points. While a weight w_i could be seen as an ordinate without an abscissa, a control point $\mathbf{p}_i \in \mathbb{R}^m$ completely defines a point in m dimensions. Besides, some properties can be attached to the position of the control points. The set of all the control points $P = \{\mathbf{p}_i\}_{i=-d}^{n-1}$ defines what is called the *polygon of control*. An interesting property is that the B-spline is always included in the convex hull of the polygon of control P.

Bivariate B-Splines One of the most simple approach to extend the univariate B-splines to two variables is known as the tensor product B-spline. The use of the expression *tensor product* will become clear later as we will see that it is related to the tensor product of two matrices (also known as the Kronecker product). We restrict our study of the tensor product splines to the case of two variables but the same principles could be applied for higher dimensions (but with a dramatic increase in the notation burden). Let $\{k_{-d_x}, \dots, k_{n_x+d_x}\}$ and $\{l_{-d_y}, \dots, l_{n_y+d_y}\}$ be two knot sequences. The (scalar-valued bivariate) tensor product B-spline of degree d_x along the x-direction and d_y along the y-direction is the function s from $[k_0, k_{n_x}] \times [l_0, l_{n_y}]$[4] to $\mathbb{R}$ defined as:

$$s(x,y) = \sum_{i=-d_x}^{n_x-1} \sum_{i=-d_y}^{n_y-1} w_{ij} N_{i,d_x+1}(x) N_{j,d_y+1}(y). \tag{2.103}$$

The main advantage of the tensor product approach is that most of the properties of the univariate B-splines are still true in the bivariate case. This is true, for instance, for the partition of unity property, or for the local support of the basis functions.

The definition of equation (2.103) can be interpreted in different ways. We give two of these interpretations in the next two paragraphs.

Linear combination of bivariate basis functions. One can consider the tensor product B-splines as a linear combination of the bivariate basis functions $B_{i,j}$ (for $i \in [\![-d_x, n_x - 1]\!]$ and $j \in [\![-d_y, n_y - 1]\!]$) which are bivariate polynomials of degree d_x in x and d_y in y defined as:

$$B_{i,j}(x,y) = N_{i,d_x+1}(x) N_{j,d_y+1}(y). \tag{2.104}$$

Given the properties of the univariate B-spline basis functions, the basis function $B_{i,j}$ is non-zero only over the interval $[k_i, k_{i+d_x+1}] \times [l_j, l_{j+d_y+1}]$. The construction of the tensor product B-spline basis functions is illustrated in figure 2.19.

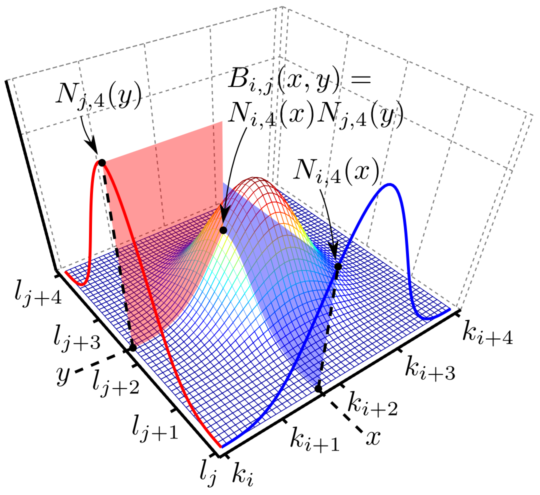


Figure 2.19: Construction of a bivariate B-spline basis function (mesh grid) as the tensor product of two univariate B-spline basis functions (blue and red curves). In this illustration, we took $d_x = d_y = 3$ and uniformly spaced knots. The value of the tensor product B-spline basis function $B_{i,j}$ at the point (x,y) is the product of the univariate basis function N_{i,d_x} and N_{j,d_y} evaluated at the points x and y respectively.

Relation to the tensor product of matrices. On a sub-domain $[k_i, k_{i+1}]$, a univariate spline s of degree d is a single polynomial of degree d. This can be written in vector form as follows:

$$s(x) = \mathbf{a}^\mathsf{T}\mathbf{x} = \begin{pmatrix} a_d & \dots & a_1 & a_0 \end{pmatrix} \begin{pmatrix} x^d & \dots & x & 1 \end{pmatrix}^\mathsf{T}, \tag{2.105}$$

[4]Or $[k_{-d_x}, k_{n_x+d_x}] \times [l_{-d_y}, l_{n_y+d_y}]$ if we consider the full definition domain instead of the natural definition domain.

where the values $\{a_i\}_{i=0}^{d}$ are the coefficients of the polynomial. On a sub-rectangle $[k_i, k_{i+1}] \times [l_j, l_{j+1}]$, a tensor product spline s is given by the tensor (or Kronecker) product of a polynomial of degree d_x in x and d_y in y. Let p_x and p_y be those two polynomials:

$$p_x(x) = \mathbf{a}^\mathsf{T}\mathbf{x} = \begin{pmatrix} a_{d_x} & \dots & a_1 & a_0 \end{pmatrix} \begin{pmatrix} x^{d_x} & \dots & x & 1 \end{pmatrix}^\mathsf{T}, \tag{2.106}$$

$$\text{and } p_y(y) = \mathbf{b}^\mathsf{T}\mathbf{y} = \begin{pmatrix} b_{d_y} & \dots & b_1 & b_0 \end{pmatrix} \begin{pmatrix} y^{d_y} & \dots & y & 1 \end{pmatrix}^\mathsf{T}. \tag{2.107}$$

The fact that a tensor product spline is the tensor product of two univariate polynomial can be seen from the following identity:

$$(\mathbf{a}^\mathsf{T}\mathbf{x}) \otimes (\mathbf{b}^\mathsf{T}\mathbf{y}) = (\mathbf{a}^\mathsf{T} \otimes \mathbf{b}^\mathsf{T})(\mathbf{x} \otimes \mathbf{y}). \tag{2.108}$$

Univariate B-splines with varying weights. Another common way of interpreting the basic definition of equation (2.103) of the tensor product B-spline is to see that each 'slice' parallel to the y-axis of the surface is a univariate spline with weights that are themselves defined by B-splines. It can be seen by rearranging equation (2.103) as follows:

$$s(x, y) = \sum_{j=-d_y}^{n_y-1} v_j N_j(y) \qquad \text{with } v_j = \sum_{i=-d_x}^{n_x-1} w_{ij} N_i(x). \tag{2.109}$$

This point of view is illustrated in figure 2.20.

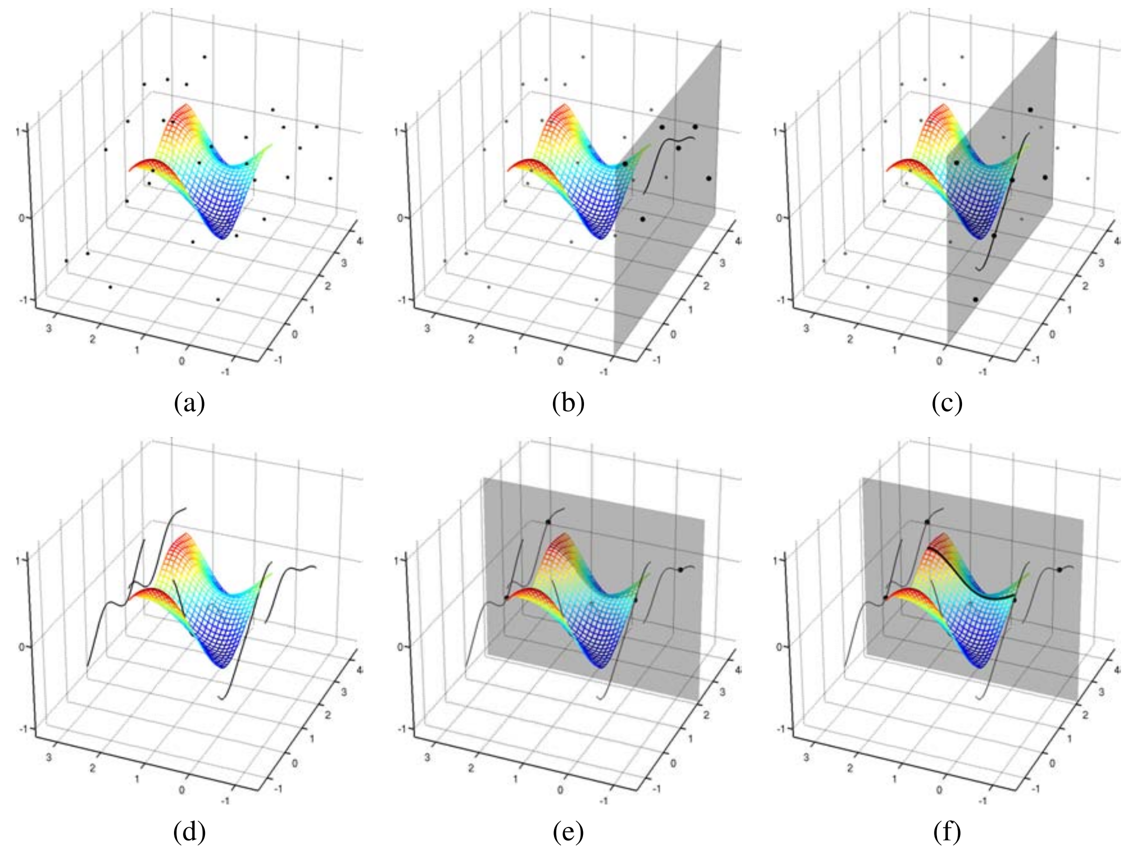


Figure 2.20: The slices of a tensor product B-spline are univariate B-splines with weights on orthogonal B-splines. (a) A tensor product B-splines with its 6×5 weights. (b) The 6 weights in the slice of equation $y = -1$ define a B-spline. (c) All the same way, the 6 weights in the slice of equation $y = 0$ define another B-spline. (d) Applying the same principle again and again, we obtain 5 B-splines. (e) The intersection of this 5 B-splines with the plane of equation $x = \alpha$ gives 5 values. (f) Those values can be used as the weights of a B-spline which is exactly the profile of the surface for $x = \alpha$ (in the example, $\alpha = 2.1$).

Vector-valued bivariate B-splines. The extensions of the B-splines to the vector-valued bivariate case can be combined. In particular, this allows one to define parametric surfaces embedded in a 3D space, which is a function from $\mathbb{R}^2$ to $\mathbb{R}^3$. An example of such surface is presented in figure 2.21. We will make use of this type of surface model for the NURBS-Warps and for the monocular reconstruction of inextensible surfaces.

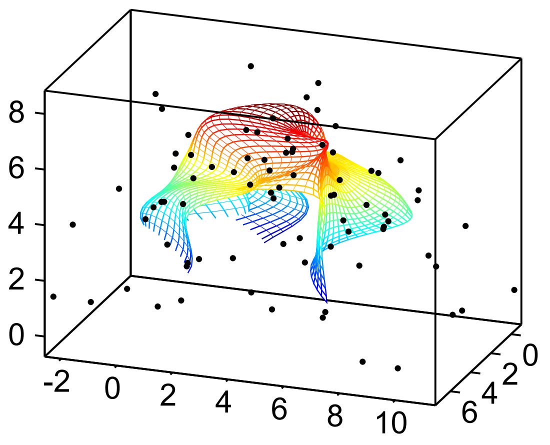


Figure 2.21: Example of a parametric surface described with a 3-vector-valued tensor-product B-spline.

Uniform cubic tensor product B-splines. As in the univariate case, bivariate tensor-product B-splines of degree 3 defined with the help of uniformly distributed knot sequences are of particular interest. Indeed, such B-splines involves only polynomials of degree 3 which make the computations more stable. Besides, these computations are simplified in the sense that the bivariate basis functions are shifted copies of each other, as illustrated in figure 2.22. This type of tensor product B-spline is also abbreviated UCBS.

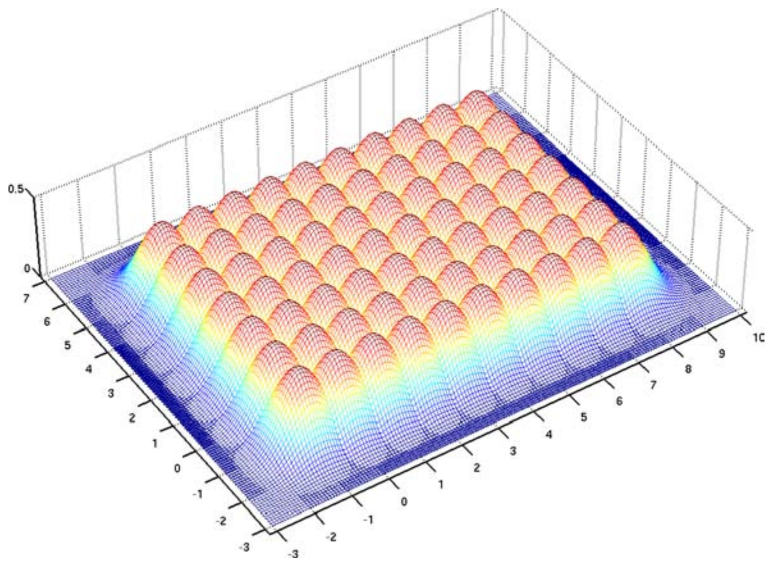


Figure 2.22: The bivariate tensor product B-spline basis functions for UCBS are shifted copies of each others. Here the full definition domain is represented, *i.e.* $[k_{-3}, k_{10}] \times [l_{-3}, l_7]$.

2.3.3 Non Uniform Rational B-Splines (NURBS)

The *rational splines* are another parametric model of function that will be used in this document. As the name may indicate, this model is defined as the ratio of two splines. Rational splines are particularly interesting in computer vision since they may be seen as the perspective projection of standard splines. Although any type of spline could be used, we focus on a particular type of rational splines that relies on B-splines: the *Non-Uniform Rational B-Splines* (NURBS).

2.3.3.1 Basics on NURBS

Definition. We first give the definition of a univariate scalar-valued NURBS. Let $\mathbf{k} = \{k_{-d} \leq \ldots \leq k_{d+n}\}$ be a knot sequence, with d the degree of the NURBS. A NURBS is a function model parametrized by $d + n$ points p_i ($i \in [\![-d, n-1]\!]$)), each one of those associated to a strictly positive[5] weight w_i. Let $r : [k_0, k_n] \to \mathbb{R}$ be the NURBS. Its expression is given by (Carlson, 2009; Farin, 1997):

$$r(x) = \frac{\sum_{i=-d}^{n-1} w_i p_i N_{i,d+1}(x)}{\sum_{i=-d}^{n-1} w_i N_{i,d+1}(x)}. \tag{2.110}$$

The weight w_i controls the influence of the point p_i. In particular, the bigger w_i the closer to p_i the NURBS. This property is illustrated on figure 2.23 for 2-vector-valued NURBS. Another common writing of the NURBS consists in expressing them as a combination of basis functions with the control points as the coefficients:

$$r(x) = \sum_{i=-d}^{n-1} p_i R_{i,d+1}(x), \tag{2.111}$$

where $R_{i,d+1}$ is the i-th *rational basis function* of degree d (order $d+1$):

$$R_{i,d+1}(x) = \frac{w_i N_{i,d+1}(x)}{\sum_{i=-d}^{n-1} w_i N_{i,d+1}(x)}. \tag{2.112}$$

Although the writing of equation (2.111) seems similar to the definition of the B-splines, it is in fact quite different. Indeed, the B-spline basis functions were functions independent of the parameters of the B-spline. On the contrary, the rational basis functions depend on the weights of the NURBS. Consequently, the NURBS are not linear with respect to the parameters. Even if the concepts behind the rational basis functions and the B-spline basis functions are different, they still share some properties. This will be reviewed and illustrated in the paragraph dedicated to the properties of the NURBS.

Higher dimensions. As for the B-splines, the NURBS can be generalized to higher dimensions. An m-vector-valued NURBS is obtained from equation (2.110) by replacing the scalars p_i by vectors $\mathbf{p}_i \in \mathbb{R}^m$. The weights remain unchanged and they still control the influence of the control points $\mathbf{p}_i$. Multi-variate NURBS are derived from equation (2.110) using tensor product B-splines for both the numerator and the denominator. For instance, given two knot sequences $\{k_{-d_x}, \ldots, k_{n_x+d_x}\}$ and $\{l_{-d_y}, \ldots, l_{n_y+d_y}\}$, a 3-vector-valued bivariate NURBS (*i.e.*

[5] Although it is unusual, it could be negative. Either way, it must be different of zero.

a parametric surface embedded in a 3D space) is the function $\mathcal{R}$ from $[k_0, k_{n_x}] \times [l_0, l_{n_y}]$ to $\mathbb{R}^3$ defined by:

$$r(x) = \frac{\displaystyle\sum_{i=-d_x}^{n_x-1} \sum_{j=-d_y}^{n_y-1} w_{ij} \mathbf{p}_{ij} N_{i,d_x+1}(x) N_{j,d_y+1}(y)}{\displaystyle\sum_{i=-d_x}^{n_x-1} \sum_{j=-d_y}^{n_y-1} w_{ij} N_{i,d_x+1}(x) N_{j,d_y+1}(y)}, \tag{2.113}$$

with $w_{ij} \in \mathbb{R}_+^*$ and $\mathbf{p}_{ij} \in \mathbb{R}^3$ for all $(i,j) \in [\![-d_x, n_x - 1]\!] \times [\![-d_y, n_y - 1]\!]$.

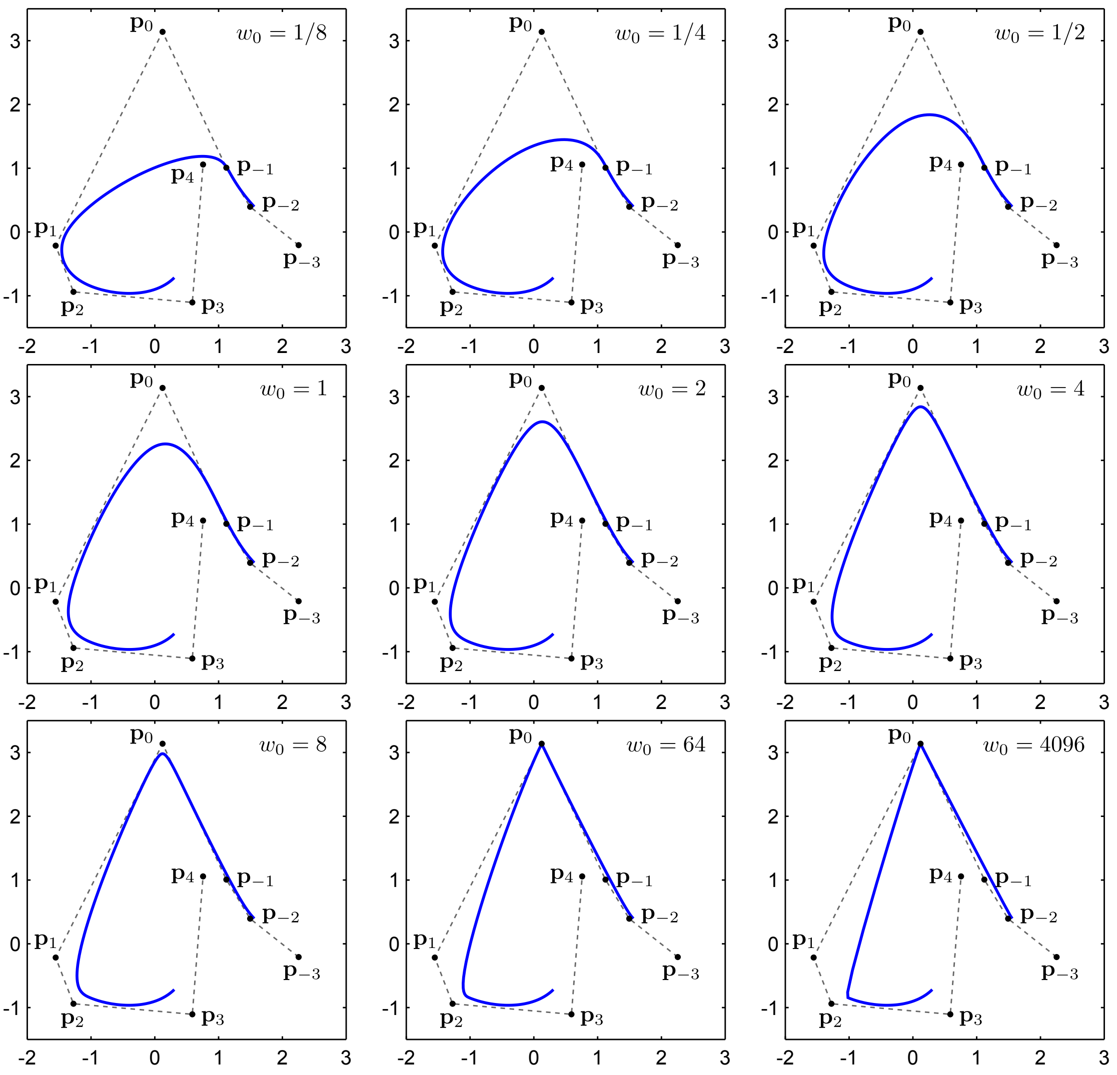


Figure 2.23: Influence of the weights associated to the control points of a 2-vector-valued NURBS.

Perspective projection. Let $\mathcal{R}$ be an m-vector-valued univariate NURBS. One may notice that the vector $\mathcal{R}(t)$ are the Cartesian coordinates of the following point expressed in homogeneous coordinates:

$$\sum_{i=-d}^{n-1} \begin{bmatrix} w_i \mathbf{p}_i \\ w_i \end{bmatrix} N_{i,d+1}(x). \tag{2.114}$$

Equation (2.114) is the exact definition of an $(m+1)$-vector-valued B-spline. In other words, an m-vector-valued NURBS is the perspective projection of an $(m+1)$-vector-valued B-spline. This fact is illustrated in figure 2.24.

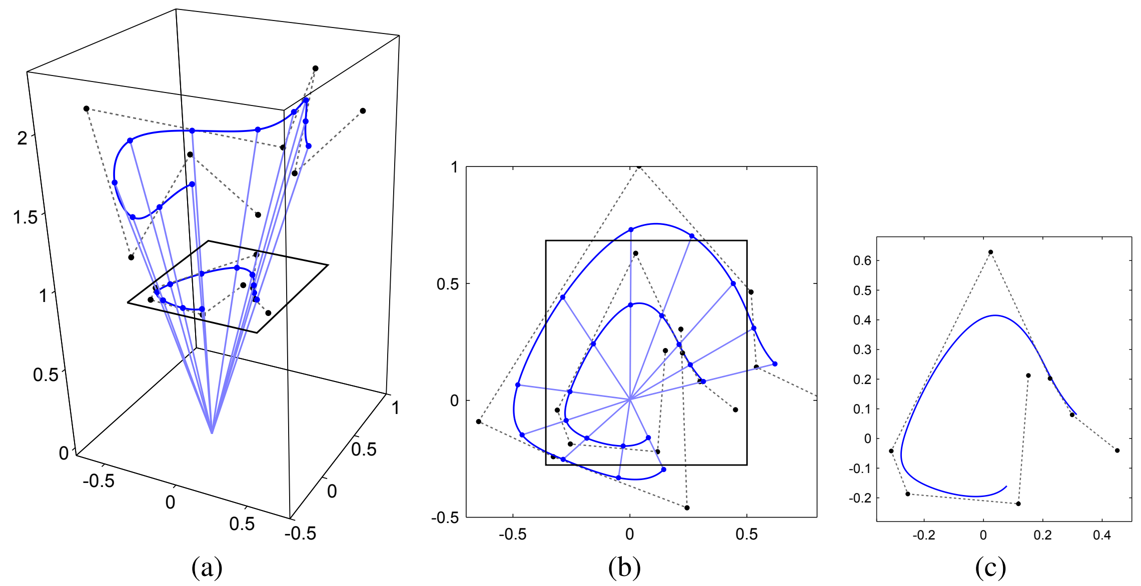


Figure 2.24: An m-vector-valued NURBS is the perspective projection of an $(m+1)$-vector-valued B-spline. In this illustration, we considered that $m = 2$. (a) A 3-vector valued B-spline and its perspective projection into the plane of equation $z = 1$. (b) Same as (a) but viewed from the top. (c) The resulting NURBS.

2.3.3.2 Properties

The NURBS share many properties with the B-splines. In addition, they have supplemental interesting properties. We now give a brief overview of these properties.

Partition of unity. As for the B-splines, the rational basis functions form a partition of unity, *i.e.* :

$$\sum_{i=-d}^{n-1} R_{i,d+1}(x) = 1 \qquad \forall x \in [k_{-d}, k_{n+d}]. \tag{2.115}$$

Note that this property hold true for any set of weights. Figure 2.25 illustrates this property. Note also that for B-splines this property was satisfied only over the natural definition domain. For NURBS, it is satisfied over all the full definition domain. This comes from the fact that the rational basis function are normalized and, therefore, the rightmost (and the leftmost) basis functions does not vanish to zero.

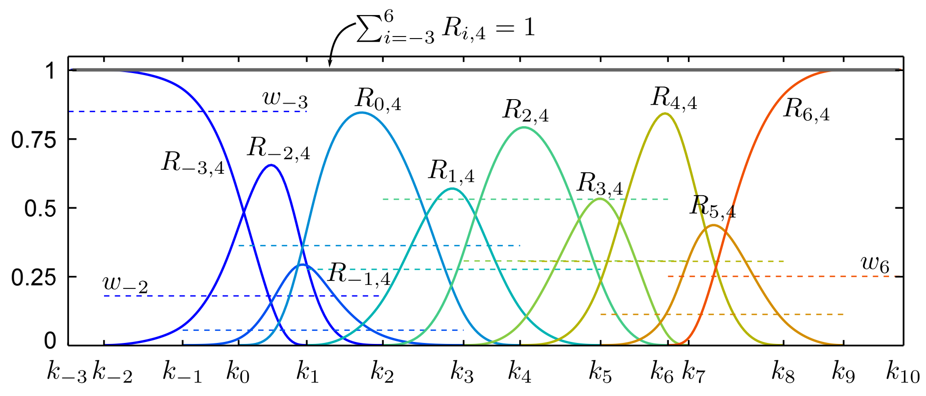


Figure 2.25: The rational basis functions form a partition of unity. In this illustration, we use a NURBS from $\mathbb{R}$ to $\mathbb{R}$ of degree 3 (order 4). The full definition domain $[k_{-3}, k_{10}]$ is shown. Note that contrarily to the B-splines, the rational basis functions still form a partition of unity outside of the natural definition domain $[k_0, k_7]$. This comes from the fact that the rational basis functions are normalized. In this example, the knots and the weights were randomly chosen. For the same reasons than for the control points of the univariate B-splines, the weights of the NURBS are represented using segments (shown in dashed lines) that span the interval of the corresponding rational spline function.

Compact support. The rational basis functions have a compact support:

$$R_{i,d+1}(x) = 0 \qquad \forall x \notin [k_i, k_{i+d+1}]. \tag{2.116}$$

In other word, the i-th control point and the i-th weight influences the shape of the resulting NURBS only over the $d+1$ knot intervals $[k_i, k_{i+d+1}]$.

Continuity. NURBS have continuous derivatives up to order d. In other words, we have that:

$$r \in \mathcal{C}^d([k_{-d}, k_{d+n}]). \tag{2.117}$$

As with the B-splines, it is possible to diminish these continuity conditions at the position of the knots by using multiple knots. This is illustrated in figure 2.26.

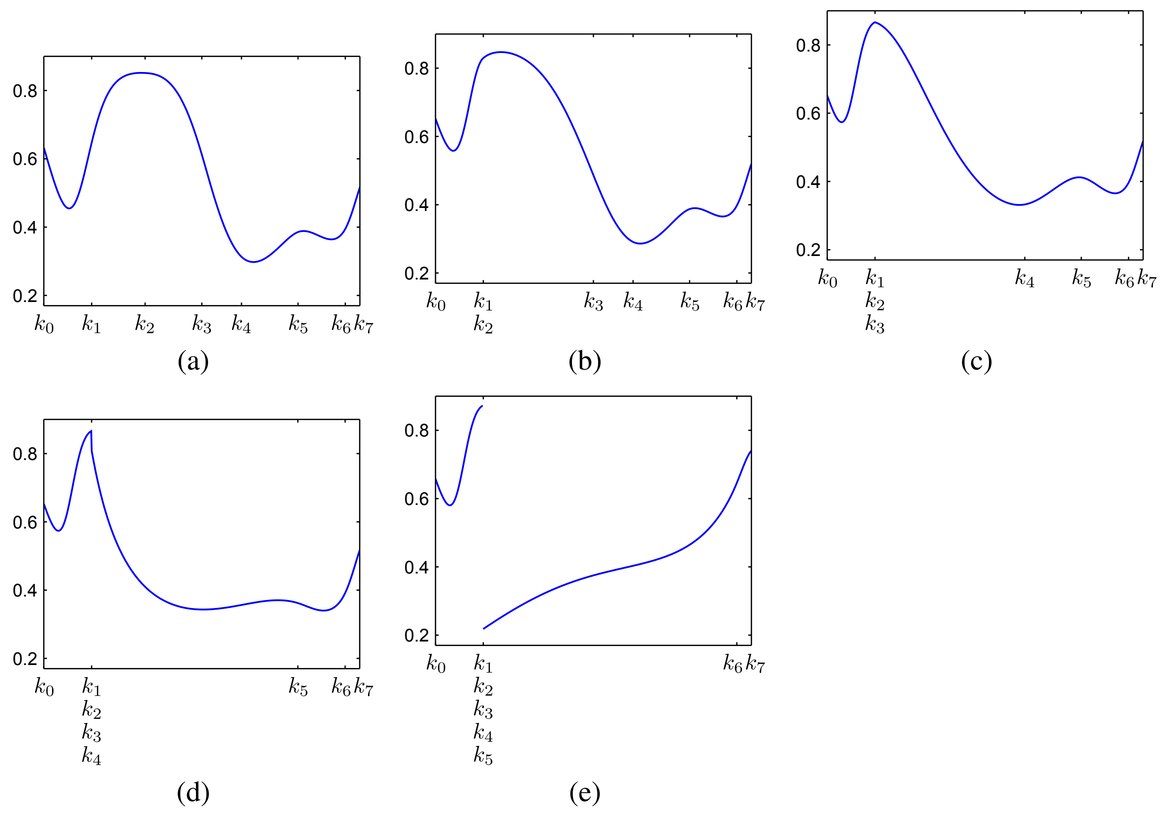


Figure 2.26: Diminishing the continuity conditions by using multiple knots with a NURBS of degree 3. A the position of the knot k_1, the NURBS is $\mathcal{C}^3$ in (a), $\mathcal{C}^2$ in (b), $\mathcal{C}^1$ in (c), $\mathcal{C}^0$ in (d), and not even continuous in (e).

Derivatives. Several 'types' of derivatives are useful for parameter estimation: with respect to the free variable x, to the control points $\mathbf{p} = \begin{pmatrix} p_{-d} & \dots & p_{d+n} \end{pmatrix}^\mathsf{T}$, and to the weights $\mathbf{w} = \begin{pmatrix} w_{-d} & \dots & w_{d+n} \end{pmatrix}^\mathsf{T}$. In this paragraph, we will write explicitly the dependency of r in all its parameters, *i.e.* $r(x; \mathbf{p}, \mathbf{w}) = r(x)$. For the sake of simplicity, let us note n and m the numerator and the denominator of r:

$$n(x; \mathbf{p}, \mathbf{w}) = \sum_{i=-d}^{n-1} p_i w_i N_{i,d+1}(x), \tag{2.118}$$

$$m(x; \mathbf{w}) = \sum_{i=-d}^{n-1} w_i N_{i,d+1}(x). \tag{2.119}$$

We just give the formula of the different derivatives. More detailed computations can be found in (Carlson, 2009).

Derivatives with respect to x.

$$\frac{\partial r}{\partial x}(x; \mathbf{p}, \mathbf{w}) = \frac{n'(x; \mathbf{p}, \mathbf{w}) - m'(x; \mathbf{w}) r(x; \mathbf{p}, \mathbf{w})}{m(x; \mathbf{p}, \mathbf{w})}, \tag{2.120}$$

where n' and m' denote the derivatives of n and m with respect to x.

Derivatives with respect to $\mathbf{p}$. The gradient of r with respect to $\mathbf{p}$ is given by:

$$\nabla_{\mathbf{p}}^{\mathsf{T}} r(x; \mathbf{p}, \mathbf{w}) = \begin{pmatrix} R_{-d,d+1}(x; \mathbf{p}, \mathbf{w}) & \dots & R_{n-1,d+1}(x; \mathbf{p}, \mathbf{w}) \end{pmatrix}, \tag{2.121}$$

where the functions $R_{i,d+1}$ ($i \in [\![-d, n-1]\!]$) are the rational basis functions defined in equation (2.112).

Derivatives with respect to $\mathbf{w}$. The gradient of r with respect to $\mathbf{w}$ is given by:

$$\nabla_{\mathbf{w}}^{\mathsf{T}} r(x; \mathbf{p}, \mathbf{w}) = \begin{pmatrix} \frac{\partial r}{\partial w_{-d}}(x; \mathbf{p}, \mathbf{w}) & \dots & \frac{\partial r}{\partial w_{n-1}}(x; \mathbf{p}, \mathbf{w}) \end{pmatrix}, \tag{2.122}$$

with

$$\frac{\partial r}{\partial w_i}(x; \mathbf{p}, \mathbf{w}) = \frac{m(x; \mathbf{w}) p_i N_{i,d+1}(x) - N_{i,d+1}(x) n(x; \mathbf{p}, \mathbf{w})}{(m(x; \mathbf{w}))^2} \tag{2.123}$$

Representation of conics. Although we will not need this property in this document, it is still worth noticing that conic sections can be represented with NURBS. It is an ability that standard splines, in particular B-splines, do not have. Splines can only approximate conic sections. We will not give the details underlying the representation of conics with NURBS (which can be found in, for instance, (Malgouyres, 2005)). We just give a classic example in figure 2.27: the representation of a circle.

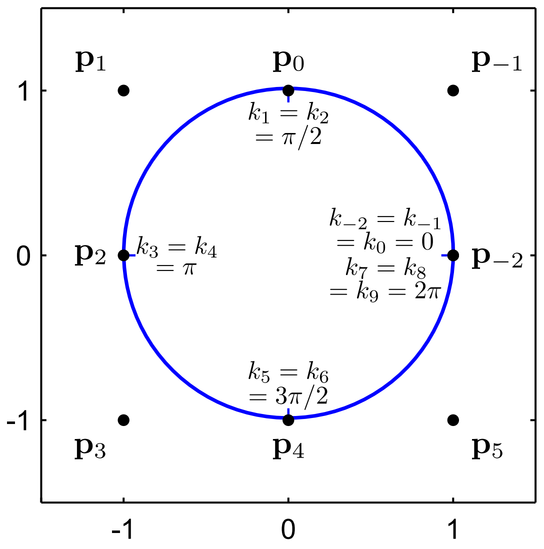


Figure 2.27: Exact representation of a circle with a NURBS of degree 2. Standard splines such as B-splines can only approximate a conic, not represent them exactly.

2.3.4 Radial Basis Functions

2.3.4.1 Generalities

Definition. Radial Basis Functions (RBF) are another model of parametric functions. Let $f : \mathbb{R}^n \to \mathbb{R}$ be an RBF. The function f is of the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{p} w_i \rho(\|\mathbf{x} - \mathbf{c}_i\|) + \sum_{i=p+1}^{p+n+1} w_i \phi_{i-p}(\mathbf{x}), \tag{2.124}$$

The real values w_i ($i \in [\![1, n+1]\!]$) are the weights of the RBF. They can be vectors of $\mathbb{R}^m$ in which case the RBF f is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. The second term in equation (2.124) is the affine part where the functions ϕ_i ($i \in [\![1, q-p]\!]$) are monomials of order up to 1 (Roberts and Stals, 2004). The vectors $\mathbf{c}_i \in \mathbb{R}^n$ ($i \in [\![1, p]\!]$) are called the *centres* of the RBF. They are generally fixed values, *i.e.* they are not part of the parameters to be estimated in a parameter estimation problem. Broadly speaking, the centres define the number of degrees of freedom of the RBF. Finally, ρ is a function from $\mathbb{R}_+$ to $\mathbb{R}$. It is the basis function of the RBF. The norm $\|\bullet\|$ in equation (2.124) is not necessarily the Euclidean norm ; it can be any norm. Common basis functions will be given in section 2.3.4.2. Contrarily to the spline model, the RBF are naturally designed as multi-variate functions. This can be an advantage compared to the splines because the tensor-product approach used with splines to handle multi-variate functions has some limitations.

Note that under some assumptions, the affine part in equation (2.124) can be dropped. The RBF thus becomes:

$$f(\mathbf{x}) = \sum_{i=1}^{p} w_i \rho(\|\mathbf{x} - \mathbf{c}_i\|). \tag{2.125}$$

Linear model. Since an RBF is linear with respect to its weights, it can be written in vector form:

$$f(\mathbf{x}) = \mathbf{l}_{\mathbf{x}}^{\mathsf{T}} \mathbf{w}, \tag{2.126}$$

where $\mathbf{l}_{\mathbf{x}}$ and $\mathbf{w}$ are two vectors of $\mathbb{R}^{p+n+1}$ that are defined in the following way:

$$\mathbf{l}_{\mathbf{x}}^{\mathsf{T}} = \begin{pmatrix} \rho(\|\mathbf{x} - \mathbf{c}_1\|) & \dots & \rho(\|\mathbf{x} - \mathbf{c}_p\|) & \mathbf{x}^{\mathsf{T}} & 1 \end{pmatrix}, \tag{2.127}$$

$$\mathbf{w}^{\mathsf{T}} = \begin{pmatrix} w_1 & \dots & w_{p+n+1} \end{pmatrix}. \tag{2.128}$$

Note that the vector $\mathbf{l}_{\mathbf{x}}$ may depend on $\mathbf{x}$ in a nonlinear fashion. However, this does not change the fact that an RBF is linear with respect to the weights $\mathbf{w}$. Note also that RBF are very similar to splines in the sense that they are both a linear combination of basis functions. The differences are the form of the basis functions and their locations.

The Gram matrix. For reasons that will become clear later in this document, the *Gram matrix* (Dachapak et al., 2005) plays an important role in parameter estimation problems with RBF. Let $\{\mathbf{x}_1, \dots, \mathbf{x}_p\}$ be a set of vectors, the number of which being identical to the number of centres of the RBF. The Gram matrix is the matrix $\mathsf{G} \in \mathbb{R}^{p \times p}$ defined as follows:

$$\mathsf{G} = \begin{pmatrix} \rho(\|\mathbf{x}_1 - \mathbf{c}_1\|) & \dots & \rho(\|\mathbf{x}_1 - \mathbf{c}_p\|) \\ \vdots & & \vdots \\ \rho(\|\mathbf{x}_p - \mathbf{c}_1\|) & \dots & \rho(\|\mathbf{x}_p - \mathbf{c}_p\|) \end{pmatrix}. \tag{2.129}$$

For now, let us just say that parameter estimation problems with RBF involves the inversion of the matrix G. Therefore, the definiteness of the Gram matrix is an important property. The definiteness of the Gram matrix depends on the basis function ρ.

2.3.4.2 Basis Functions

In this section, we give details on classical basis functions for RBF (Powell, 2005). Table 2.2 gives the expression and some basic properties of common basis functions for RBF. These basis functions are illustrated in figure 2.28 in the univariate and the bivariate cases.

Name	$\rho(r)$	Gram matrix
Gaussian	$\exp(-\sigma r^2), r \geq 0, \sigma > 0$	p.d.
Truncated Gaussian	$\begin{cases} \exp(-\sigma r^2) & \text{if } r \leq \alpha \\ 0 & \text{otherwise} \end{cases}$	s.
Inverse multiquadric	$(r^2 + c^2)^{-\frac{1}{2}}, r \geq 0, c > 0$	p.d.
Multiquadric	$(r^2 + c^2)^{\frac{1}{2}}, r \geq 0, c > 0$	p.s.d.
Thin-plate spline	$\begin{cases} \frac{r^2}{\sigma^2} \log \frac{r}{\sigma} & \text{if } r > 0 \\ 0 & \text{otherwise} \end{cases}, \sigma > 0$	p.s.d.
Wendland's function	$\begin{cases} \left(1 - \frac{r}{\sigma}\right)^4 \left(4\frac{r}{\sigma} + 1\right) & \text{if } 0 \leq r \leq \sigma \\ 0 & \text{otherwise} \end{cases}, \sigma > 0$	p.d.

Table 2.2: Some common basis functions (s.: singular, p.d.: positive definite, p.s.d.: positive semidefinite).

Among the basis functions described in table 2.2, two are of particular interest in computer vision: the Thin-Plate Spline (TPS) basis functions and the Wendland's basis functions.

TPS. The TPS are sometimes considered as the natural extension to the bivariate case of the cubic B-splines (Wahba, 1990). This comes from the fact that these two models are the functions that minimizes the following variational minimization problem:

$$\min_{f:\mathbb{R}^n \to \mathbb{R}} B[f], \tag{2.130}$$

with $n = 1$ in the cubic B-spline case and $n \geq 2$ in the TPS case. The functional B is the bending energy. If $n = 2$, it is defined as:

$$B[f] = \iint_{\mathbb{R}^2} \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 \mathrm{d}x\mathrm{d}y \tag{2.131}$$

TPS have been widely used in computer vision (Bartoli, 2008a; Bartoli et al., 2010; Bookstein, 1989; Donato and Belongie, 2002).

Wendland's basis functions. The Wendland's basis functions are another type of basis functions that are particularly interesting for several reasons (Fornefett et al., 2001). First, they are expressed as polynomials of relatively low order, which makes them easier to compute than basis functions relying on 'exotic' functions such as the logarithm or the exponential. Second, the Wendland's basis function have a bounded support. It means that the influence of a weight is spatially limited. This is not the case with, for instance, the TPS.

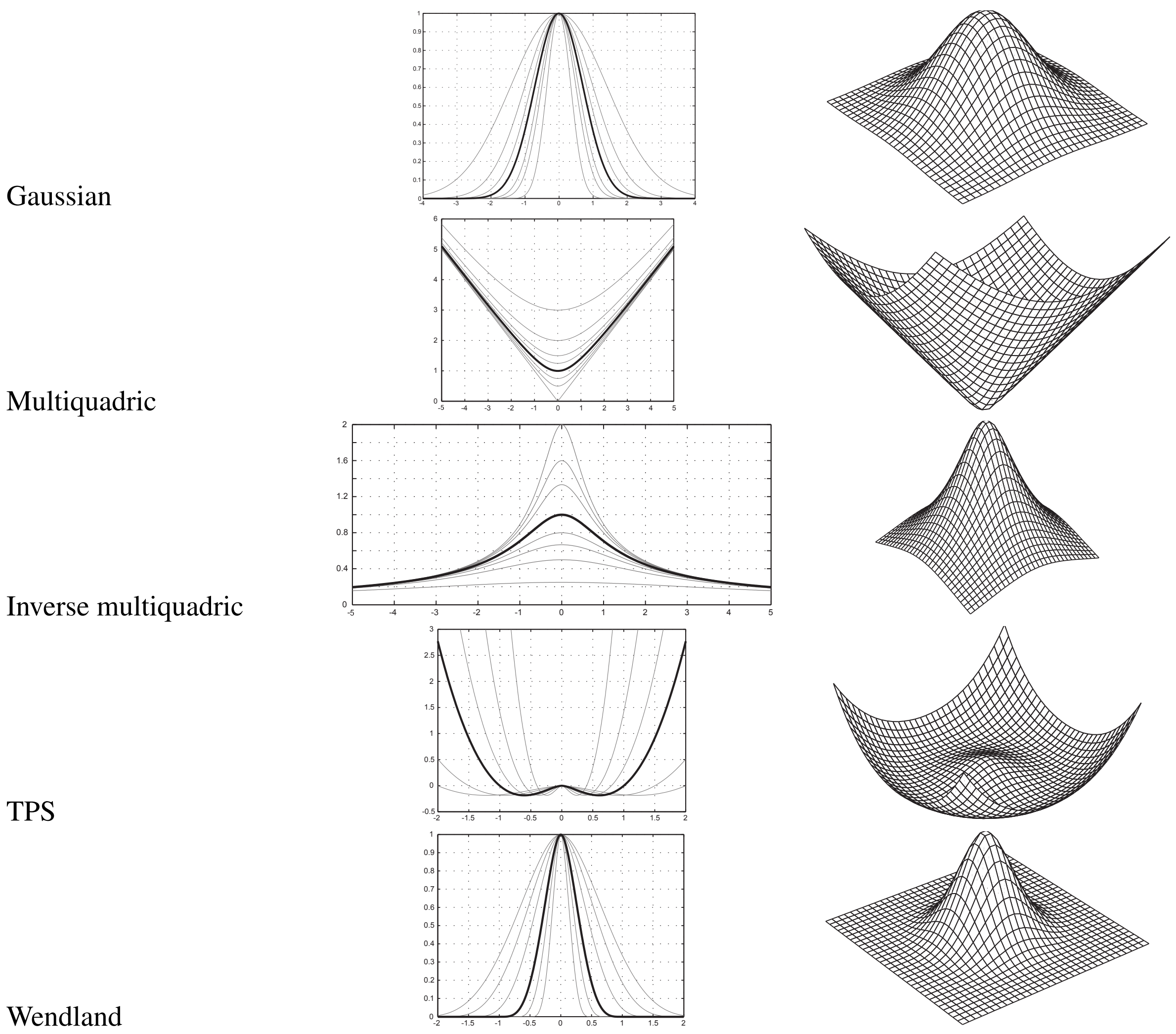


Figure 2.28: Graphical representation of RBF basis functions. Left-most column: univariate case. Right-most column: bivariate case.

Chapter 3

General Points on Parameter and Hyperparameter Estimation

Most of the topics addressed in this thesis can be reduced to parameter and hyperparameter estimation problems. The purpose of this chapter is to give the theory and computational tools for solving such problems. As its name may indicate, a parameter estimation problem is the problem of finding the parameters of a model so that the considered data set is correctly modelled. We first give the theoretical building blocks underlying parameter estimation problems. We then review some of the most common statistical methods (such as Maximum Likelihood Estimation and Maximization A Posteriori) used to solve such problems. In particular, we show how these methods may be formally derived from statistical considerations. Related to the problem of estimating parameters is the problem of estimating hyperparameters. Hyperparameters are additional parameters such as the weighting of different terms in a compound cost function that also influence the result but that, for reasons that will become clear, cannot be estimated the same way as classical parameters. In this chapter, we give the fundamental definitions and concepts related to the hyperparameters. We finally review the standard and general techniques in hyperparameter estimation.

3.1 Parameter Estimation

The vast majority of the problems treated in this document reduces to *parameter estimation problems*. In this section, we give the basic tools and concepts related to such problems. Broadly speaking, the goal of parameter estimation is to estimate (or infer) the unknown parameters of a fixed model in order to fit some noisy measurements. It relies on an analysis of the probability density functions of the error in the measurements.

This section is organized as follows. We first give some general points on parameter estimation. We then give some specializations of the general points.

3.1.1 General Points on Parameter Estimation

3.1.1.1 Parametric Models of Function

The first thing to do in a parameter estimation problem is to choose a parametric model of function. Note that in this document the expression 'parametric model of function' will often be abbreviated to 'parametric model' or even 'model'. A parametric model is a family of functions that can be described with a finite set of parameters. These parameters are generally grouped in a vector $\mathbf{p} \in \mathbb{R}^p$. The parametric model is the following family of functions:

$$\{\mathcal{F}(\bullet; \mathbf{p}) : \mathbb{R}^n \to \mathbb{R}^m \mid \mathbf{p} \in \mathbb{R}^p\}. \tag{3.1}$$

We generally designate a parametric model with the notation $\mathcal{F} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$. A standard convention is to note $\mathcal{F}_\mathbf{p}$ a particular element of the parametric model (for a given set of parameters $\mathbf{p}$). We do not follow this convention in this document. This stems from the fact that we sometimes want to consider the function $\mathcal{F}$ as a function of its natural parameters for a fixed set of parameters $\mathbf{p}$ (*i.e.* the function $\mathcal{F}(\bullet; \mathbf{p}) : \mathbb{R}^n \to \mathbb{R}^m$). Some other times we consider $\mathcal{F}$ as a function of the parameters for a given set of natural parameters $\mathbf{x}$ (*i.e.* the function $\mathcal{F}(\mathbf{x}; \bullet) : \mathbb{R}^p \to \mathbb{R}^m$).

The choice of the model depends mainly on the hypothesis we are able to make on the data to fit. For instance, if one wanted to study the motion of a ball thrown by a basketball player, a good model may be a polynomial of degree 2. The most used models in this thesis have been detailed in section 2.3.

The goal of parameter estimation is to determine a set of parameters $\mathbf{p}$ so that the resulting function $\mathcal{F}(\bullet; \mathbf{p})$ is close to the data measurements.

3.1.1.2 Data Measurements, Errors

From now on and without any loss of generality, we assume that the model is made of scalar-valued functions. The data points, *i.e.* the measurements, can be seen as noisy instances of a reference function $\underline{f} : \mathbb{R}^n \to \mathbb{R}$. Let us note $\{\mathbf{x}_i \leftrightarrow y_i\}_{i=1}^d$ the set of the d measurements with $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$. We have that:

$$y_i = \underline{f}(\mathbf{x}_i) + \varepsilon_i, \quad \forall i \in [\![1, d]\!] \tag{3.2}$$

where ε_i is a random variable. Estimation techniques relies on an analysis of the errors ε_i. In particular, an estimation technique depends on the probability distribution assumed for the random variables ε_i.

3.1.1.3 Probability Density Function (PDF)

Let X be a continuous random variable. The probability density function p of the random variable X is a function from $\mathbb{R}$ to $\mathbb{R}_+$ such that:

$$P[a \leq X \leq b] = \int_a^b p(x)\mathrm{d}x, \tag{3.3}$$

where $P[a \leq X \leq b]$ denotes the probability that X lies in the interval $[a, b]$. In order to be a PDF, a function p must satisfy several criteria. First, it must be a positive function. Second, it must be integrable. Third, it must satisfies:

$$\int_{-\infty}^{+\infty} p(x)\mathrm{d}x = 1. \tag{3.4}$$

The PDF of a random variable defines the distribution of that variable. Some common distributions (and what they imply in terms of parameter estimation) will be detailed later in this section.

The cumulative distribution function F associated to the PDF p is the function from $\mathbb{R}$ to $\mathbb{R}_+$ defined by:

$$F(x) = \int_{-\infty}^{x} p(x)\mathrm{d}x. \tag{3.5}$$

Note that since p is a positive function, F is an increasing function.

3.1.1.4 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a statistical tool used to find the parameters of a model given a set of data. The general idea behind MLE is to find the 'most probable' parameters $\mathbf{p}$ given the observations. Let $\{\mathbf{x}_i \leftrightarrow y_i\}_{i=1}^d$ be the observations. Let e_i be the residual[1] for the i-th data point for a given set of parameters $\mathbf{p}$, *i.e.* $e_i = y_i - f(\mathbf{x}_i; \mathbf{p})$. Note that e_i depends on the parameters $\mathbf{p}$ even though this dependency is not explicitly written. We consider the residuals e_i as random variables. Let us assume that the variables e_i are independent and identically distributed (*i.i.d.*). This allows us to say that the joint probability of the variables $\{e_i\}_{i=1}^d$ given the model parameters $\mathbf{p}$ is:

$$P[e_1, \dots, e_d | \mathbf{p}] = \prod_{i=1}^{d} P[e_i | \mathbf{p}], \tag{3.6}$$

where $P[A|B]$ denotes the probability of the event A given the event B and $P[A_1, \dots, A_n]$ denotes the joint probability of the events $A_1, \dots, A_n$. In terms of probability density function, it means that we have:

$$p(e_1, \dots, e_d; \mathbf{p}) = \prod_{i=1}^{d} p(e_i; \mathbf{p}), \tag{3.7}$$

where p is the probability density function associated to the distribution of the random variables e_i. The likelihood function $\mathcal{L}$ is the joint probability density function considered as a function of the parameters $\mathbf{p}$:

$$\mathcal{L}(\mathbf{p}) = p(e_1, \dots, e_d; \mathbf{p}) = \prod_{i=1}^{d} p(e_i; \mathbf{p}). \tag{3.8}$$

[1]The residuals and the errors must not be confused: the *residuals* are the discrepancies between the the measurements and the fitted model (*i.e.* the predictions) while the *errors* are the discrepancies between the measurements and the actual underlying model (which is generally unknown in practice).

MLE amounts to maximize the likelihood of the parameters $\mathbf{p}$, *i.e.*:

$$\max_{\mathbf{p}} \mathcal{L}(\mathbf{p}). \tag{3.9}$$

From a practical point of view, it is often more convenient to consider the log-likelihood instead of the likelihood. The log-likelihood $\hat{\mathcal{L}}$ is the function defined as:

$$\hat{\mathcal{L}} = \ln(\mathcal{L}). \tag{3.10}$$

In this way, equation (3.9), which is a maximization problem with a cost function defined as a product of numerous factors, can easily be turned into a minimization problem with a cost function defined as a simple sum:

$$\min_{\mathbf{p}} \hat{\mathcal{L}}(\mathbf{p}) \qquad \Leftrightarrow \qquad \min_{\mathbf{p}} \sum_{i=1}^{d} -\ln\big(p(e_i; \mathbf{p})\big). \tag{3.11}$$

MLE will be instantiated to specific distributions later in this section. For now, let us just say that least-squares problems are the natural consequence of MLE with normally-distributed errors.

3.1.1.5 Maximum A Posteriori Estimation

For MLE, we specified a distribution $p(e_i; \mathbf{p})$ for the residuals e_i given the parameters $\mathbf{p}$. For the method of Maximum a Posteriori (MAP), we also specify a *prior distribution* $p(\mathbf{p})$ on the parameters $\mathbf{p}$ that reflects our knowledge about the parameters independently of any observation (Hastie et al., 2001). As an example of parameter prior, one may assume that the fitted function must be 'smooth'. The posterior distribution $p(\mathbf{p}; e_1, \dots, e_d)$ is then computed using the Bayes rule:

$$p(\mathbf{p}; e_1, \dots, e_d) = \frac{p(e_1, \dots, e_d; \mathbf{p}) p(\mathbf{p})}{\int p(e_1, \dots, e_d; \mathbf{p}) p(\mathbf{p}) \mathrm{d}\mathbf{p}}. \tag{3.12}$$

The denominator in the right hand side of equation (3.12) acts as a normalizing constant chosen such that $p(\mathbf{p}; e_1, \dots, e_d)$ is an actual probability density function. The MAP estimation method amounts to solve the following maximization problem:

$$\max_{\mathbf{p}} p(\mathbf{p}; e_1, \dots, e_d). \tag{3.13}$$

Note that MAP is equivalent to MLE if the prior distribution of the parameters $\mathbf{p}$ is chosen to be a uniform distribution. In this case, we say that we have an uninformative prior.

3.1.1.6 Estimator

Definition. An estimator is a function that maps a set of observations to an estimate of the parameters. In other words, an estimator is the result of an estimation process and of a model applied to a set of data. If $\mathbf{p}$ are the parameters we wish to estimate, then the estimator of $\mathbf{p}$ is typically written by adding the 'hat' symbol: $\hat{\mathbf{p}}$. The notion of estimator is independent of any particular model or estimation process. In particular, it may not be formulated as an optimization problem. This is the reason why the generic notation $\hat{\mathbf{p}}$ is used. Note that, by abuse of language, $\hat{\mathbf{p}}$ designates either the estimator itself or the estimated parameters.

Bias. The bias of an estimator is the average error between estimations $\hat{\mathbf{p}}$ and the corresponding true parameters $\underline{\mathbf{p}}$ (for a given set of data). Let $\mathbf{e} \in \mathbb{R}^d$ be the residuals associated to the estimated model, *i.e.*

$e_i = f(\mathbf{x}_i; \hat{\mathbf{p}}) - y_i$. The bias is formally defined as (Hartley and Zisserman, 2003b):

$$E[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}] - \underline{\mathbf{p}} = \int_{\mathbf{e}} p(\mathbf{e}; \underline{\mathbf{p}}) \hat{\mathbf{p}} \, \mathrm{d}\mathbf{e} - \underline{\mathbf{p}}, \tag{3.14}$$

where $E[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}]$ is the expectation of having the estimate $\hat{\mathbf{p}}$ knowing the true parameters $\underline{\mathbf{p}}$ and the errors $\mathbf{e}$. Another way to explain the bias of an estimator is to consider a given set of parameters $\underline{\mathbf{p}}$. Then an experiment is repeated with the same parameters $\underline{\mathbf{p}}$ but with different instance of the noise at each trial. The bias is the difference between the average value of the estimated parameters $\hat{\mathbf{p}}$ and the true parameters $\underline{\mathbf{p}}$.

A desirable property for an estimator is to be *unbiased*. This means that on average with an infinite set of data, there is no difference between the estimated model $\hat{\mathbf{p}}$ and the corresponding true model $\underline{\mathbf{p}}$.

Variance. Another important property of an estimator is its variance. Imagine an experiment repeated many times with the same parameters $\underline{\mathbf{p}}$ but with different instance of the noise at each trial. The variance of the estimator is the variance of the estimated parameters $\hat{\mathbf{p}}$. In case where $\mathbf{p}$ is a vector, we talk about the covariance matrix instead of the variance. It is formally defined as:

$$\mathrm{Var}[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}] = E[(\hat{\mathbf{p}} - E[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}])^2; \underline{\mathbf{p}}, \mathbf{e}]. \tag{3.15}$$

A low variance means that a change in the instance of noise have little effect on the estimated parameters $\hat{\mathbf{p}}$. The variance of an estimator is sometimes called its *efficiency*. For an estimator, it is a desirable property to be efficient, *i.e.* have a low variance. In other words, its preferable that the noise does not affect too much an estimator.

Mean-squared residual. Usually, we are more interested in the Mean-Squared Residual (MSR) of an estimator than in its variance. The MSR measures the average error of the estimated parameters with respect to the true parameters. It is formally defined as:

$$MSR[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}] = E[(\hat{\mathbf{p}} - \underline{\mathbf{p}})^2; \underline{\mathbf{p}}, \mathbf{e}]. \tag{3.16}$$

The variance, the bias and the MSR of an estimator are linked with the following relation:

$$MSR[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}] = \mathrm{Var}[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}] + (E[\hat{\mathbf{p}}; \underline{\mathbf{p}}, \mathbf{e}] - \underline{\mathbf{p}})^2. \tag{3.17}$$

Note that the rightmost term in equation (3.17) is the squared bias of the estimator.

3.1.2 Specific Techniques in Parameter Estimation

In this section we specify some classical estimators that derives from particular assumptions on the distribution of the noise. We will also give the properties of these estimators.

3.1.2.1 Normal Distribution and Least-Squares

Normal distribution. The *normal distribution*, also known as the *Gaussian distribution*, is a continuous probability distribution used to describe a random variable that concentrates around its mean. This distribution is parametrized by two parameters: the mean $\mu \in \mathbb{R}$, and the standard deviation $\sigma \in \mathbb{R}_+^*$. The normal distribution with parameters μ and σ is denoted $\mathcal{N}(\mu, \sigma)$. Table 3.1 gives the main properties of $\mathcal{N}(\mu, \sigma)$. Figure 3.1 illustrates the PDF of the normal distribution.

Parameters	$\mu \in \mathbb{R}$, $\sigma \in \mathbb{R}_+$
Support	$\mathbb{R}$
Mean	μ
Variance	σ^2
PDF	$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

Table 3.1: Some properties of the normal distribution $\mathcal{N}(\mu, \sigma)$

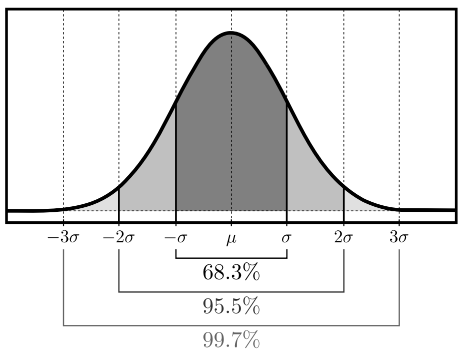


Figure 3.1: Graphical representation of the PDF of the normal distribution $\mathcal{N}(\mu, \sigma)$. This figure shows that a normally-distributed random variable is concentrated around the distribution mean μ. For example, in average, 95.5% of the values taken by a normally-distributed random variable fall within the interval $[-2\sigma, 2\sigma]$. Figure 3.2 illustrates how unlikely it is for a normally-distributed random variable to deviate significantly from the distribution mean.

The multivariate case. The *multivariate normal distribution* is a generalization of the normal distribution to higher dimensions, *i.e.* to random vectors instead of random variables. If we consider random vectors of dimension $k \in \mathbb{N}$, then this distribution is parametrized by two parameters: the mean $\boldsymbol{\mu} \in \mathbb{R}^k$ and the covariance matrix $\Sigma \in \mathbb{R}^{k\times k}$. The multivariate normal distribution of dimension k with parameters $\boldsymbol{\mu}$ and Σ is denoted $\mathcal{N}_k(\boldsymbol{\mu}, \Sigma)$. The covariance matrix Σ is the multidimensional counterpart of the monodimensional variance σ^2. It is a nonnegative-definite matrix. The fact that the matrix Σ is diagonal indicates that the components of the random vector are independent. Having $\Sigma = \mathbf{I}$ means that the components of the random vector are *i.i.d.* Table 3.2 sums up the main properties of the multivariate normal distribution.

Parameters	$\boldsymbol{\mu} \in \mathbb{R}^k$, $\Sigma \in \mathbb{R}^{k\times k}$
Support	$\mathbb{R}^k$
Mean	$\boldsymbol{\mu}$
Variance	Σ
PDF	$f_{\mathcal{N}_k(\boldsymbol{\mu},\Sigma)}(\mathbf{x}) = \frac{1}{\lvert\Sigma\rvert^{\frac{1}{2}}(2\pi)^{\frac{k}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$

Table 3.2: Some properties of the multivariate normal distribution $\mathcal{N}_k(\boldsymbol{\mu}, \Sigma)$.

Link with least-squares estimators. A least square estimator corresponds to the maximum likelihood with data corrupted with a normally-distributed noise. Let $\{\mathbf{x}_i \leftrightarrow \mathbf{y}_i\}_{i=1}^{d}$ be a dataset corrupted with an additive zero-mean *i.i.d.* Gaussian noise, *i.e.* $\mathbf{e}_i \sim \mathcal{N}_m(\mathbf{0}, \Sigma)$ with $\mathbf{e}_i = \mathcal{F}(\mathbf{x}_i; \underline{\mathbf{p}}) - \mathbf{y}_i$, $\mathcal{F} : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^m$ is the parametric model and $\underline{\mathbf{p}}$ are the true parameter of the model. The probability of seeing the data point $\mathbf{x}_i \leftrightarrow \mathbf{y}_i$ given the parameters $\mathbf{p}$ is:

$$p(\mathbf{x}_i \leftrightarrow \mathbf{y}_i; \mathbf{p}) = \frac{1}{(2\pi)^{\frac{m}{2}}\lvert\Sigma\rvert^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathcal{F}(\mathbf{x}_i;\mathbf{p}) - \mathbf{y}_i)^{\mathsf{T}}\Sigma^{-1}(\mathcal{F}(\mathbf{x}_i;\mathbf{p}) - \mathbf{y}_i)\right). \quad (3.18)$$

If we apply the principle of MLE of equation (3.11) to equation (3.18), then we obtain the following minimization problem:

$$\min_{\mathbf{p}} \sum_{i=1}^{d} \|\mathcal{F}(\mathbf{x}_i; \mathbf{p}) - \mathbf{y}_i\|_{\Sigma}^2, \tag{3.19}$$

where $\|\bullet\|_\Sigma$ is the Mahalanobis norm. Equation (3.19) is the very definition of a weighted least-squares minimization problem. Note that if we further assume that the components of the error vectors $\mathbf{e}_i$ are *i.i.d.* or, equivalently, that $\Sigma = \mathbf{I}$, then problem (3.19) reduces to a simple least-squares minimization problem. The tools that allows one to solve these types of problem have been presented in section 2.2.2.

3.1.2.2 Heavy-tailed Distributions and M-estimators

Inliers, outliers, and robustness of an estimator. The data used to estimate the parameters of a model are generally contaminated with noise. It is a common practice to assume that this noise is normally-distributed. A common justification for such an assumption is the *central limit theorem* which states that the sum of a sufficiently large amount of independent random variables is normally-distributed (even if each one of these random variables is not normally-distributed and as long as their mean and variance are defined and finite). However, it is not always reasonable to assume that the data measurements are normally-distributed. In particular, there can be gross errors in the data, which are extremely unlikely if we only consider a normally-distributed noise (see figure 3.2). These 'false' measurements are called *outliers*. Stated otherwise, outliers are arbitrarily large errors. By contrast, measurements that are not outliers are called *inliers*. An estimator is said to be *robust* when these outliers does not affect much the estimation.

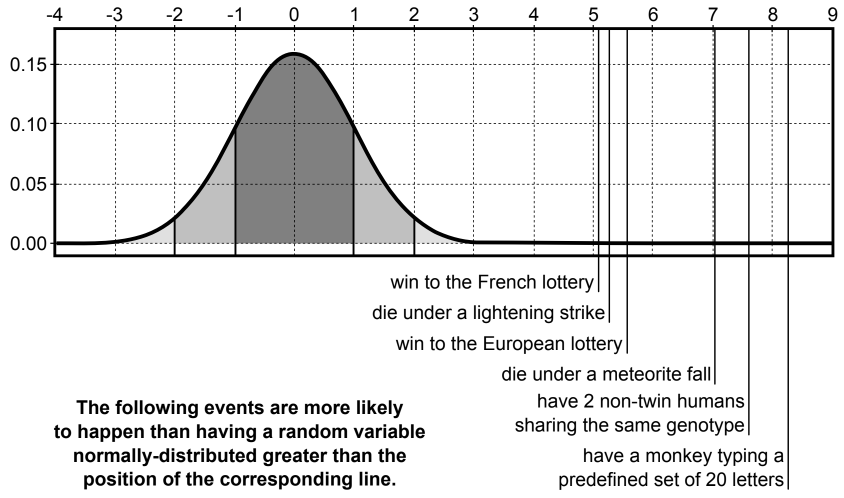


Figure 3.2: Illustration of the probability density function for a centred normal distribution with standard deviation $\sigma = 1$. The probability that a realization of this law lies in $[-\sigma, \sigma]$ is 68.3%, 95.5% in $[-2\sigma, 2\sigma]$, and 99.7% in $[-3\sigma, 3\sigma]$. A value that deviates from the distribution mean more than 5 times the standard deviation is more than extremely unlikely.

M-estimators. M-estimators are robust estimators that are constructed from MLE (or MAP) assuming a non-normally-distributed noise. The principle consists in taking a distribution that has a PDF with 'heavy tails'. Having such tails means that large errors are less improbable than it would be with the normal distribution. Many different M-estimators based upon several noise distributions have been proposed. Note that some of these M-estimators are statistically grounded in the sense that the considered distributions are related to actual facts. Other M-estimators are a little bit more artificial in the sense that their underlying probability distributions have been made up for the need of robustness. Sometimes, the underlying probability distribution of an

M-estimator is not even an actual probability distribution since it does not sums up to 1. In this case, the 'probability distribution' is to be considered as some kind of 'score function' which takes high values for probable inputs and lower values for less probable inputs. This does not forbid one to apply the MLE principle to build a cost function.

We now give some notation and vocabulary on M-estimators. After that, we will detail some typical M-estimators.

Notation and vocabulary on M-estimators. An M-estimator is typically defined using what is called a ρ-*function*. The ρ function of an M-estimator is a function from $\mathbb{R}$ to $\mathbb{R}_+$ that defines the underlying probability distribution[2] of the M-estimator:

$$p(\mathbf{e}) = \exp(-\rho(\mathbf{e})). \tag{3.20}$$

The cost function minimized by the M-estimator is obtained by applying the MLE principle to the distribution of equation (3.20), which gives:

$$\min_{\mathbf{p}} \sum_{i=1}^{d} \rho(\mathbf{e}_i). \tag{3.21}$$

Note that the least-squares cost function is just a particular case of M-estimator with $\rho(x) = x^2$. Two other interesting functions can be associated to an M-estimator: the *influence* function and the *weight* function. The influence function, denoted ψ, is defined as:

$$\psi(x) \overset{\text{def}}{=} \frac{\partial \rho}{\partial x}(x). \tag{3.22}$$

The weight function w, also known as the attenuation function, is defined as:

$$w(x) \overset{\text{def}}{=} \frac{\psi(x)}{x}. \tag{3.23}$$

It measures the weight given to a particular measurement in the cost function according to the error it produces. With M-estimators, the principle is to give large weights to small errors and small weights to large errors (which are likely to come from outliers). Note that, as defined in equation (3.23), the weight function w is not defined at zero. It is generally possible to define it at this particular point with a continuous extension.

Common M-estimators. Here comes a list of some typical M-estimators along with some comments. A graphical illustration of these M-estimators is given in figure 3.3. Other M-estimators can be found in, for instance, (Hartley and Zisserman, 2003a) or (Zhang, 1997).

- **Least-squares.** As it was already mentioned before, the least-squares cost function is a special case of M-estimator with the following ρ-function:

  $$\rho(x) = x^2$$

  Of course, since it relies on a normally-distributed noise, this 'M-estimator' is not robust. In fact, its breakdown point[3] is 0.

[2] For many M-estimators, equation (3.20) does not define an actual probability density function since it does not sum up to one. In this case, we may talk of *pseudo-distribution* but, for the sake of simplicity, we still refer to it as a distribution.

[3] The breakdown point of an M-estimator is a value between 0 and 1 that indicates the proportion of outliers the dataset can contain without altering the accuracy of the M-estimator. Even though the breakdown point may lie in $[0, 1]$, it is not really possible to have a breakdown point greater than 0.5 for obvious reasons.

- **Least absolute values (L_1 norm).** Instead of using a cost function defined as a sum of square, the least absolute values estimator uses a cost function defined as a sum of absolute values. It corresponds to the following ρ-function:
$$\rho(x) = |x|$$
Although more robust than the least-squares cost function, the L_1 norm is not really robust. The main advantage of the L_1 norm is that it is convex (which is a nice property for minimizing it). However, problems may arise with the L_1 norm since it is not differentiable at 0.

- **Huber M-estimator.** The Huber M-estimator is a combination of the least-squares cost function and of the L_1 norm (Huber, 1981). It is defined with:
$$\rho(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| < c_h \\ c_h \left(|x| - \frac{c_h}{2}\right) & \text{otherwise,} \end{cases}$$
with $c_h \in \mathbb{R}_+^*$ a constant that tunes the sensitivity to outliers.

- **Tukey M-estimator.** The ρ-function of the Tukey M-estimator is the bisquare function:
$$\rho(x) = \begin{cases} \frac{c_t^2}{6}\left(1 - \left(1 - \left(\frac{x}{c_t}\right)^2\right)^3\right) & \text{if } |x| < c_t \\ \frac{c_t^2}{6} & \text{otherwise,} \end{cases}$$
with $c_t \in \mathbb{R}_+^*$ a constant that tunes the sensitivity to outliers. The Tukey M-estimator is a *saturated* M-estimator. It means that it takes a constant value after a certain threshold. Some of our contributions will make use of this property. This is one of the most robust M-estimator presented in this document (since the influence of outliers is extremely limited).

- **Cauchy M-estimator.** The Cauchy M-estimator is defined by:
$$\rho(x) = \log\left(1 + \frac{x^2}{\gamma^2}\right),$$
where $\gamma \in \mathbb{R}_+^*$ is the scale parameter. It is derived from the Cauchy distribution whose PDF is a bell-shaped curve as for the normal distribution but with heavier tails. The Cauchy distribution is interesting because some of its properties are directly linked to the notion of outlier. For instance, the mean of the Cauchy distribution does not exists. Intuitively, this stems from the fact that the average of the successive outcomes of a Cauchy-distributed random variable cannot converge since it will always be a value so large that it forbids the convergence (as an outlier would do).

- **Modified Blake-Zisserman.** The Blake-Zisserman M-estimator (Hartley and Zisserman, 2003a) relies on the following assumptions: inliers are considered to be normally-distributed while outliers follow a uniform distribution. It is defined with the following ρ-function:
$$\rho(x) = -\log\left(\exp(-x^2) + \varepsilon\right)$$
This cost function approximates the least-squares cost function for inliers while it asymptotically tends to $-\log(\varepsilon)$ for outliers. Note that this M-estimator was initially presented in a slightly different way in (Blake and Zisserman, 1987).

- **Corrupted Gaussian.** The corrupted Gaussian M-estimator assumes a normal distribution for both the inliers and the outliers. However, a Gaussian with a large standard deviation is used for the outliers. It results in the following ρ-function:

$$\rho(x) = -\log\left(\alpha \exp(-x^2) + (1-\alpha)\frac{1}{w}\exp\left(-\frac{x^2}{w^2}\right)\right),$$

  here w is the ratio of standard deviations of the outliers to the inliers and α is the expected fraction of inliers.

Optimization of M-estimators. Let $f : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}$ be the parametric model and let $\{\mathbf{x}_i \leftrightarrow y_i\}_{i=1}^d$ be the dataset. Let us denote e_i the i-th residual: $e_i(\mathbf{p}) = f(\mathbf{x}_i; \mathbf{p}) - y_i$. The general form of an M-estimator cost function is:

$$\min_{\mathbf{p}} \sum_{i=1}^{d} \rho(e_i(\mathbf{p})). \tag{3.24}$$

Solving this problem can be achieved with a generic optimization algorithm such as the Newton method. However, it is possible to turn problem (3.24) into a weighted least-squares problem with variable weights which can be solved with IRLS. Using IRLS is sometimes preferable than using a generic algorithm such as Newton's method. Turning problem (3.24) into an IRLS problem is achieved by writing the optimality condition (see section 2.2) and developing them in the following manner:

$$\begin{aligned}
\boldsymbol{\nabla}_{\mathbf{p}}\left(\sum_{i=1}^{d}\rho(e_i(\mathbf{p}))\right) = \mathbf{0} \quad &\Leftrightarrow \quad \sum_{i=1}^{d}\psi(e_i(\mathbf{p}))\boldsymbol{\nabla}_{\mathbf{p}}e_i(\mathbf{p}) = \mathbf{0}\\
&\Leftrightarrow \quad \sum_{i=1}^{d}w(e_i(\mathbf{p}))e_i(\mathbf{p})\boldsymbol{\nabla}_{\mathbf{p}}e_i(\mathbf{p}) = \mathbf{0}\\
&\Leftrightarrow \quad \sum_{i=1}^{d}w(e_i(\mathbf{p}))\frac{1}{2}\boldsymbol{\nabla}_{\mathbf{p}}\left(e_i^2(\mathbf{p})\right) = \mathbf{0}.
\end{aligned} \tag{3.25}$$

If we consider the value of $\mathbf{p}$ fixed in the factor $w(e_i(\mathbf{p}))$, then equation (3.25) corresponds to the optimality condition of the following weighted least-squares problem:

$$\min_{\mathbf{p}} \sum_{i=1}^{d} w_i e_i^2(\mathbf{p}), \tag{3.26}$$

with $w_i = w(e_i(\mathbf{p}))$. Therefore, the initial problem (3.24) can be solved with the IRLS principle by iteratively updating the values w_i for $i \in [\![1, d]\!]$.

Note that minimizing an M-estimator cost function is not a trivial task even though an optimization algorithm is available. Indeed, the M-estimator cost functions are generally not convex (except for the L_1 norm and the Huber M-estimator). It is thus easy to fall in a local minimum.

3.1.2.3 Other robust estimators

M-estimators are not the only robust estimators. There exist other methods that cope with the presence of outliers in the data. We will not detail here the methods not used in the rest of this document. For instance, RANSAC (RANdom SAmple Consensus) (Fischler and Robert, 1981) is a popular robust estimators able to fit a parametric model to noisy data containing outliers. The least median of squares (Erickson et al., 2004;

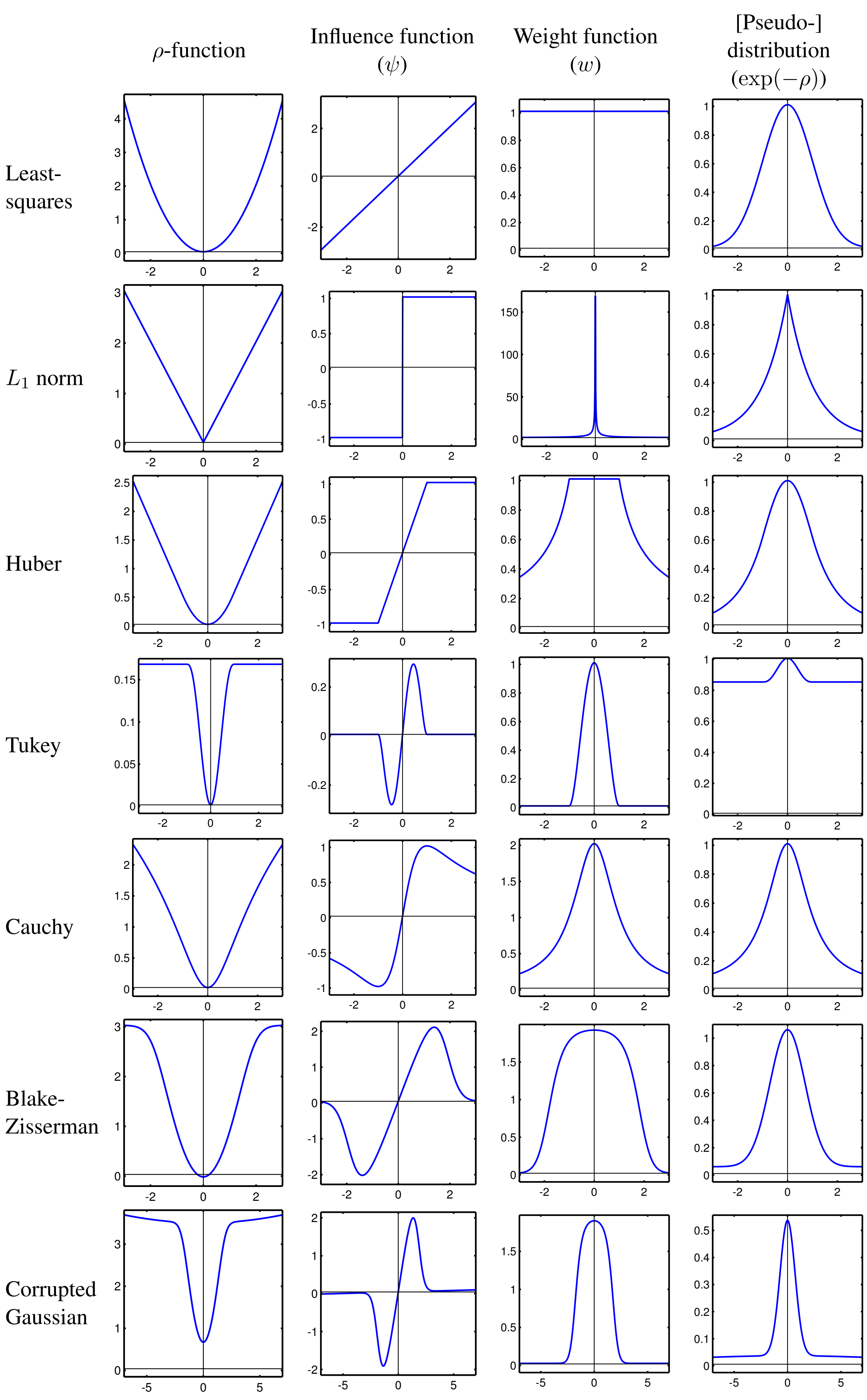


Figure 3.3: Illustration of the different M-estimators presented in section 3.1.2.2.

Rousseeuw, 1984) is another robust estimator which consists in replacing the sum of squares in the classical least-squares approach by the median of the squares:

$$\min_{\mathbf{p}} \operatorname*{med}_{i \in [\![1,d]\!]} (f(\mathbf{x}_i; \mathbf{p}) - y_i)^2. \tag{3.27}$$

This approach is based on the fact that the median is a robust statistic while the mean is not (see the next paragraph). A common technique to solve problem (3.27) is to use a sampling approach. While this is possible and effective when the number of parameters to estimate is limited, it may become intractable for large numbers of parameters.

Mean, median, and trimmed mean. Sometimes, it is interesting to get information on the central tendency of a set of values $\{x_i\}_{i=1}^d$. To do so, using the (arithmetic) mean of the dataset seems to be a natural choice. However, the mean is not a robust statistic. Indeed, a single outlier in the dataset can perturb as much as we want this statistic (the breakdown point of the mean is 0).

Another measure of central tendency is the median. The median is the value that separates the higher half and the lower half of the dataset. Since the outliers are among the highest or the lowest values of the dataset, they do not perturb the median. The breakdown point of the median is $\frac{1}{2}$.

The trimmed mean is another robust central statistic based on the same idea (*i.e.* that outliers are among the highest or the lowest values of the dataset). The trimmed is defined as the mean of the central values of the dataset (*i.e.* the initial dataset with the α lowest and highest values removed). The breakdown point of the trimmed mean is $\frac{2\alpha}{n}$, with n the total number of points.

3.2 Hyperparameters

In this section, we discuss an important aspect of parameter estimation problems: the hyperparameters. This section is organized as follows. We first try to give a general definition of what the hyperparameters are. Since the distinction between classical parameters and hyperparameters is somewhat unclear, the general definition is then complemented with a simple practical example which allows one to get a good grasp at this concept. We finish this section with an explanation of the common approach to determine automatically hyperparameters. The principle will come along with a review of the most common and practical approaches used to determine hyperparameters.

Note that the problems induced by the hyperparameters are one of the central topics tackled in this thesis. Consequently, more advanced aspects on hyperparameters, including our contributions, will be spread across this document. Table 3.3 synthesizes the parts of this document in which hyperparameters are involved.

Where	What
3.2	Definition, description, and common tools.
4.2	The L-Tangent Norm: a contribution that allows one to tune hyperparameters for range surface fitting
4.2.2	Review of another approach to determine hyperparameters: the L-curve criterion
4.3.1.4	Another approach to determine hyperparameters specific to range surface fitting adapted from the Morozov's discrepancy principle
5.3	One of our contribution that uses the specific setup of feature-based image registration to automatically tune hyperparameters

Table 3.3: Parts of this thesis that deal with hyperparameters.

3.2.1 Generalities

As it was explained in the previous sections, the goal of parameter estimation is to find the *natural parameters* of a parametric model. The natural parameters are, to cite a few, the coefficients of a polynomial, the weights of a B-spline or the control points of a NURBS. Generally, in such problems, one also has to determine 'extra-parameters' which are, for instance, a value controlling the configuration of the parametric model or a parameter of a prior distribution in a MAP estimation scheme. This supplemental parameters are called the *hyperparameters*. In other words, an hyperparameter is a parameter of a prior (as distinguished from the parameters of the model for the underlying system under analysis). For example, the degree of a B-spline is an hyperparameter. The number of knots of a B-spline is another example of hyperparameter linked to the parametric model itself. The weight given to a regularization term in a cost function is also an hyperparameter (linked to the cost function instead of the parametric model of function).

3.2.1.1 A Practical Example

In order to make things clearer, we now illustrate the concept of hyperparameter on a simple example. It will allow us to show how some usual types of hyperparameters arise in a parameter estimation problem. Let us consider that we have a set of data points $\{x_i \leftrightarrow y_i\}_{i=1}^{n}$ with $x_i \in \mathbb{R}$ and $y_i \in \mathbb{R}$. These data are illustrated in figure 3.4. We now consider the problem of fitting a polynomial of degree d to this data set. The parametric

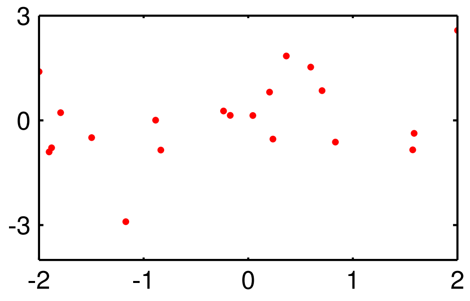


Figure 3.4: Noisy data to fit with a polynomial.

model to use is a function f from $\mathbb{R} \times \mathbb{R}^{d+1}$ to $\mathbb{R}$ that can be written as:

$$f(x; \mathbf{a}) = \sum_{i=0}^{d} a_i x^i. \tag{3.28}$$

Let us further assume that the data points are noisy and that the errors are normally-distributed with zero mean and standard deviation σ. If we note $\underline{f}$ the true function, then we have that:

$$y_i = \underline{f}(x_i) + \varepsilon_i, \tag{3.29}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma)$ for all $i \in [\![1, n]\!]$. One may also want to make an assumption on the fitted polynomial. For instance, smooth polynomial could be preferred. This kind of assumption is quite usual since it helps in compensating the undesirable effects caused by for instance, noise, outliers, or lack of data. In the MAP estimation framework, this can be achieved using a prior distribution on the parameters. For example, we could say that smooth polynomials (*i.e.* the ones that minimizes the bending energy) must be more probable than other ones. This can be implemented using the following prior distribution on the parameters:

$$P(\mathbf{a}) \propto \exp(-\lambda b(\mathbf{a})), \tag{3.30}$$

where b is the function from $\mathbb{R}^{d+1}$ to $\mathbb{R}_+$ that gives the pseudo-bending energy of the polynomial:

$$b(\mathbf{a}) = \int_{\Omega} (f(x; \mathbf{a}))^2 \, \mathrm{d}x, \tag{3.31}$$

where $\Omega \subset \mathbb{R}$ is the definition domain of the polynomial f with respect to the free variable x. The PDF of the prior distribution defined in figure 3.30 is a bell-shaped curve which width is controlled by the parameter $\lambda \in \mathbb{R}_+^*$. A large λ results in a narrow bell-shaped curved which means that only very smooth polynomials will have a non-negligible probability. On the contrary, a small lambda produces a large bell-shaped curve which makes probable rugged polynomials. The prior parameter λ is often named the *smoothing parameter* or *regularization parameter*. If we apply the MAP principles (as described in section 3.1.1.5) with the hypothesis we have just made, then estimating the coefficients $\mathbf{a}$ of the polynomial is equivalent to solve the following minimization problem:

$$\min_{\mathbf{a} \in \mathbb{R}^{d+1}} \sum_{i=1}^{d} (f(x_i; \mathbf{a}) - y_i)^2 + \lambda b(\mathbf{a}). \tag{3.32}$$

In equation (3.32), the coefficients of the vector $\mathbf{a}$ are the natural parameters of the estimation problem while the values d and λ are the hyperparameters. More precisely, the degree d is a *model hyperparameter* since it is a parameter that 'tune' the underlying model of function. The value λ is what we call a *cost hyperparameter* since it is linked to the cost function itself. If we consider that d and λ are given, then problem (3.32) is a simple linear least-squares problem. However, determining correct values for the hyperparameters is a difficult task, especially if we want this determination to be automatic. It is nonetheless mandatory in order to get proper results. Figure 3.5 shows the influence of the hyperparameters d and λ of our example with the data of figure 3.4.

A fact that makes difficult the automatic computation of the hyperparameters is that they cannot be directly included in the initial optimization problem. For instance, in the context of our example, the following minimization problem will not lead to the expected results:

$$\min_{\substack{\mathbf{a} \in \mathbb{R}^{d+1} \\ d \in \mathbb{N} \\ \lambda \in \mathbb{R}_+}} \sum_{i=1}^{d} (f(x_i; \mathbf{a}) - y_i)^2 + \lambda b(\mathbf{a}). \tag{3.33}$$

For instance, the bending term b is a positive value. Consequently, the best way to minimize the influence of this term in the cost function of equation (3.33) is to set $\lambda = 0$. This is clearly not the expected result since, for some reasons, we wanted a 'smoothed' polynomial. The same kind of dubious effect will arise for the degree of the polynomial. Indeed, the error in equation (3.33) will be completely minimized (*i.e.* the value of the cost function is null) if the fitted polynomial interpolates the data points. This is achieved by taking $d = n - 1$ where n is the number of data points. In our example, we had 25 data points but the best fit is realized with a polynomial of degree 10. The general principles used to build a correct approach for automatic hyperparameter selection will be explained in section 3.2.2.

3.2.1.2 Formal Definitions

We give here a set of definitions which are more or less[4] related to the hyperparameters.

[4]Some of these definitions are completely specific to the hyperparameters while some others are more generic but conveniently explained in this section.

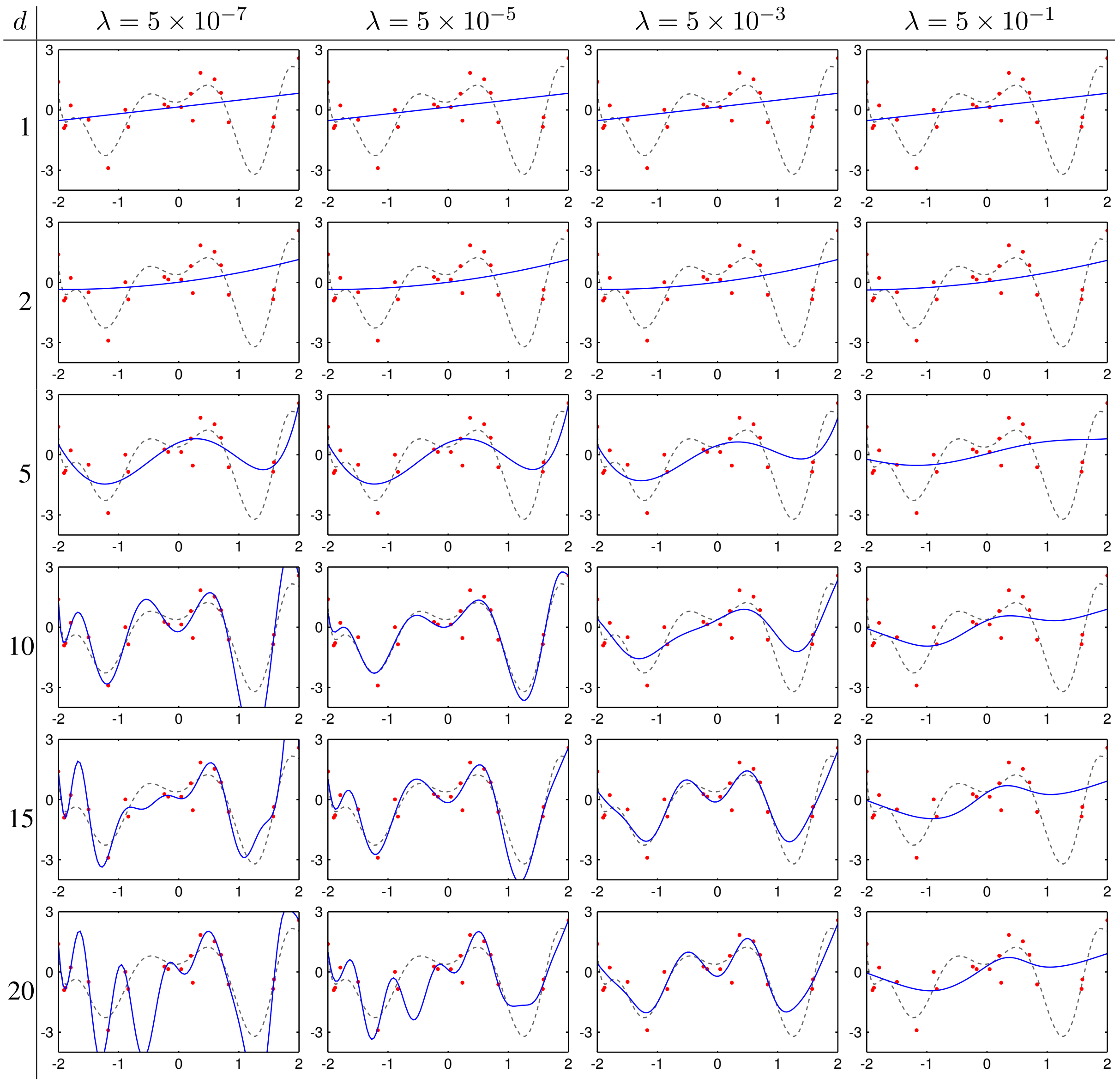


Figure 3.5: Polynomials fitted on the data of figure 3.4 with different hyperparameters. The columns correspond to different regularization parameters λ. The row correspond to different polynomial degree d. The red dots are the data points. The grey dashed curve is the true function (which is, of course, considered as unknown in the parameter estimation problem. The blue solid line is the fitted polynomial. This figure shows how important is the choice of the hyperparameters. Here, the best fit seems to correspond to $d = 10$ and $\lambda = 5 \times 10^{-5}$. In this artificial setup, it is easy to say that this particular couple of hyperparameters is the good choice since the ground truth function is available. In practice, choosing correct hyperparameter is far from being a trivial task.

Hyperparameter classification. In this thesis, we classify the hyperparameters into two main categories. First, we consider the parameters linked to the prior distribution given to the natural parameter to estimate. We call this type of hyperparameters the *cost hyperparameters* since, after having derived an estimation procedure with, for instance with MAP, they appear as tuning values of the cost function. Second, we consider the *model hyperparameters* which are directly linked to the parametric model of function. Examples of this second type of hyperparameter include the degree of a polynomial or of a B-spline, the number of control points of a B-spline, the kernel bandwidth of a RBF, the number of centres of a RBF, etc.

Model complexity. Roughly speaking, the (effective) complexity of a parametric model corresponds to its number of degrees of freedom minus the 'number'[5] of constraints implicitly induced by the prior given to the parameters of the model. Consequently, the complexity of a parametric model is mostly tuned by the hyperparameters. In order to get a correct parametric estimator, the complexity of the model must meet the actual complexity carried by the data. By 'actual', we mean the complexity of the data excluding the noise. In practice, this actual complexity is generally not known precisely. Indeed, given a set of data, it is hard to distinguish between the noise and the true information. This fact systematically leads to problems such as underfitting and overfitting which are closely related to the concepts of bias and variance of an estimator. These concepts are explained in the next paragraphs along with some other definitions.

Residual error. The residual error is the sum of the errors between the estimated parametric model and the data (at the location of the data points). For a parametric model f and a set of parameters $\mathbf{p}$, the residual error may be defined as:

$$\sum_{i=1}^{n} d(f(x_i; \mathbf{p}), y_i), \tag{3.34}$$

where d is a function that gives the distance between its two arguments (for instance, a common choice is the squared Euclidean norm). Note that equation (3.34) is similar to what is generally minimized in a parameter estimation problem. However, as it will be seen later, having a minimal residual error does not always lead to the proper results.

Prediction error. The prediction error (also known as the *expected prediction error* or *generalization error* or *test error*) is a quantity that measures how well an estimated model is able to generalize (Hastie et al., 2001). Given a point x_0 different of the points used to estimate the parameters, the prediction error measure the difference between the estimated value $f(x_0; \mathbf{p})$ and the true value $\underline{f}(x_0)$. In other words, it measures how well the estimated model performs, *i.e.* how close it is to the true function. The prediction error at x_0 is given by:

$$E\left[\left(\underline{f}(x_0) - f(x_0; \mathbf{p})\right)^2\right] \tag{3.35}$$

The prediction error can be broken down into two terms:

$$\left(E\left[f(x_0; \mathbf{p})\right] - \underline{f}(x_0)\right)^2 + \mathrm{Var}(f(x_0; \mathbf{p})). \tag{3.36}$$

The first term is the squared bias of the estimator while the second term is the variance of the estimator.

Bias-variance compromise. Equation (3.36) should ideally be the cost function to be minimized for parameter estimation but this is not possible in practice because $\underline{f}$ is not known. A parametric model with low

[5] 'Number' is probably not the right term since it includes, for instance, the strength of a regularization term.

complexity leads to a high bias and low variance. On the contrary, a parametric model with high complexity leads to a low bias and high variance. This is called the *bias-variance compromise*. Note that minimizing usual cost functions, which are particular forms of Equation (3.34), is equivalent to minimize only the bias term in equation (3.36). This is the reason why the hyperparameters cannot be directly included in the standard optimization problem for parameter estimation since it would only reduce the bias of the estimator, not its variance.

Underfitting, overfitting. The phenomena of underfitting and overfitting are closely related to the bias-variance compromise. The underfitting problem arises when the complexity of the parametric model is not enough high. In other words, the parametric model is not flexible enough to model the data. It corresponds to an estimator with high bias and low variance. On the other side, there is the overfitting problem. It corresponds to a parametric model with a too high level of complexity. In this case, it is so flexible that the estimated parametric model will model not only the data but also the noise. It is an estimator with high variance and low bias.

Approximation, interpolation. An estimated parametric model f is said to *interpolate* the data $\{x_i \leftrightarrow y_i\}_{i=1}^{n}$ if we have that:

$$f(x_i; \mathbf{p}) = y_i \qquad \forall i \in [\![1, n]\!]. \tag{3.37}$$

It corresponds to the estimator with minimal bias. If the data point are noisy (as it is always the case in the real world), it also corresponds to the estimator with maximal variance. Consequently, *interpolators* are generally not the best estimators (at least, when there is noise in the data). It is thus preferable to have estimators that *approximates* the data instead of *interpolating* them, *i.e.* such that:

$$f(x_i; \mathbf{p}) \approx y_i \qquad \forall i \in [\![1, n]\!]. \tag{3.38}$$

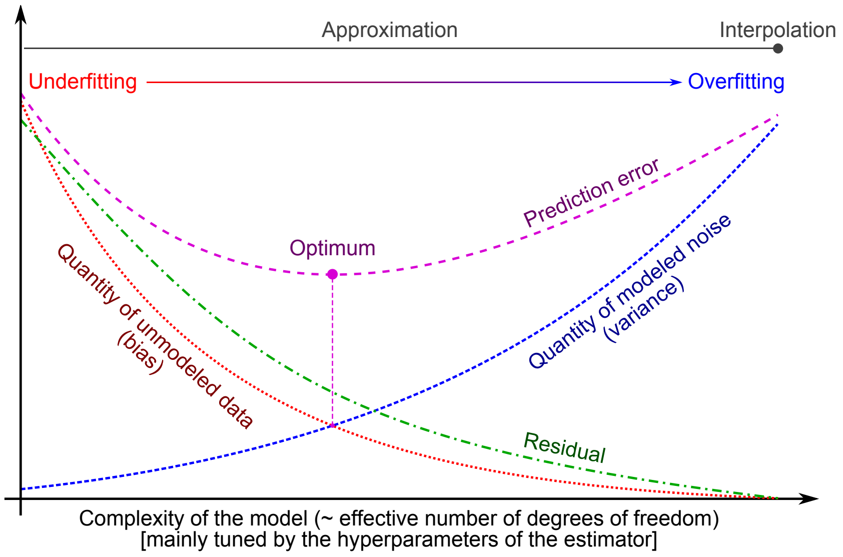

Approximation
Interpolation
Underfitting
Overfitting
Prediction error
Optimum
Quantity of unmodeled data (bias)
Quantity of modeled noise (variance)
Residual
Complexity of the model (~ effective number of degrees of freedom) [mainly tuned by the hyperparameters of the estimator]

Figure 3.6: Illustration of the concepts presented in section 3.2.1.2. The hyperparameters in an estimation scheme (both cost and model hyperparameters) allows one to tune the complexity of a parametric model. The best hyperparameters for an estimator are the one that would result in a minimization of the prediction error (purple line). The prediction error cannot be computed in practice since its definition relies on the knowledge of the true underlying function. Instead, one generally minimizes a quantity related to the residual error (green line). This quantity is closely related to the bias of the estimator. Minimizing the bias of an estimator is not sufficient enough since it may lead to a fitted function that model the noise.

3.2.2 Automatic Computation of the Hyperparameters

3.2.2.1 General Principle

As we just explained, the hyperparameters are important in order to estimate proper parameters. We also said that determining correct hyperparameters was not a simple problem in the sense that it cannot be directly included in the minimization problem derived to estimate the natural parameters. It is nonetheless possible to use automatic procedure to determine the hyperparameters. In fact, there exists an incredibly large number of such procedures. For that matter, some of the contributions we propose in this document are new ways of automatically determining hyperparameters.

The most general approach to automatically determine hyperparameters (in other words, to select the model complexity) consists in minimizing an external criterion which, in a nutshell, assesses the quality of the estimated model. Let us call c this criterion. In the general case, c is a function from $\mathbb{R}^h$ to $\mathbb{R}$ where h is the number of hyperparameters (which are grouped in a vector denoted $\mathbf{h} \in \mathbb{R}^h$ in this section). Always in the general case, $c(\mathbf{h})$ is small when the function induced by the hyperparameters $\mathbf{h}$ (and the subsequent parameters) is correct, *i.e.* it does not underfit nor overfit the data. Let r be the cost function used to determine the natural parameters of a model from the data. For the needs of this section, we consider r as a function of both the natural parameters and the hyperparameters. In particular, r is a function from $\mathbb{R}^n \times \mathbb{R}^h$ where n is the number of natural parameters and h is the number of hyperparameters. The general principle of parameter estimation with automatic hyperparameters selection is to solve the following nested optimization problem:

$$\min_{\mathbf{p} \in \mathbb{R}^n} r \left(\mathbf{p}; \arg \min_{\mathbf{h} \in \mathbb{R}^h} c(\mathbf{h}) \right). \tag{3.39}$$

Note that the problem stated in equation (3.39) is quite different of the inconsistent problem in which both the parameters and the hyperparameters are mixed in the initial optimization problem:

$$\min_{\substack{\mathbf{p} \in \mathbb{R}^n \\ \mathbf{h} \in \mathbb{R}^h}} r(\mathbf{p}; \mathbf{h}). \tag{3.40}$$

Indeed, for the reasons explained above, the solution of problem (3.40) would be the one with minimal bias and, consequently, it would completely overfit the data (if possible, it will even try to interpolate the data instead of finding an appropriate approximation).

Given what was just said, the main difficulty for building an automatic hyperparameter selection procedure is obviously to find a criterion c which reach a minimum value for correct hyperparameters[6]. Of course, the best choice for c would be the prediction error but, unfortunately, it is not possible since this criterion relies on an unknown: the true function. Many criteria have been proposed. They take their root in different domain: information theory, Bayesian reasoning, philosophy, etc. We review some of the most classical ones in the next sections.

3.2.2.2 Cross-Validation

Cross-Validation is one of the most common approach to automatically tune hyperparameters. At a glance, the principle of Cross-Validation is to build a criterion that, for a given set of hyperparameters, measures how well the resulting estimated model is able to generalize the data. In other words, it measures how good the behaviour of the estimated model is 'between' the data. Of course, the ability to generalize of an estimated model can be

[6]Practical considerations such as non-convexity of the criterion make this problem even more difficult even though such properties are really desirable for automatic optimization.

computed precisely only if the true function is available. The magic of Cross-Validation is to be able to estimate this ability only from the data and without requiring the true model.

The general principle of Cross-Validation relies on an approach commonly used in learning theory: dividing the dataset into several subsets. Each one of these subsets is then alternatively used as a training set or as a test set to build Cross-Validation score function. The way the dataset is divided gives rise to different variant of Cross-Validation. In some special cases, efficient approximations can be derived which make computations faster. In the next paragraphs, we review some of the most common flavour of Cross-Validation.

Cross-validation is very interesting for several reasons (Arlot and Celisse, 2010). In particular, it is 'universal' in the sense that the data splitting technique is very generic. The only assumption to be made for using cross-validation is that the data must be identically distributed. Besides, it is not linked to a particular framework as most as other methods for hyperparameters selection are. For instance, Mallow's C_P is only applicable for least-squares cost functions (see section 3.2.2.3).

Leave-one-out Cross-Validation. Leave-one-out Cross-Validation (LOOCV) is one of the simplest variant of the general Cross-Validation principle. A detailed study of LOOCV is available in (Wahba, 1990). For a given set of hyperparameters $\mathbf{h} \in \mathbb{R}^h$, let $\mathbf{p}_{\mathbf{h}}^{(k)}$ be the parameters of the model estimated from the data with the kth data point left out. The LOOCV criterion is defined as:

$$\textsc{loocv}(\mathbf{h}) = \frac{1}{n} \sum_{k=1}^{n} d\left(y_k, f\left(x_k; \mathbf{p}_{\mathbf{h}}^{(k)}\right)\right), \tag{3.41}$$

where d is the function that gives the discrepancy between its two arguments. For instance, it may be the squared Euclidean norm. Equation (3.41) means that every point of the initial dataset is used as a validation set for the model estimated from the other data points. If there are enough points in the data set (say, for instance, $n > 50$) then it is reasonable to think that the parameters estimated with $n-1$ points are close to the ones that would be estimated with all the n initial points. Consequently, for a given k, $d\left(y_k, f\left(x_k; \mathbf{p}_{\mathbf{h}}^{(k)}\right)\right)$ is a good approximation of the generalization error made at x_k. Therefore, $c_{\textsc{loocv}}$ is an approximation of the generalization error resulting of the given set of hyperparameters $\mathbf{h}$.

A major drawback. LOOCV is quite appealing and often gives satisfactory results in practice. However, as one may notice from equation (3.41), it can be extremely heavy to compute. Indeed, each term of the sum in equation (3.41) requires one to estimate the set of parameters $\mathbf{p}_{\mathbf{h}}^{(k)}$. If n is large then the sum in equation (3.41) has many terms, each one of which being potentially long to compute.

A special case of interest: linear least squares. In the case where the cost function is a sum of squared linear terms (which happens if the parametric model is linear with respect to its parameters and if the noise is assumed to be normally distributed), a closed-form non-iterative approximation of the LOOCV criterion exists. The general form of a linear least squares parameter estimation problem is:

$$\min_{\mathbf{p}} \|\mathsf{M}_{\mathbf{h}}\mathbf{p} - \mathbf{y}\|^2, \tag{3.42}$$

where $\mathbf{y}$ is a vector containing the right part of the measurement (the y_i for $i \in [\![1, n]\!]$). The matrix $\mathsf{M}_{\mathbf{h}}$ is a matrix that depends on the left part of the measurements (the $\mathbf{x}_i$ for $i \in [\![1, n]\!]$), on the parametric model of function, and on the hyperparameters $\mathbf{h}$. The solution $\mathbf{p}_{\mathbf{h}}^{\star}$ of problem (3.42) is given by:

$$\mathbf{p}_{\mathbf{h}}^{\star} = (\mathsf{M}_{\mathbf{h}}^{\mathsf{T}}\mathsf{M}_{\mathbf{h}})^{-1}\mathsf{M}_{\mathbf{h}}^{\mathsf{T}}\mathbf{y}. \tag{3.43}$$

Note that $\mathbf{p}^\star_\mathbf{h}$ depends on the hyperparameters $\mathbf{h}$ (which are considered to be given and fixed in equation (3.43)). The *hat matrix*, denoted $\mathsf{H}_\mathbf{h}$, is the matrix of the linear application that maps the measurements to the values predicted by the estimated model (denoted $\hat{y}_i$ for $i \in [\![1, n]\!]$ and gathered in a vector $\hat{\mathbf{y}}$). It is defined by:

$$\hat{\mathbf{y}} = \mathsf{M}_\mathbf{h}\mathbf{p}^\star = \underbrace{\mathsf{M}_\mathbf{h}(\mathsf{M}_\mathbf{h}^\mathsf{T}\mathsf{M}_\mathbf{h})^{-1}\mathsf{M}_\mathbf{h}^\mathsf{T}}_{\mathsf{H}_\mathbf{h}}\mathbf{y}. \tag{3.44}$$

The closed-form approximation to the LOOCV criterion, denoted LOOCV$_a$, is given by:

$$\textsc{loocv}_a(\mathbf{h}) = \frac{1}{n}\left\|\operatorname{diag}\left(\frac{1}{\mathbf{1} - \operatorname{diag}(\mathsf{H}_\mathbf{h})}\right)(\mathsf{M}_\mathbf{h}\mathbf{p}^\star_\mathbf{h} - \mathbf{y})\right\|^2. \tag{3.45}$$

From equation (3.45), we see that for a given set of hyperparameter $\mathbf{h}$, the computational complexity of the criterion LOOCV$_a$ is reduced by one order of magnitude compared to the exact but iterative criterion LOOCV.

Proof. A proof of the derivation of the LOOCV$_a$ criterion as an approximation of the LOOCV criterion can be found in, for instance, (Bartoli, 2008b; Gentle et al., 2004; Wahba, 1990).

Generalized Cross-Validation. Generalized Cross-Validation (GCV) is a variation of the LOOCV$_a$ criterion proposed in (Craven and Wahba, 1979). It consists in replacing the individual diagonal coefficients of the hat matrix in equation (3.45) by the average of all the coefficients of the diagonal. The GCV criterion is thus defined as:

$$\textsc{gcv}(\mathbf{h}) = \frac{\|\mathsf{M}_\mathbf{h}\mathbf{p}^\star_\mathbf{h} - \mathbf{y}\|^2}{\frac{1}{n}(n - \operatorname{trace}(\mathsf{H}))^2}. \tag{3.46}$$

The GCV criterion was originally proposed to reduce the computational complexity but it has been found that this criterion possesses several interesting properties. In particular, it has been shown (Golub et al., 1979) that the GCV criterion is an approximation of Mallow's C_P criterion (see section 3.2.2.3). It has also been shown to be rotation-invariant approximation of the PRESS (PREdiction Sum of Squares) statistic of Allen[7].

Leave-p-out Cross Validation. Leave-p-out Cross-Validation (LPOCV) (Shao, 1993) is another variant of the Cross-Validation principle. It is similar to the LOOCV in the sense that it uses an *exhaustive data splitting* strategy. With the 'basic' LOOCV, all the n possible subsets of size 1 were alternatively used as test sets. With the LPOCV, every possible subset of size p (with p fixed and chosen in $[\![1, n-1]\!]$) is successively used as a test set. Of course, the computational burden of such an approach is potentially gigantic since there are $\binom{n}{p}$ subsets of size p in a set of size n. Note that if $p = 1$ then LPOCV is equivalent to LOOCV.

V-fold Cross-Validation. Another common flavour of the Cross-Validation principle is the V-fold Cross-Validation (VCV). It was introduced by (Geisser, 1975) as an alternative to the computationally expensive LOOCV. A full review of the VCV can be found in (Brabanter et al., 2003). (Arlot and Celisse, 2010) gives some useful insight about the VCV. Again, the initial dataset is split into training and test sets. However, contrarily to LOOCV and LPOCV, the VCV is not an *exhaustive approach*. In the VCV case, the dataset is split into V subsets of approximately equal size. These V subsets form a partition of the initial dataset. This splitting is done before the computation of the criterion and does not vary after that. Each subset is then used alternatively used as a test set. Given a set of hyperparameters $\mathbf{h}$, let $\mathbf{p}^{[v]}_\mathbf{h}$ be the set of parameters obtained from

[7]The PRESS statistic is similar to the LOOCV criterion but for a cost function with a data term only (Allen, 1971; Bartoli, 2008b)

the data with the vth subset left out. Let m_v be the size of the vth subsets. The VCV criterion is defined by:

$$\text{VCV}(\mathbf{h}) = \sum_{v=1}^{V} \frac{m_v}{n} \sum_{k=1}^{m_v} \frac{1}{m_v} d\left(y_k, f\left(x_k; \mathbf{p}_{\mathbf{h}}^{[v]}\right)\right), \tag{3.47}$$

where d is a function that measures the difference between its two arguments (for instance, the squared Euclidean norm). In practice, the number V of subsets used for computing the VCV is often chosen as $V = \min(\sqrt{n}, 10)$. This gives a good trade-off between accuracy of the approximation of the generalization error and the computational burden. Although very appealing, our experience leads us to say that the VCV criterion does not give results as good as the ones obtained with the LOOCV or the GCV.

Other cross-validation criteria. The list of cross-validation criteria we presented so far is not even close to being exhaustive. We will not detail here all the variant. For the sake of the culture, we will just cite a few of them: Gauss-Newton leave-one-out cross-validation (Farenzena et al., 2008), balanced incomplete cross-validation, repeated learning-testing, Monte-Carlo cross-validation, bias-corrected cross-validation, *etc.* Note that (Arlot and Celisse, 2010) gives a good overview on some of these cross-validation criteria.

3.2.2.3 Mallow's C_P

Introduced in (Mallows, 1973), Mallow's C_P is a criterion that can be used to automatically tune hyperparameters. The validity of this criterion is limited to linear least squares estimators. Besides, it is designed to work with one integer hyperparameter. For the sake of simplicity, let us consider that the model is a polynomial and that the only hyperparameter is the degree d of this polynomial. Let us further assume that d_m is the maximal degree allowed for the polynomial (*i.e.* $d \in [\![0, d_m]\!]$). Mallow's C_P is given by:

$$C_P(d) = \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i; \mathbf{p}_d)\right)^2 - n + 2p, \tag{3.48}$$

where p is the number of parameter of the model induced by the hyperparameter d ($p = d + 1$ in the case of a polynomial), $\mathbf{p}_d$ is the maximum likelihood estimate of the parameters given the hyperparameter d, n is the number of data points, and $\hat{\sigma}^2$ is an estimate of the error variance that is usually computed as the residual sum of squares with the full model (in this case, with a polynomial of degree d_m).

Other criteria based on the initial Mallows C_P criterion have been proposed. For instance, robust versions of Mallow's C_P criterion are proposed in (Ronchetti, 1997; Ronchetti and Staudte, 1994).

3.2.2.4 Akaike Information Criterion

Akaike Information Criterion (AIC) is a criterion that relies on information theoretic foundations. It was introduced in (Akaike, 1974). Useful insights on AIC are given in (Burnham and Anderson, 2004; Gentle et al., 2004). In this section, we give the basic elements that allows one to progressively build the AIC criterion. The AIC criterion will finally be given in equation (3.56).

The Kullback-Leibler (K-L) divergence is the quantity defined as:

$$I(\underline{f}, f) = \int_{\Omega} \underline{f}(x) \log\left(\frac{\underline{f}(x)}{f(x; \mathbf{p})}\right) \mathrm{d}x. \tag{3.49}$$

It may be seen as a 'distance'[8] between the true model $\underline{f}$ and the estimated model $f(\bullet; \mathbf{p})$. The K-L divergence

[8]The term 'distance' is abusive in this case since the Kullback-Leibler divergence is not an actual distance in the mathematical

cannot be used directly as a criterion to choose the hyperparameters since it would require one to know the true model. Therefore, the selection criterion will aim at minimizing an expected estimated K-L divergence:

$$E_y[I(\underline{f}, f)]. \tag{3.50}$$

The K-L divergence may rewritten as:

$$I(\underline{f}, f) = \int \underline{f}(x) \log(\underline{f}(x)) \mathrm{d}x - \int \underline{f}(x) \log(f(x; \mathbf{p})) \mathrm{d}x \tag{3.51}$$

$$= E_x[\log(\underline{f}(x))] - E_x[\log(f(x; \mathbf{p}))] \tag{3.52}$$

For varying hyperparameters, the first term in equation (3.52) is a constant, and therefore:

$$I(\underline{f}, f) = c - E[\log(f(x; \mathbf{p}))]. \tag{3.53}$$

The second term in equation (3.52) is the *relative Kullback-Leibler divergence*. In order to get a practical criterion, the relative K-L need to be approximated (because it still depends on $\underline{f}$). This is the goal of AIC:

$$\max E_y E_x[\log(f(x; \mathbf{p}))]. \tag{3.54}$$

Akaike's work says that an approximately unbiased estimate of $\max E_y E_x[\log(f(x; \mathbf{p}))]$ is:

$$\log(\mathcal{L}(\hat{\mathbf{p}})) - k, \tag{3.55}$$

with $\mathcal{L}$ the likelihood function, $\hat{\mathbf{p}}$ the maximum likelihood estimate of $\mathbf{p}$, and k the number of parameters. The AIC is given by:

$$\text{AIC} = -2 \log(\mathcal{L}(\hat{\mathbf{p}})) + 2k. \tag{3.56}$$

The terms in equation (3.56) can be identified to the terms in the bias-variance trade-off: the first term corresponds to the bias while the second term corresponds to the variance.

A special case: least-squares. If the errors are assumed to be *i.i.d.* normally-distributed, the AIC can be expressed by:

$$\text{AIC} = n \log \left(\frac{RSS}{n} \right) + 2k, \tag{3.57}$$

where $RSS = \sum_{i=1}^{n} r_i^2$.

Variations on AIC. Although theoretically founded, several improvement and variations have been built upon the AIC. For instance, if the sample size is small with respect to the number of parameters, then it is preferable to use the corrected AIC given by (Sugiura, 1978):

$$\text{AIC}_c = -2 \log(\mathcal{L}(\mathbf{p})) + 2k + \frac{2k(k+1)}{n-k-1}. \tag{3.58}$$

definition of it.

3.2.2.5 Bayesian Information Criterion

The Bayesian Information Criterion (BIC) is another criterion that can be used to automatically tune hyperparameters. It was proposed by (Schwarz, 1978). The BIC is defined as follows:

$$\text{BIC} = -2\ln(\mathcal{L}) + k\log(n). \tag{3.59}$$

Equation (3.59) shows that BIC is very similar to the AIC. The practical difference between these two criteria is that BIC tends to penalize complex models more heavily than AIC (because the second term in equation (3.59) depends on the number n of model parameters). The other major difference between AIC and BIC is that BIC is motivated in a quite different way than AIC. To this matter, and as noticed in (Burnham and Anderson, 2004), the name BIC is a bit misleading since this criterion does not rely on information theory but, instead, on a Bayesian approach to model selection. A synthetic explanation of the BIC foundations is given in (Hastie et al., 2001, §7.7). We now give a quick justification of the BIC. Let F be the set of models generated by all the possible hyperparameters, *i.e.* :

$$F = \left\{ f_{\mathbf{h}} = f(\bullet; \mathbf{p}_{\mathbf{h}}, \mathbf{h}) \mid \mathbf{h} \in \mathbb{R}^h \right\}, \tag{3.60}$$

where $f : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^h \to \mathbb{R}$ is the parametric model (in which we write explicitly the dependency on the hyperparameters $\mathbf{h}$). The goal is to find the best element in the set of functions F. Let us assume that we have a prior distribution $P(\mathbf{p}_{\mathbf{h}}|f_{\mathbf{h}})$ of the parameters of each model in F. The posterior probability of a given model is given by:

$$P(f_{\mathbf{h}}|X) \propto P(f_{\mathbf{h}})P(X|f_{\mathbf{h}}) \tag{3.61}$$

$$\propto P(f_{\mathbf{h}}) \int P(X|\mathbf{p}_{\mathbf{h}}, f_{\mathbf{h}})P(\mathbf{p}_{\mathbf{h}}|f_{\mathbf{h}})\mathrm{d}\mathbf{p}_{\mathbf{h}}, \tag{3.62}$$

where $X = \{x_i \leftrightarrow y_i\}_{i=1}^n$ represents the data. If one wants to compare the relative quality of two set of hyperparameters $\mathbf{h}$ and $\mathbf{h}'$, he can look at the ratio of their posterior probability:

$$\frac{P(f_{\mathbf{h}}|X)}{P(f_{\mathbf{h}'}|X)} = \frac{P(f_{\mathbf{h}})}{P(f_{\mathbf{h}'})}\frac{P(X|f_{\mathbf{h}})}{P(X|f_{\mathbf{h}'})}. \tag{3.63}$$

If the ratio in equation (3.63) is greater than one, the model induced by the hyperparameters $\mathbf{h}$ is considered to be better than the one induced by the hyperparameters $\mathbf{h}'$. It is a reasonable assumption to consider that the prior distribution of the models is uniform. Consequently, the first factor in equation (3.63) is just a constant. The second factor (known as the *Bayes factor*) can be approximated using this identity (Hastie et al., 2001):

$$\log(P(X|f_{\mathbf{h}})) \approx \log(P(X|\hat{\mathbf{p}}_{\mathbf{h}}, f_{\mathbf{h}})) - \frac{d_{\mathbf{h}}}{2}\log(n), \tag{3.64}$$

where $\hat{p}_{\mathbf{h}}$ is the maximum likelihood estimate of the parameters given the hyperparameters and $d_{\mathbf{h}}$ is the number of free parameter in $f_{\mathbf{h}}$. This last equation is equivalent to the BIC. Consequently, choosing the hyperparameters that give the minimum BIC is equivalent to maximize the (approximate) posterior probability.

3.2.2.6 Minimum Description Length

The Minimum Description Length (MDL) criterion is another criterion that can be used to determine automatically hyperparameters. It was first introduced by (Rissanen, 1978) and it relies on information theory. A good synthesis of the underlying principles of MDL are given by (Nannen, 2003a,b). In a nutshell, MDL is a formal implementation of the well known principle of the Occam's razor which states that "entities must not be multiplied beyond necessity"[9]. In other words, if several models can explain the same set of data, the simplest is most likely to be the correct one. Although appealing, this philosophical principle should be considered with extreme care, especially because the notion of 'simplest' is quite unclear. Without care, it is virtually possible to consider as the simplest any set of hyperparameters.

The formal definition of the MDL criterion is actually exactly the same[10] than the definition of the BIC, *i.e.* :

$$\text{MDL} = -\log(\mathcal{L}) + \frac{k}{2}\log(n). \tag{3.65}$$

The fundamental difference between BIC and MDL lies in their theoretical roots. Indeed, MDL relies on information theory and, in particular, the Kolmogorov complexity. The Kolmogorov complexity, also known as the algorithmic complexity or Turing complexity, is a concept that was independently developed by (Chaitin, 1966; Kolmogorov, 1965; Solomonoff, 1964). In a nutshell, it measures the complexity of a message (which can be seen as a particular way of describing a parametric model) as the length of the minimal Turing machine that generates this same message. Unfortunately, the Kolmogorov complexity cannot be calculated because it is related to the famous *halting problem* which was proven by Turing (1936) to be incomputable. Fortunately, a famous theorem due to (Shannon, 1948) gives a lower bound for the Kolmogorov complexity:

$$E[\text{len}(z)] \geq -\int P(z)\log_2(z)\mathrm{d}z, \tag{3.66}$$

where $\text{len}(z)$ is the length of the binary string encoding the continuous random variable z of distribution $P(z)$. A consequence of this theorem is that encoding a continuous random variable z of distribution $P(z)$ requires approximately $-\log_2(P(z))$ bits of information[11]. We replace the binary logarithm by the $\log$ function since the only difference is an unimportant multiplicative constant. Let us now consider the dataset made of the inputs $X = \{\mathbf{x}_i\}_{i=1}^n$ and of the outputs $Y = \{y_i\}_{i=1}^n$. The required length for encoding Y is the sum of the length required to encode the parameters of a model and of the length required to encode the discrepancy between the true model and the actual measurement, *i.e.* :

$$-\log(P(Y|\mathbf{p}, f, X)) - \log(P(\mathbf{p}|f)). \tag{3.67}$$

The first term is the negative logarithm of the posterior probability while the second term is the average length for encoding the parameters. (Rissanen, 1978) postulated from results by (Shannon, 1948) that the length of the code for $\mathbf{p}$ was $\log(\sqrt{n})$ for each parameter, hence the final expression of MDL. Minimizing the MDL criterion is therefore equivalent to determine the simplest representation of the data, which is why MDL is considered as a formal implementation of the Occam's razor principle.

3.2.2.7 Other Approaches

L-curve, Morozov's discrepancy principle. In addition to the very generic and widely spread criteria presented in the previous section to automatically compute hyperparameters, there exists others more specific (and generally less-known) approach to determine hyperparameters. Some of these criteria (such as the L-curve or the Morozov's discrepancy principle) will be used in this document. Given their lack of generality, we do not

[9] *entia non sunt multiplicanda praeter necessitatem* (as it was said in a time where English had not overruled everything else)

[10] Except for a multiplicative positive constant which plays no role since the goal is to minimize the criterion.

[11] Note that here we do not bother with the precision of the representation of z which may be infinite since it is a real number but which may be reasonably approximated.

review them in this section. Instead, we will detail them when appropriate in the document.

One. Although it is not a founded approach, it appears that one of the most common technique to determine hyperparameters is probably to give them some empirical value determined by hand. A very popular choice is the value 1. This is what we call, with a bit of irony we must admit, the '$\lambda = 1$ approach'. Note that this trivial approach has been proved to give reasonable results in many problems (at least with appropriately chosen instances of these problems). This is probably why a lot of researchers do not believe that the determination of the hyperparameters is a problem of interest. In our work, we beg to differ and we will, as much as possible, give details on how hyperparameters are chosen.

Selection of the hyperparameters based on experiments. Another common approach is to determine typical values for the hyperparameters by looking at the results of extensive experiments. This type of approach has been used in, for instance, (Pilet et al., 2008; Pizarro and Bartoli, 2010). Even though the exact way in which hyperparameters are selected is not always mentioned in scientific papers, we can imagine that most authors use extensive experimentations to determine typical values. Note that, sometimes, 'typical values' do not exist. This is the case when there exists an important variability between instances of the same problem. For instance, it is not reasonable to consider that there exists a set of typical hyperparameters for range surface fitting (see chapter 4). Indeed, in this case, the choice of the hyperparameters heavily relies on the shape of the observed scene.

Chapter 4

Range Surface Fitting

Transforming a discrete representation of a surface (*i.e.* a finite set of 3D points) into a smooth and analytical form is a problem of wide interest. Such a representation can then be used for complex computations such as the estimation of geodesic distances. Finding a smooth surface that approximates a set of 3D points has other advantages. In particular, it allows one to reduce the amount of noise inherent to datasets acquired with real sensors such as Time-of-Flight cameras. A large part of our work is dedicated to this problem in the case of range data.

In this chapter, we tackle the problem of fitting an analytic surface to range data. We start by giving the first definitions and concepts related to range data. In particular, we give a brief review of the sensors and techniques that allows one to acquire range data. We also give an introductory and practical example that allows us to explain the basic concepts of range data fitting. This example can also be considered as a practical implementation of the concepts related to parameters and hyperparameters estimation that were reviewed in the previous two chapters.

We then present our contributions related to range data fitting. We propose two main contributions which will be presented in separate sections. First, we propose a new criterion, the *L-Tangent Norm*, that enables one to automatically select the regularization trade-off in range surface fitting (Brunet et al., 2008). Second, we propose a method for fitting a surface to range data with heteroskedastic noise, *i.e.* noise whose variance is not the same for all the data points (Brunet et al., 2009a). By exploiting the fact that the data are arranged on a regular grid, our method features a low computational complexity.

4.1 First Definitions and Concepts

4.1.1 First Definitions

Range surface. A range surface is a particular type of 3D surface. It is different from a general 3D surface in the sense that it is defined as a set of depths with respect to 2D coordinates. A topographic map is a good example of range surface: to each pair (longitude, latitude) is associated an altitude. In a manner of speaking, range surfaces are a subset of general 3D surfaces. This is the reason why range surfaces are sometimes called 2.5D surfaces. An example of range surface is a 3D scene observed from a single point of view. Compared to a full 3D surface, a range surface may introduce difficulties. Indeed, even if we consider a continuous 3D world, the range surface corresponding to a point of view may have discontinuities. Discontinuities are generally hard to handle.

Mathematically speaking, range surfaces are adequately modelled by scalar-valued functions from $\mathbb{R}^2$ (or a subset of it) to $\mathbb{R}$. Such surfaces are also known as Monge patches (Gray, 1997). Figure 4.1 gives an example of range surface.

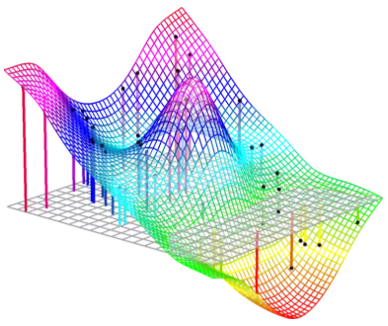


Figure 4.1: An example of range surface. In a range surface, an elevation is associated to each couple of 2D points on the plane of the grey grid.

Range data. Range data are data points which follow the same principle as range surfaces. Range data points are written using the following notation:

$$\mathbf{x} \leftrightarrow z, \tag{4.1}$$

where $z \in \mathbb{R}$ is the altitude (or elevation, or depth) corresponding to the 2D point $\mathbf{x} \in \mathbb{R}^2$. Note that $\left[\mathbf{x}^\mathsf{T} \ z\right]^\mathsf{T}$ is a classical 3D point. An example of range data points is given in figure 4.2.

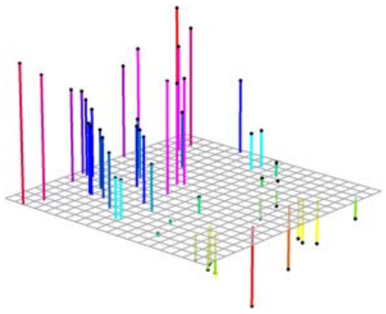


Figure 4.2: An example of range data points (black dots). The coloured segments represent the depth of each point.

Depth map. A depth map is a set of range data points organized on a regular and uniformly spaced grid[1]. In other words, a depth map is like a standard digital picture (*i.e.* an array of numbers each one of which coding for a colour or a level of grey) but the numbers represent distances instead of colours. The individual elements of a depth map are still named pixels even though they are not 'little squares of colours' but elementary pieces of surface.

4.1.2 Acquisition of Range Data

Nowadays, there exists many approaches and sensors to acquire range data. We emphasize here that the acquisition of range data is not the central topic of this thesis. We are more interested in the processing of range data than in its acquisition. Therefore, we only give a brief overview of the ways in which range data may be acquired.

4.1.2.1 Generalities

The devices that allows one to acquire range data may be categorized into two classes:

Passive sensors. The central part of the capturing device is a 'simple' passive sensor such as a CMOS or a CCD sensor. Nothing is emitted from the device. The acquisition of range data with such sensors is therefore completely non-intrusive in the sense that the observed scene is not 'modified' for the acquisition. The main part of these approaches generally relies on complex algorithms.

Active sensors. In a way or another, the capturing device illuminates the scene in order to get supplementary information about its geometry. The data acquired with such sensors are generally of better quality than with passive sensors. However, these sensors are more invasive[2] than passive sensors. Complex algorithms are also used in the acquisition process.

4.1.2.2 Passive Sensors

We give here a non-exhaustive list of passive technique for acquiring range data. As we already mentioned, we do not pretend to make a detailed review on range data acquisition. However, each one of the succinctly described method is associated to some references. It is a common practice to name these techniques with the prefix 'Shape-from-'. As a casual remark, it would have been more correct to use the prefix 'Range-from-'.

Shape-from-Stereo (stereo reconstruction). Stereo reconstruction is an important problem in computer vision which has been addressed in many ways. In a certain sense, it consists in emulating the human vision by exploiting the discrepancies between two images of the same scene taken from a different point of view. The core principle is to use the discrepancies between two images of the same scene to infer 3D information. This may be achieved using, for instance, point triangulation and epipolar geometry (see figure 4.3). The interested reader may found a complete review of such techniques in documents such as (Bartoli and Sturm, 2004; Faugeras, 1993; Faugeras et al., 2004; Hartley and Zisserman, 2003a; Xú and Zhang, 1996). Other approaches exist such as the reconstruction of discrepancy maps (Boykov et al., 2001; Kanade and Okutomi, 1991; Scharstein and Szeliski, 1998)

[1]As for standard digital pictures, extensions to non-uniform grids could be envisaged but we take the side of not considering them in this manuscript.

[2]Although there are now active sensors that illuminates the scene using infra-red (or 'near infra-red') wavelengths which are not visible to the human eye.

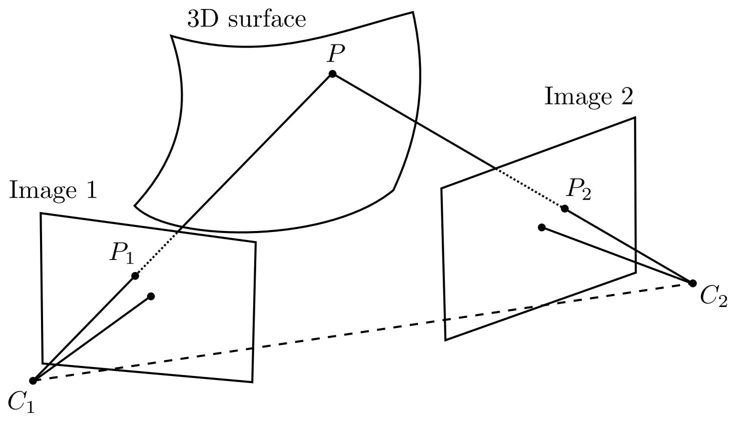


Figure 4.3: Illustration of the basic principles of shape-from-stereo (stereo reconstruction). The reconstruction of a 3D point may be achieved using the epipolar geometry.

Shape-from-Shading. Shape-from-shading is a technique that reconstruct the 3D structure of a scene from the lighting of the object. Detailed information on shape-from-shading may be found in (Zhang et al., 1999). Usually, some general assumptions are made. For instance, one may consider that there is only one light source. One may further assume that the surfaces of the scene exhibits Lambertian reflectance which means that the intensity reflected by a surface to an observer is the same regardless of the observer's angle of view. We may then apply the Lambert's cosine law which state that the intensity i_s of a surface at a given point is given by:

$$i_s = \mathbf{l}^\mathsf{T}\mathbf{n}ci_l, \tag{4.2}$$

with $\mathbf{l}$ a vector pointing from the surface to the light source, $\mathbf{n}$ the surface's normal vector, c the colour of the surface, and i_l the intensity of the light source (see figure 4.4). By definition of the scalar product of two vectors, we then have that

$$\mathbf{l}^\mathsf{T}\mathbf{n} = \|\mathbf{n}\|\|\mathbf{l}\|\cos(\alpha), \tag{4.3}$$

where α is the angle between the vectors $\mathbf{n}$ and $\mathbf{l}$. In shape-from-shading, the observed intensity is used to compute a field of normal vectors of the surface with equation (4.3). This field is then integrated to reconstruct the surface.

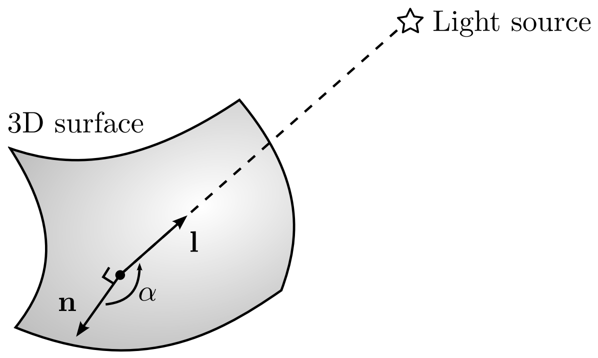


Figure 4.4: Illustration of Shape-from-Shading. Under some assumptions about the observed surface and the lighting conditions, it is possible to retrieve a field of normal vectors from the colour of the surface. The actual 3D surface is obtained by integrating the vector field.

Shape-from-Motion. Shape-from-Motion is a technique that retrieve the shape of a scene from the spatial and temporal changes occurring in an image sequence. More details on this technique may be found in (Bartoli and Sturm, 2005; Del Bue and Agapito, 2006, 2007; Olsen and Bartoli, 2008; Tomasi and Kanade, 1992). To some extent, it is similar to shape-from-stereo. There are nonetheless some important differences. In particular, the typical discrepancies between two successive images are much smaller than those of typical stereo pairs. This makes shape-from-motion more sensible to noise than shape-from-stereo. Another important difference is the fact that the transformation linking two consecutive frames is not necessarily a single rigid 3D transformation. Compared to shape-from-motion, shape-from-motion has its advantages. For instance, pixel correspondences may be easier to find with shape-from-motion since the baseline is smaller than with typical shape-from-stereo.

Shape-from-Focus. Shape-from-focus is a technique that reconstruct range data by acquiring two or more images of the same scene under various focus settings (Nayar and Nakagawa, 1994). Once the best focused image is found, a model that links focus and distance is used to determine the distance.

Shape-from-Texture. Shape-from-texture is a technique that relies on the deformation of the individual texels (texture elements) to retrieve 3D information from a scene. Complementary information on this technique may be found in (Aloimonos, 1988). The shape reconstruction typically exploits two types of distortion: perspective and foreshortening transformations (see figure 4.5). The perspective distortion makes the objects far from the camera appear smaller. The foreshortening distortion makes objects not parallel to the image plane shorter. It is possible to estimate the amount of those distortions from the textures contained in an image. As for other methods described in this section, the raw output of this algorithm is a discrete field of normal vector to the observed surface. If this field is dense enough and if the assumption of a smooth surface is made, then it is possible to recover the whole surface shape.

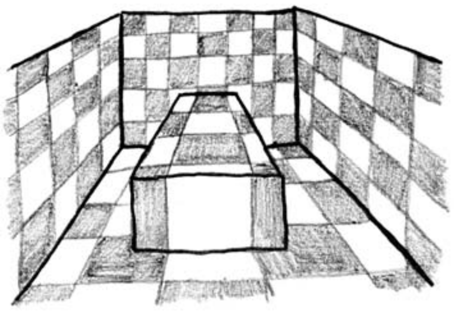


Figure 4.5: Illustration of Shape-from-Texture. This technique exploits the local transformations (perspective and foreshortening deformations) of the texture to reconstruct a field of normal vectors to the surface.

4.1.2.3 Active Sensors

We now explain the basic principles of a few active sensors used to acquire range data. We only consider sensors commonly used in computer vision.

Structured light. Structured light is a technique to reconstruct the geometry of the scene that uses a light projector and an intensity camera placed at a distance from each other. Detailed information on this technique may be found in (Zhang, 2005). The projector emits a light pattern (whose shape is perfectly known). For instance, the pattern may be a simple line. From the point of view of the camera, the projected line is a distorted curve whose shape depends on the shape of the scene.

Laser-based sensors (LIDAR). LIDAR (LIght Detection And Ranging) is an optical sensor that measures properties of reflected light to determine the distance of a distant target. The LIDAR technology is also known under the acronym LADAR (LAser Detection And Ranging). In a nutshell, it relies on the same principles as

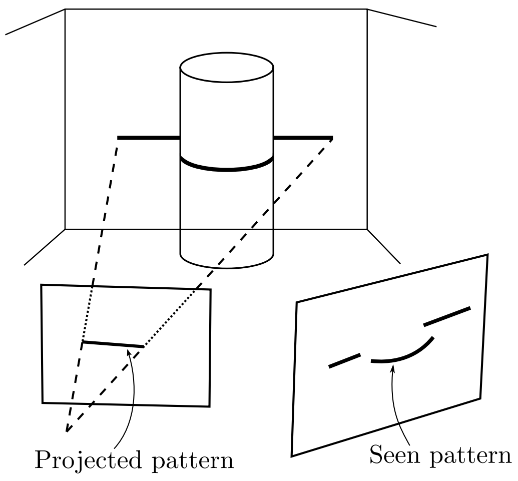


Figure 4.6: Basic principles of 3D reconstruction using structured light: a pattern with known shape is projected on the scene. A camera observes the pattern. An algorithm deduces the shape of the scene from the deformation of the pattern.

the standard RADAR. The main difference is that RADAR uses radio waves while LIDAR uses much shorter wavelength (typically in the ultraviolet, visible, or near infrared range). Given the fact that such sensors are generally not able to detect objects smaller than the wavelength they use, LIDAR is able has a much finer resolution than RADAR. The distance of an object is determined by measuring the time delay between the transmission of a pulse and detection of the reflected signal (see figure 4.7). A laser has a very narrow beam. This allows one to make a measurements at a very fine precision. Note that the measurements are generally 'point-wise', *i.e.* the distance of only one point is computed. Having a complete range map of a scene requires one to make multiple measurements. Since there is necessarily a delay between two measurements, it can be a problem if the scene is not strictly rigid.

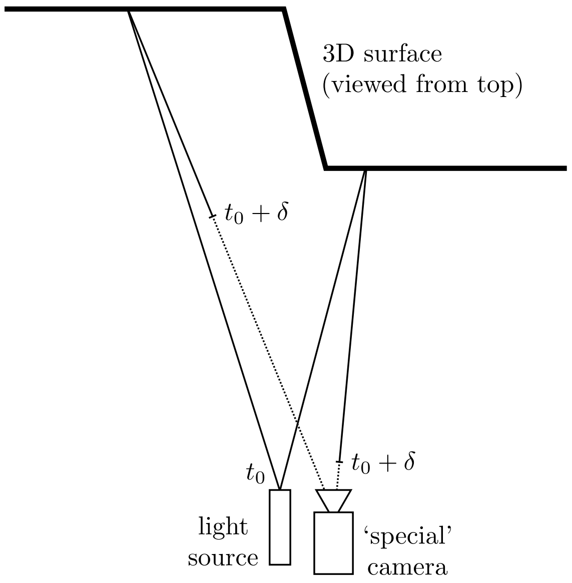


Figure 4.7: Basic principle of LADAR. With this technique, point-wise measurements of depths are achieved by measuring the time delay between the emission of a laser pulse and the reception of that pulse by a special sensor.

Time-of-Flight Cameras The general principle of time-of-flight (ToF) cameras is similar to that of LIDAR sensors. However ToF cameras have the advantage of being able to capture a whole scene at the same time. Besides, ToF cameras can provide up to 100 images (range maps) per second. Additional informations on ToF cameras may be found in (Falie and Buzuloiu, 2007; Gokturk et al., 2004; Luan, 2001; Viarani et al., 2004; Xu et al., 2005). ToF cameras have been used in many fields such as medical imaging (Penne et al., 2009; Ulrich et al., 2010), augmented reality (Bartczak et al., 2008; Koch et al., 2009), intelligent vehicles (Hsu et al., 2006).

The general principle of ToF cameras is illustrated in figure 4.8. The device is made of two parts: a near infrared light source and an optical sensor. The light source emits a phase-modulated signal. The optical sensor is a 'variant' of a standard CMOS sensor: in addition to the intensity of the reflected light, it is also able to capture the phase of the received signal. The signal modulation is synchronized between the light source and the optical sensor. For each pixel of the optical sensor, the depth is computed by comparing the current phase modulation (at the time of reception) with the phase of the received signal. Note that there exist ToF cameras that use slightly different principle for retrieving depths. However, the principle described above is the one used by the cameras we had access to, namely the products from PMDTec[3] (cameras 19k-S) and Mesa Imaging[4] (SwissRanger 3000 and SwissRanger 4000).

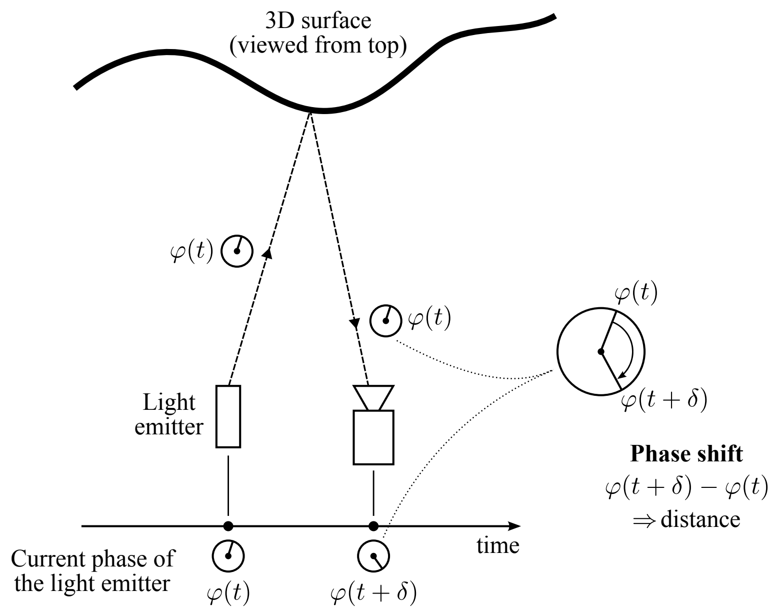


Figure 4.8: Basic principle of ToF cameras. ToF cameras are able to capture range maps by measuring a phase delay for each one of the pixels.

Although very useful and efficient, ToF cameras have also some drawbacks. For instance, they have a resolution which is quite limited if we compare them to the typical resolutions of standard digital cameras. The range of non-ambiguous distances is also limited: typically 7 metres. This range may be increased but it is achieved to the detriment of the depth precision. Another drawback of ToF cameras is that the depth measurements are influenced by the material (mainly by its reflexivity). The accuracy of the measurements are also influenced by multiple reflexions of the emitted infrared signal (this phenomenon often happens when the observed scene has concave corners). The measurements may also be perturbed by ambient light. As a consequence of these drawbacks, the depth maps acquired with ToF cameras are usually extremely noisy. However, one should put these disadvantages into perspective since ToF cameras are amongst the only sensors able to provide 3D information in real-time (and with a high acquisition rate) regardless of the appearance of the observed scene[5].

[3]`http://www.pmdtec.com`
[4]`http://www.mesa-imaging.ch`
[5]Contrarily to other approaches such as shape-from-texture or shape-from-shading which work only under some quite strong assumptions on the scene.

4.1.3 An Introductory Example

In this section, we present an example that illustrates the basic principle for fitting a parametric surface to range data. More generally, this example is also the occasion to make concrete the theoretical tools presented in the previous chapters. In particular, we will show how parametric estimation can be achieved with range data. We also give an example of automatic hyperparameter selection. More advanced topics on range surface fitting, including our contributions, will come later in this chapter.

Data. In this example, we use geographical range data extracted from a topographic map[6]. The data, shown in figure 4.9, represent *Puy Pariou*, an extinct volcano near *Clermont-Ferrand*. Here, the data does not have any particular structure. In particular, the data points are not arranged on a regular grid. As they were extracted manually from a topographic map, there is not much noise in the data points.

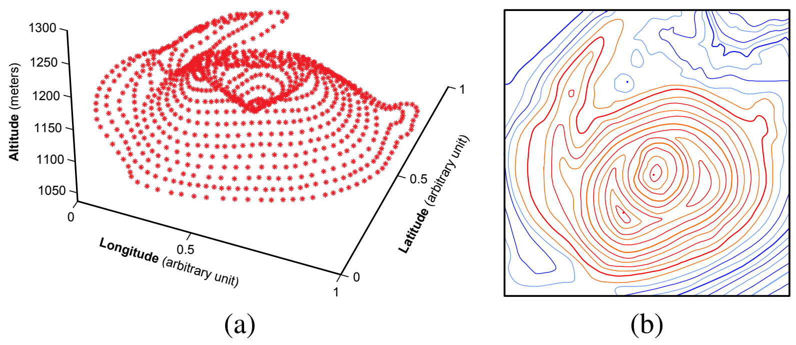


Figure 4.9: Example of range data representing the *Puy Pariou*, a volcano in *Chaîne des Puys* next to Clermont-Ferrand, France. (a) Data represented as points. (b) Data represented with contour lines.

Range Surface Fitting. We now give an (oversimplified) approach to range surface fitting. Given a set of range data points $\{\mathbf{x}_i \leftrightarrow z_i\}_{i=1}^{n}$, the goal of range surface fitting is to find a function $f : \mathbb{R}^2 \to \mathbb{R}$ that fits the data point, *i.e.*:

$$f(\mathbf{x}_i) = z_i \qquad \forall i \in [\![1, n]\!]. \tag{4.4}$$

In practice, the equality in equation (4.4) does not make much sense. Indeed, there is usually noise in the data points. This noise can come from the underlying physics of the sensor used to acquire the data (such as a Time-of-Flight camera or a LIDAR). Let us note $\underline{f}$ the true surface. The available data may be seen as a noisy discretization of the true surface, *i.e.*:

$$z_i = \underline{f}(\mathbf{x}_i) + \varepsilon_i, \tag{4.5}$$

where the ε_i (for $i \in [\![1, n]\!]$) are random variables. The goal of range surface fitting is to find a function f as close as possible of $\underline{f}$.

This problem can be turned into a parameter estimation problem. It means that the range surface fitting problem will be formulated as a minimization problem, that consists in finding a set of parameters $\mathbf{p}^\star$ that minimizes a certain cost function. From this point of view, one must answer the following five questions:

- What parametric model should we use?
- What is the probability distribution of the noise (given equation (4.5), we have already assume that the noise is additive)?

[6]The data were retrieved from `www.geoportail.fr`

- What assumptions (priors) can we make about the surface to reconstruct?
- What optimization algorithm can be used?
- How can the hyperparameters be automatically selected?

In the next paragraphs, we discuss these five points and illustrate them using our example of range surface fitting with simplified assumptions. Note that this set of five questions is quite generic in the sense that they have to be answered for almost any parametric estimation problem.

What parametric model should we use? The choice of a correct parametric model is important. It must be flexible enough to be able to model correctly the data. Ideally, the true surface $\underline{f}$ should be a special case of the parametric model. In other words, if $f : \mathbb{R}^2 \times \mathbb{R}^p \to \mathbb{R}$ is the parametric function model, there should exist a set of parameters $\underline{\mathbf{p}}$ such that $\underline{f} = f(\bullet; \underline{\mathbf{p}})$. In practice, such a condition is difficult to satisfy because $\underline{f}$ is unknown and it may vary in great proportion. Therefore, the function model f is just expected to have enough degrees of freedom. One should also take care of the fact that the parametric model should be simple enough in order to get tractable computations.

For the range surface fitting problem, we may use a uniform cubic B-spline model with a relatively high number of control points. This choice is motivated by the fact that such a model is quite generic, has a high number of degree of freedom, and generally leads to 'simple' computations.

What is the 'nature' of the noise? Determining the 'nature' of the noise allows one to derive a proper cost function using, for instance, one of the statistical approaches presented in section 3.1.1, namely Maximum Likelihood Estimation (MLE) or Maximum A Posteriori Estimation (MAP). A good model of the noise is particularly important to cope with noise and outliers.

In our specific example, we consider that the errors ε_i (for $i \in [\![1, n]\!]$) are all normally-distributed with zero mean and that they all have the same variance. We also assume that the errors are independent. Such an hypothesis on the noise is generally a reasonable choice. Besides, it naturally leads to a least-squares optimization problem, as explained in section 3.1.2.1.

What assumptions (priors) can we make about the surface to reconstruct? A proper parametric model and a proper model of noise may not be sufficient to fit a surface to range data. For example, if the number of degrees of freedom of the parametric model is greater than the number of data points, it will result in an under-constrained problem. Even if there are enough data points, there still can be ambiguities due to the noise: is a small bump in the data just a noisy measurement or the true shape of the surface? These problems can be addressed by using a MAP estimation scheme with extra assumptions on the surface. For instance, one may require the surface to be 'smooth'.

In our example, we want the surface to minimize the bending energy (while still being close to the data points, of course). The bending energy is defined by:

$$B(\mathbf{p}) = \iint_{\mathbb{R}^2} \left(\frac{\partial^2 f}{\partial x^2}(\mathbf{x}; \mathbf{p})\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}(\mathbf{x}; \mathbf{p})\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}(\mathbf{x}; \mathbf{p})\right)^2 \mathrm{d}\mathbf{x}. \tag{4.6}$$

More will come about the bending energy in the sequel of this document. For now, let us just say that the lower the bending energy the smoother the surface and that a surface of minimal bending energy is a plane. Imposing a small bending energy can be seen as a physical prior on the surface. Indeed, it is equivalent to say that a set of parameters leading to a smooth surface is more probable than another set of parameters. This can be formulated in statistical terms by saying that, for instance, we have a prior probability defined as:

$$P(\mathbf{p}) \propto \exp(-\lambda B(\mathbf{p})). \tag{4.7}$$

Such an expression makes surfaces with minimal bending energy the most probable surfaces and this probability decreases when the bending energy becomes higher. The parameter λ controls the width of the bell-shaped curve defined by equation (4.7). A small λ results in a wide bell-shaped curve which means that surfaces with high bending energy are possible with a reasonably high probability. On the contrary, a high value for λ results in a narrow bell-shaped curve which is equivalent to say that surface with high bending energy are utmost improbable.

What optimization algorithm can be used? The assumptions made on the noise and on the surface prior together with the MAP principle leads to the following minimization problem:

$$\min_{\mathbf{p}} \sum_{i=1}^{n} (f(\mathbf{x}_i; \mathbf{p}) - z_i)^2 + \lambda B(\mathbf{p}). \tag{4.8}$$

Since we decided to use the B-spline model and given the expression of the bending energy B, the cost function in equation (4.8) is a sum of squared terms, each one of which being linear with respect to the parameters $\mathbf{p}$. Consequently, equation (4.8) is a linear least squares minimization problem. The tools and algorithms described in section 2.2.2.6, section 2.2.2.8, section 2.2.2.9, and section 2.2.2.10 can thus be used to solve this minimization problem.

How can the hyperparameters be automatically selected? In this example, we consider that the only hyperparameter of interest is the *regularisation parameter* λ in problem (4.8). It is reasonable to make this assumption. Indeed, one could also have considered the number of control points of the B-spline. We think that, in this context, it is simpler to consider a large number of control points. The resulting high complexity of the model is then compensated by the regularisation prior.

In this example, we have decided to use the ordinary cross-validation approach to automatically determine a proper value for the regularization parameter. Since problem (4.8) is a linear least-squares optimization problem, it is possible to use the closed-form formula for computing this criterion. The profile of the cross-validation criterion is give in figure 4.10.

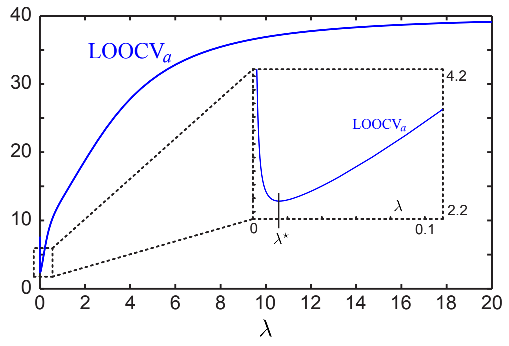
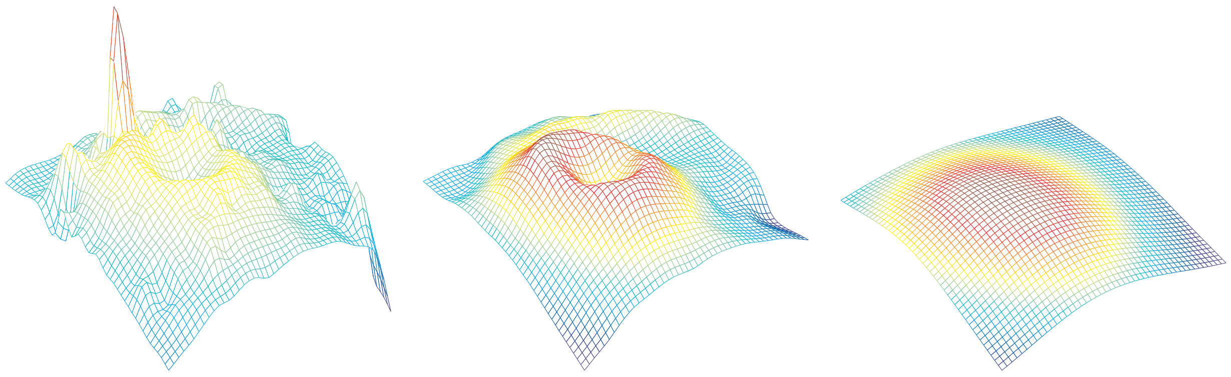


Figure 4.10: Ordinary cross-validation score function (LOOCV$_a$) for selecting the regularization parameter λ in our example of range surface fitting. Here, the optimal value is $\lambda = 0.015$.

Final results. The final result of the range surface fitting process is shown in figure 4.11 (b). Figures 4.11 (a) and (c) respectively illustrate the overfitting and underfitting phenomena that arise if the regularization parameter is not properly chosen.



(a) $\lambda = 10^{-4}$ (b) $\lambda = 0.015$ (c) $\lambda = 2$

Figure 4.11: Final results for our example of range surface fitting. (a) Overfitting phenomenon (the prior on the regularisation is not strong enough and, consequently, the complexity of the surface model is much larger than the actual complexity of the data). (b) Correct regularization parameter chosen by minimizing the cross-validation criterion. (c) Underfitting phenomenon (the large regularization parameter overly reduces the effective complexity of the model).

4.2 The L-Tangent Norm

In this section, we expose one of our contributions concerning range surface fitting. This work was first published in (Brunet et al., 2008). As it was previously presented, range surface fitting is commonly achieved by minimizing a compound cost function that includes two terms: a goodness of fit term and a regularization term. The trade-off between these two terms is controlled by the so-called regularization parameter. Many approaches can be used to determine automatically a proper value for this hyperparameters. Some, such as the cross-validation, are very generic. We already reviewed them in section 3.2.2. Some others are more specific in the sense that they have been designed to work with the type of cost function considered in this section, *i.e.* a mix of a data term and a regularization term. This is the case of the *L-curve* approach (which will be detailed later in this section). However, all these methods are not fully satisfactory. Indeed, the methods based on cross-validation generally suffer from their computational complexity. The L-curve is more efficient in terms of the computational burden but the resulting criterion is hard to minimize in the sense that there are usually many local minima. Therefore, we propose a new criterion to tune the regularization parameter in the context of range surface fitting. We called this new criterion the *L-Tangent Norm* (LTN). Even though empirical, the LTN gives sensible results with a much lower computational cost.

This section is organized as follows. Firstly, we give supplemental details on range surface fitting. In particular, we give the details on the bending energy term for the B-spline model. Secondly, we review another criterion for hyperparameter selection: the L-curve criterion. Thirdly, we present our new criterion: the L-Tangent Norm. Finally, we conclude with some experimental results on both synthetic and real data.

4.2.1 Supplementary Details on Range Surface Fitting

4.2.1.1 Generalities

The main building blocks of range surface fitting have been presented in the introductory example of section 4.1.3. In this section, we also use the B-spline model to represent the surface. We use the same principle as the one utilized in the introductory example. However, we use a slightly different way of writing it:

$$\min_{\mathbf{p}} \mathcal{E}_d(\mathbf{p}) + \frac{\lambda}{1-\lambda}\mathcal{E}_r(\mathbf{p}), \tag{4.9}$$

where $\mathcal{E}_d$ is the data term that measures the discrepancy between the fitted surface and the data, and $\mathcal{E}_r$ is the bending energy term that measures how 'smooth' the surface is. Besides, the regularization parameter is reparametrized so that it lies within $[0, 1[$ instead of $[0, \infty[$. If we denote by $f : \mathbb{R}^2 \times \mathbb{R}^p \to \mathbb{R}$ the parametric model of surface, then we have that:

$$\mathcal{E}_d(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n} \left(f(\mathbf{q}_i; \mathbf{p}) - z_i\right)^2, \tag{4.10}$$

$$\text{and } \mathcal{E}_r(\mathbf{p}) = \iint_{\Omega} \sum_{d=0}^{2} \binom{2}{d} \left(\frac{\partial^2 f(\bullet; \mathbf{p})}{\partial x^{2-d}\partial y^d}\right)^2 \mathrm{d}x\mathrm{d}y, \tag{4.11}$$

where $\Omega \subset \mathbb{R}^2$ is the definition domain of the range surface. It can be chosen as, for instance, the bounding box of the data points. Since f is a tensor-product B-spline, it can be written as a linear combination of the control points, *i.e.* :

$$f(\mathbf{q}; \mathbf{p}) = \mathbf{n}_{\mathbf{q}}^{\mathsf{T}}\mathbf{p}, \tag{4.12}$$

where $\mathbf{w_q}$ is the vector defined by (with $\mathbf{q}^{\mathsf{T}} = (x\ y)$):

$$\mathbf{n}_{\mathbf{q}}^{\mathsf{T}} = \Big(N_{-3}(x)N_{-3}(y) \ldots N_{-3}(x)N_p(y) \quad \ldots \quad N_p(x)N_p(y)\Big) \tag{4.13}$$

Consequently the data term $\mathcal{E}_d$ can be written in matrix form:

$$\mathcal{E}_d(\mathbf{p}) = \frac{1}{n}\left\|\mathsf{N}\mathbf{p} - \mathbf{z}\right\|^2, \tag{4.14}$$

where $\mathsf{N} \in \mathbb{R}^{n\times p}$ is the matrix such that $\mathsf{N}^{\mathsf{T}} = \begin{bmatrix}\mathbf{n}_{\mathbf{q}_1} & \ldots & \mathbf{n}_{\mathbf{q}_n}\end{bmatrix}$ and $\mathbf{z} \in \mathbb{R}^n$ is obtained by stacking the depth of the data points, *i.e.* $\mathbf{z}^{\mathsf{T}} = \begin{pmatrix}z_1 & \ldots & z_n\end{pmatrix}$. The bending term can also be written as the squared norm of some vector. One trivial way of doing it would be to approximate the bending energy term of equation (4.11) by discretizing the integral over a regular grid:

$$\mathcal{E}_r(\mathbf{p}) \approx \frac{1}{ab}\sum_{i=0}^{a-1}\sum_{j=0}^{b-1}\sum_{d=0}^{2}\binom{2}{d}\left(\frac{\partial^2 f(\frac{i}{a}, \frac{j}{b}; \mathbf{p})}{\partial x^{2-d}\partial y^d}\right)^2. \tag{4.15}$$

Since the second derivatives of a B-spline are also linear with respect to the control points, the approximation of equation (4.15) is a sum of squared terms linear with respect to the parameters. Even though this approach is effective in practice, there is a better way to write the bending energy term as a squared norm of a vector. Indeed, we showed that it was possible to get an exact formula for the bending energy of the form:

$$\mathcal{E}_r(\mathbf{p}) = \left\|\mathsf{B}\mathbf{p}\right\|^2, \tag{4.16}$$

where $\mathsf{B} \in \mathbb{R}^{p\times p}$ is a matrix that we call the *bending matrix*. The details of the computations of the bending matrix will be given in section 4.2.1.2. For now, let us just assume that the matrix B exists. Given all the previous elements, the initial minimization problem of equation (4.9) is equivalent to the following linear least-squares minimization problem in matrix form:

$$\min_{\mathbf{p}} \left\| \begin{bmatrix} \mathsf{N} \\ \frac{n\lambda}{1-\lambda}\mathsf{B} \end{bmatrix} \mathbf{p} - \begin{bmatrix} \mathbf{z} \\ \mathbf{0} \end{bmatrix} \right\|^2 . \tag{4.17}$$

The solution to this problem is given by using, for instance, the normal equations. For a given regularization parameter $\lambda \in [0, 1[$, we get that:

$$\mathbf{p}^\star_\lambda = \left(\mathsf{N}^\mathsf{T}\mathsf{N} + \left(\frac{n\lambda}{1-\lambda} \right)^2 \mathsf{B}^\mathsf{T}\mathsf{B} \right)^{-1} \mathsf{N}^\mathsf{T}\mathbf{z}. \tag{4.18}$$

4.2.1.2 The Bending Matrix

In this section, we show how the bending matrix B can be derived and practically computed. To the best of our knowledge, the practical computation of the bending matrix cannot be found in the literature. For the sake of simplicity, we first consider the case of a univariate and scalar-valued B-spline. For the same reasons, we also restrict this section to uniform cubic B-splines, which is the flavour of B-spline the most used in this thesis. Let f be the B-spline with knot sequence $k_{-3} < \ldots < k_{n+3}$. We consider the bending energy over the natural definition domain of the B-spline, *i.e.* :

$$b(\mathbf{p}) = \int_{k_0}^{k_n} \left(\frac{\partial^2 f(x; \mathbf{p})}{\partial x^2} \right)^2 \mathrm{d}x. \tag{4.19}$$

The integral bending energy can be divided on each knot interval and, therefore, equation (4.19) can be rewritten as:

$$b(\mathbf{p}) = \sum_{j=0}^{n-1} \int_{k_j}^{k_{j+1}} \left(\frac{\partial^2 f(x; \mathbf{p})}{\partial x^2} \right)^2 \mathrm{d}x. \tag{4.20}$$

As we have seen in section 2.3.2.3, the B-spline f may written as:

$$f(x; \mathbf{p}) = \sum_{i=0}^{3} p_{\iota(x)+i} b_i(o(x)), \tag{4.21}$$

where the four functions b_i (for $i \in [\![0, 3]\!]$) are the pieces of the B-spline basis functions. The functions ι and o were defined in section 2.3.2.3. A similar notation is possible for the derivatives of the B-spline:

$$\frac{\partial^d f}{\partial x^d}(x; \mathbf{p}) = \frac{1}{s^d} \sum_{i=0}^{3} p_{\iota(x)+i} b_{i,d}(o(x)), \tag{4.22}$$

where s is the width of a knot interval (*i.e.* $s = k_{i+1} - k_i$ for all $i \in [\![-3, n+2]\!]$). The function $b_{i,d}$ is the derivative of the d-th order of the function b_i (for $i \in [\![0, 3]\!]$). In particular, for the second derivatives ($d = 2$), we have that:

$$b_{0,2}(x) = -x + 1, \tag{4.23}$$

$$b_{1,2}(x) = 3x - 2, \tag{4.24}$$

$$b_{2,2}(x) = -3x + 1, \tag{4.25}$$

$$b_{3,2}(x) = x. \tag{4.26}$$

Reminder. Here is a formula which will be useful for the computation of the bending matrix: the integration by substitution. It is given by:

$$\int_{\phi(a)}^{\phi(b)} f(x)\mathrm{d}x = \int_a^b f(\phi(t))\phi'(t)\mathrm{d}t. \tag{4.27}$$

Actual computation. We now compute $e_j(\mathbf{p})$ *i.e.* the bending energy of a B-spline over the knot interval $[k_j, k_{j+1}]$.

$$\begin{aligned}
e_j(\mathbf{p}) &= \int_{k_j}^{k_{j+1}} \left(\frac{\partial^2 f}{\partial x^2}(x;\mathbf{p})\right)^2 \mathrm{d}x & (4.28)\\
&= \frac{1}{s^4}\int_{k_j}^{k_{j+1}} \left(\sum_{i=0}^{3} p_{j+i} b_{i,2}(o(x))\right)^2 \mathrm{d}x & (4.29)\\
&= \frac{1}{s^3}\int_{k_j}^{k_{j+1}} \left(\sum_{i=0}^{3} p_{j+i} b_{i,2}(o(x))\right)^2 \frac{1}{s}\mathrm{d}x & (4.30)\\
&= \frac{1}{s^3}\int_{o(k_j)}^{o(k_{j+1})} \left(\sum_{i=0}^{3} p_{j+i} b_{i,2}(x)\right)^2 \mathrm{d}x & (4.31)\\
&= \frac{1}{s^3}\int_{0}^{1} \left(\sum_{i=0}^{3} p_{j+i} b_{i,2}(x)\right)^2 \mathrm{d}x & (4.32)\\
&= \frac{1}{s^3}\int_{0}^{1} \left(\begin{pmatrix} x & \frac{1}{3} & 0 & 0\end{pmatrix}\begin{pmatrix} -1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{pmatrix}\begin{pmatrix} p_j\\ p_{j+1}\\ p_{j+2}\\ p_{j+3}\end{pmatrix}\right)^2 \mathrm{d}x & (4.33)\\
&= \frac{1}{s^3}\int_{0}^{1} \left(\begin{pmatrix} x & \frac{1}{3} & 0 & 0\end{pmatrix}\mathsf{M}\bar{\mathbf{p}}\right)^2 \mathrm{d}x & (4.34)\\
&= \frac{1}{s^3}\int_{0}^{1} \bar{\mathbf{p}}^\mathsf{T}\mathsf{M}^\mathsf{T}\begin{pmatrix} x^2 & \frac{x}{3} & 0 & 0\\ \frac{x}{3} & \frac{1}{9} & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}\mathsf{M}\bar{\mathbf{p}}\mathrm{d}x & (4.35)\\
&= \frac{1}{s^3}\int_{0}^{1} \bar{\mathbf{p}}^\mathsf{T}\begin{pmatrix} -1 & 3\\ 3 & -6\\ -3 & 3\\ 1 & 0\end{pmatrix}\begin{pmatrix} x^2 & \frac{x}{3}\\ \frac{x}{3} & \frac{1}{9}\end{pmatrix}\begin{pmatrix} -1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\end{pmatrix}\bar{\mathbf{p}}\mathrm{d}x & (4.36)\\
&= \frac{1}{s^3}\int_{0}^{1} \bar{\mathbf{p}}^\mathsf{T}\mathsf{M}_2^\mathsf{T}\begin{pmatrix} x^2 & \frac{x}{3}\\ \frac{x}{3} & \frac{1}{9}\end{pmatrix}\mathsf{M}_2\bar{\mathbf{p}}\mathrm{d}x & (4.37)\\
&= \frac{1}{s^3}\bar{\mathbf{p}}^\mathsf{T}\mathsf{M}_2^\mathsf{T}\left(\int_{0}^{1}\begin{pmatrix} x^2 & \frac{x}{3}\\ \frac{x}{3} & \frac{1}{9}\end{pmatrix}\mathrm{d}x\right)\mathsf{M}_2\bar{\mathbf{p}} & (4.38)\\
&= \frac{1}{s^3}\bar{\mathbf{p}}^\mathsf{T}\mathsf{M}_2^\mathsf{T}\left[\begin{pmatrix} x^3/3 & x^2/6\\ x^2/6 & x/9\end{pmatrix}\right]_0^1\mathsf{M}_2\bar{\mathbf{p}} & (4.39)\\
&= \frac{1}{s^3}\bar{\mathbf{p}}^\mathsf{T}\mathsf{M}_2^\mathsf{T}\begin{pmatrix} 1/3 & 1/6\\ 1/6 & 1/9\end{pmatrix}\mathsf{M}_2\bar{\mathbf{p}} & (4.40)\\
&= \frac{1}{s^3}\bar{\mathbf{p}}^\mathsf{T}\frac{1}{6}\begin{pmatrix} 2 & -3 & 0 & 1\\ -3 & 6 & -3 & 0\\ 0 & -3 & 6 & -3\\ 1 & 0 & -3 & 2\end{pmatrix}\bar{\mathbf{p}} & (4.41)\\
&= \frac{1}{s^3}\bar{\mathbf{p}}^\mathsf{T}\bar{\mathsf{B}}\bar{\mathbf{p}} \qquad \text{with } \bar{\mathsf{B}}\begin{pmatrix} 2 & -3 & 0 & 1\\ -3 & 6 & -3 & 0\\ 0 & -3 & 6 & -3\\ 1 & 0 & -3 & 2\end{pmatrix} & (4.42)\\
&= \frac{1}{s^3}\mathbf{p}^\mathsf{T}\mathsf{B}_j\mathbf{p} & (4.43)
\end{aligned}$$

where B_j is the matrix defined as:

$$\mathsf{B}_j = \begin{pmatrix} \mathbf{0}_{j\times j} & \mathbf{0}_{j\times 4} & \mathbf{0}_{j\times(n-j-4)} \\ \mathbf{0}_{4\times j} & \bar{\mathsf{B}} & \mathbf{0}_{4\times(n-j-4)} \\ \mathbf{0}_{(n-j-4)\times j} & \mathbf{0}_{(n-j-4)\times 4} & \mathbf{0}_{(n-j-4)\times(n-j-4)} \end{pmatrix} \tag{4.44}$$

The bending matrix for the bending energy over the entire natural definition domain is obtained by summing the bending matrix for single knot intervals. Therefore, if we define the bending matrix as:

$$\mathsf{B} = \frac{1}{s^3}\sum_{i=0}^{n-1} \mathsf{B}_i, \tag{4.45}$$

then the bending energy is given by:

$$b(\mathbf{p}) = \mathbf{p}^\mathsf{T}\mathsf{B}\mathbf{p} = \left\| \mathsf{B}^{1/2}\mathbf{p} \right\|^2. \tag{4.46}$$

Note that in practice, one generally does not need to explicitly compute a square root of the matrix B. This stems from the fact that it is usually the form $\mathbf{p}^\mathsf{T}\mathsf{B}\mathbf{p}$ which is used in practical computations.

The bending matrix is interesting for several reasons. Of course, it is nice to have an exact formula of the bending energy that can easily be integrated into a linear least-squares optimization problem. Besides, the bending matrix as defined in equation (4.45) is a sparse matrix. Indeed, B is an heptadiagonal symmetric matrix. This sparsity structure allows one to use efficient optimization algorithms. The last interesting property of this bending matrix is that it is usually much smaller that the matrix that would arise by using the approximation of equation (4.15). Indeed, B is a matrix of $\mathbb{R}^{(n+3)\times(n+3)}$. With the approximate formula, the resulting matrix has usually much more rows (it depends on the fineness of the grid used for the discretization of the integral).

The bivariate case. We now derive an exact formula for the bending energy term for bivariate spline. This derivation follows the same principle as the one used for the univariate case. Let f be the uniform cubic tensor-product bi-variate B-spline with knots $k^x_{-3} < \ldots < k^x_{m+3}$ along the x direction and $k^y_{-3} < \ldots < k^y_{n+3}$ along the y direction. The bending energy, denoted e, is defined by:

$$e(\mathbf{p}) = \int_{k_0^y}^{k_n^y}\int_{k_0^x}^{k_m^x} \left(\frac{\partial^2 f}{\partial x^2}(x,y;\mathbf{p})\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}(x,y;\mathbf{p})\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}(x,y;\mathbf{p})\right)^2 \mathrm{d}x\mathrm{d}y. \tag{4.47}$$

As in the univariate case, the bending energy term over the entire natural definition domain can be split on each knot domain:

$$e(\mathbf{p}) = \sum_{b=0}^{n-1}\sum_{a=0}^{m-1} e_{a,b}(\mathbf{p}), \tag{4.48}$$

where $e_{a,b}(\mathbf{p})$ is the bending energy of the B-spline over the knot domain $[k_a^x, k_{a+1}^x] \times [k_b^y, k_{b+1}^y]$. The bending energy $e_{a,b}(\mathbf{p})$ can be further split in three terms:

$$\begin{aligned} e_{a,b}(\mathbf{p}) = \int_{k_b^y}^{k_{b+1}^y} \int_{k_a^x}^{k_{a+1}^x} \left(\frac{\partial^2 f}{\partial x^2}(x,y) \right)^2 \mathrm{d}x\mathrm{d}y \\ + 2 \int_{k_b^y}^{k_{b+1}^y} \int_{k_a^x}^{k_{a+1}^x} \left(\frac{\partial^2 f}{\partial x \partial y}(x,y) \right)^2 \mathrm{d}x\mathrm{d}y \\ + \int_{k_b^y}^{k_{b+1}^y} \int_{k_a^x}^{k_{a+1}^x} \left(\frac{\partial^2 f}{\partial y^2}(x,y) \right)^2 \mathrm{d}x\mathrm{d}y \end{aligned} \quad (4.49)$$

We will give the details of the computation only for the first term in equation (4.49) since the two other terms are computed in an almost identical way. But before that, we define a few other notation. As it was explained in section 2.3.2.5, the bivariate B-spline can be written as:

$$f(x,y;\mathbf{p}) = \left(\mathbf{r}_{o(y)}^\mathsf{T} \otimes \mathbf{r}_{o(x)}^\mathsf{T} \right) (\mathsf{M}_G \otimes \mathsf{M}_G) \mathrm{vect}(\bar{\mathsf{P}}), \quad (4.50)$$

where $\bar{\mathsf{P}} = \mathsf{P}_{\iota(x):\iota(x)+3,\iota(y):\iota(y)+3}$, M_G is the geometric matrix (defined in equation (2.99)), and $\mathbf{r}_{o(x)}$ is the vector of $\mathbb{R}^4$ defined as a function of the free variable x by:

$$\mathbf{r}_{o(x)}^\mathsf{T} = \begin{pmatrix} o(x)^3 & o(x)^2 & o(x) & 1 \end{pmatrix}. \quad (4.51)$$

The partial derivatives of the bi-variate spline can be written in a similar way:

$$\frac{\partial^{c+d} f}{\partial x^c \partial y^d}(x,y;\mathbf{p}) = \left(\mathbf{r}_{o(y),d}^\mathsf{T} \otimes \mathbf{r}_{o(x),c}^\mathsf{T} \right) (\mathsf{M}_G \otimes \mathsf{M}_G) \, \mathrm{vect}(\bar{\mathbf{p}}), \quad (4.52)$$

where $\mathbf{r}_{o(x),c}$ is the derivative of the cth order of the vector $r_{o(x)}$. Along the x direction, these vectors are given by:

$$\mathbf{r}_{o(x),0} = \mathbf{r}_{o(x)}, \quad (4.53)$$
$$\mathbf{r}_{o(x),1} = \frac{1}{s_x} \begin{pmatrix} 3o(x)^2 & 2o(x) & 1 & 0 \end{pmatrix}, \quad (4.54)$$
$$\mathbf{r}_{o(x),2} = \frac{1}{s_x^2} \begin{pmatrix} 6o(x) & 2 & 0 & 0 \end{pmatrix}, \quad (4.55)$$
$$\mathbf{r}_{o(x),3} = \frac{1}{s_x^3} \begin{pmatrix} 6 & 0 & 0 & 0 \end{pmatrix}, \quad (4.56)$$

where s_x is the width of a knot interval along the x direction (*i.e.* $s_x = k_{i+1}^x - k_i^x$ for all $i \in [\![-3, m-2]\!]$). These formulas are identical for the y direction expect for s_x which is replaced by s_y, the width of a knot interval along the y direction.

Actual computation.

First term of $e_{a,b}(\mathbf{p})$.

$$\int_{k_b^y}^{k_{b+1}^y} \int_{k_a^x}^{k_{a+1}^x} \left(\frac{\partial^2 f}{\partial x^2}(x,y;\mathbf{p}) \right)^2 \mathrm{d}x\mathrm{d}y$$

$$\begin{aligned}
&= \int_{k_b^y}^{k_{b+1}^y}\int_{k_a^x}^{k_{a+1}^x} \left(\sum_{j=0}^{3}\sum_{i=0}^{3} p_{a+i,b+j}\frac{1}{s_x^2} b_{i,2}(o(x)) b_i(o(y))\right)^2 \mathrm{d}x\mathrm{d}y \\
&= \frac{1}{s_x^3}\int_{k_b^y}^{k_{b+1}^y}\int_{k_a^x}^{k_{a+1}^x} \left(\sum_{j=0}^{3}\sum_{i=0}^{3} p_{a+i,b+j} b_{i,2}(o(x)) b_i(o(y))\right)^2 \frac{1}{s_x}\mathrm{d}x\mathrm{d}y \\
&= \frac{1}{s_x^3}\int_{k_b^y}^{k_{b+1}^y}\int_{o(k_a^x)}^{o(k_{a+1}^x)} \left(\sum_{j=0}^{3}\sum_{i=0}^{3} p_{a+i,b+j} b_{i,2}(x) b_i(o(y))\right)^2 \mathrm{d}x\mathrm{d}y \\
&= \frac{1}{s_x^3}\int_{k_b^y}^{k_{b+1}^y}\int_{0}^{1} \left(\sum_{j=0}^{3}\sum_{i=0}^{3} p_{a+i,b+j} b_{i,2}(x) b_i(o(y))\right)^2 \mathrm{d}x\mathrm{d}y \\
&= \frac{1}{s_x^3}\int_{k_b^y}^{k_{b+1}^y}\int_{0}^{1} \left(\sum_{j=0}^{3}\sum_{i=0}^{3} p_{a+i,b+j} b_{i,2}(x) b_i(o(y))\right)^2 \mathrm{d}x\frac{s_y}{s_y}\mathrm{d}y \\
&= \frac{s_y}{s_x^3}\int_{0}^{1}\int_{0}^{1} \left(\sum_{j=0}^{3}\sum_{i=0}^{3} p_{a+i,b+j} b_{i,2}(x) b_i(y)\right)^2 \mathrm{d}x\mathrm{d}y \\
&= \frac{s_y}{s_x^3}\int_{0}^{1}\int_{0}^{1} \left(\begin{pmatrix} y^3 & y^2 & y & 1\end{pmatrix} \otimes \begin{pmatrix} 6x & 2 & 0 & 0\end{pmatrix}\mathsf{M}\mathbf{p}\right)^2 \mathrm{d}x\mathrm{d}y \\
&= \frac{s_y}{s_x^3}\int_{0}^{1}\int_{0}^{1} \left(\mathbf{x}_{xx}^{\mathsf{T}}\mathsf{M}\mathbf{p}\right)^2 \mathrm{d}x\mathrm{d}y \\
&= \frac{s_y}{s_x^3}\int_{0}^{1}\int_{0}^{1} \mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\mathbf{x}_{xx}\mathbf{x}_{xx}^{\mathsf{T}}\mathsf{M}\mathbf{p}\ \mathrm{d}x\mathrm{d}y \\
&= \frac{s_y}{s_x^3}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\left(\int_{0}^{1}\int_{0}^{1} \mathbf{x}_{xx}\mathbf{x}_{xx}^{\mathsf{T}}\mathrm{d}x\mathrm{d}y\right)\mathsf{M}\mathbf{p} \\
&= \frac{s_y}{s_x^3}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\left(\int_{0}^{1} \mathbf{r}_y\mathbf{r}_y^{\mathsf{T}} \otimes \left(\int_{0}^{1}\mathbf{r}_{x,2}\mathbf{r}_{x,2}^{\mathsf{T}}\mathrm{d}x\right)\mathrm{d}y\right)\mathsf{M}\mathbf{p} \\
&= \frac{s_y}{s_x^3}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\left(\left(\int_{0}^{1} \mathbf{r}_y\mathbf{r}_y^{\mathsf{T}}\mathrm{d}y\right) \otimes \left(\int_{0}^{1}\mathbf{r}_{x,2}\mathbf{r}_{x,2}^{\mathsf{T}}\mathrm{d}x\right)\right)\mathsf{M}\mathbf{p} \\
&= \frac{s_y}{s_x^3}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\left(\begin{pmatrix} \frac{1}{7} & \frac{1}{6} & \frac{1}{5} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{5} & \frac{1}{4} & \frac{1}{3} \\ \frac{1}{5} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 \end{pmatrix} \otimes \begin{pmatrix} 12 & 6 & 0 & 0 \\ 6 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right)\mathsf{M}\mathbf{p} \\
&= \frac{s_y}{s_x^3}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\mathsf{B}_{xx}\mathsf{M}\mathbf{p}
\end{aligned}$$

Second and third terms of $e_{a,b}(\mathbf{p})$. The second and the third terms of $e_{a,b}(\mathbf{p})$ are computed in a similar way as the first term. We obtain the following formulae:

$$\text{Second term:}\quad \frac{1}{s_x s_y}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\mathsf{B}_{xy}\mathsf{M}\mathbf{p} \tag{4.57}$$

$$\text{Third term:}\quad \frac{s_x}{s_y^3}\mathbf{p}^{\mathsf{T}}\mathsf{M}^{\mathsf{T}}\mathsf{B}_{yy}\mathsf{M}\mathbf{p} \tag{4.58}$$

Bending energy over a single knot domain. Consequently, the exact formula for the bending energy over the knot domain $[k_a^x, k_{a+1}^x] \times [k_b^y, k_{b+1}^y]$ is:

$$e_{a,b}(\mathbf{p}) = \mathbf{p}^\mathsf{T} \mathsf{M} \left(\frac{s_y}{s_x^3} \mathsf{B}_{xx} + \frac{1}{s_x s_y} \mathsf{B}_{xx} + \frac{s_x}{s_y^3} \mathsf{B}_{yy} \right) \mathsf{M} \mathbf{p} \tag{4.59}$$

$$= \mathbf{p}^\mathsf{T} \mathsf{B}_{a,b} \mathbf{p}. \tag{4.60}$$

Bending energy over the entire natural definition domain. Finally, the bending energy for the entire natural definition domain is:

$$e(\mathbf{p}) = \sum_{b=0}^{n-1} \sum_{a=0}^{m-1} e_{a,b}(\mathbf{p}) \tag{4.61}$$

$$= \mathbf{p}^\mathsf{T} \left(\sum_{b=0}^{n-1} \sum_{a=0}^{m-1} \mathsf{B}_{a,b} \right) \mathbf{p} \tag{4.62}$$

$$= \mathbf{p}^\mathsf{T} \mathsf{B} \mathbf{p} \tag{4.63}$$

In the bivariate case, the bending matrix B has similar properties than the bending matrix in the univariate case. In particular, the matrix B is still sparse ; although its sparsity pattern is a bit more complicated. Indeed, it is made of 7 diagonals of blocks each one of which being an heptadiagonal matrix. This is illustrated in figure 4.12.

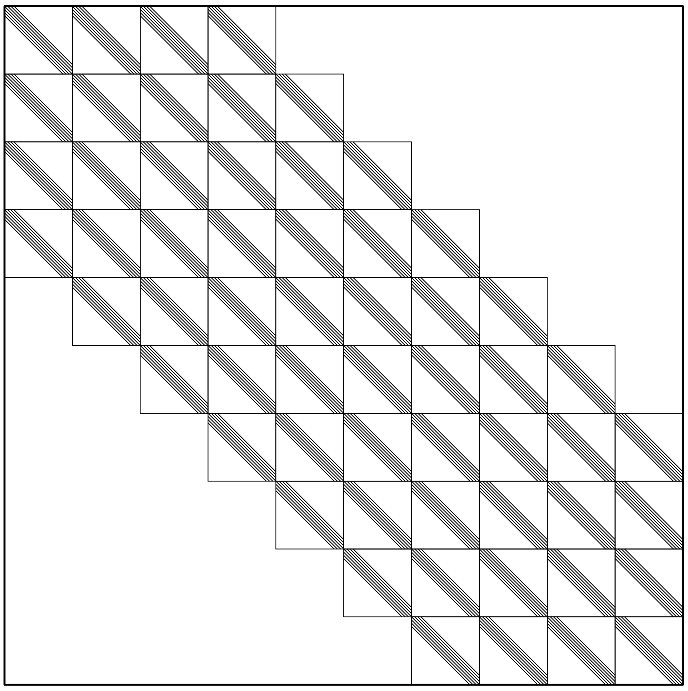


Figure 4.12: Sparsity structure of the bending matrix of bi-variate B-splines. Note that all the blocks are heptadiagonal matrices but they are not all identical to each others. The entire bending matrix is symmetric as are the individual heptadiagonal blocks.

4.2.2 The L-Curve Criterion

The L-curve criterion is a criterion used to automatically determine a regularization parameter in an inverse problem. It was first introduced in (Lawson and Hanson, 1974). This criterion is not as generic as the criteria reviewed in section 3.2.2. Indeed, it is specifically designed to determine only one type of hyperparameter (a regularization parameter) in the special context of a least-squares cost function. The interested reader may find an extensive review of the L-curve criterion in, for instance, (Hansen, 1992, 2005; Mc Carthy, 2003; Rodriguez and Theis, 2005).

The underlying idea of the L-curve criterion is to find a compromise between underfitting and overfitting. We remind the reader that $\mathbf{p}_\lambda^\star$ is the vector of the control points solution to the problem (4.17) for a given regularization parameter λ. Let $\rho(\lambda) = \|\mathsf{N}\mathbf{p}_\lambda^\star - \mathbf{z}\|^2$ be the *residual norm* and let $\eta(\lambda) = \|\mathsf{B}^{1/2}\mathbf{p}_\lambda^\star\|^2$ be the *solution norm*[7]. In an underfitting situation, the solution norm is expected to be small while the residual norm is likely to be large. On the contrary, in an overfitting situation, the solution norm is probably large and the residual norm is expected to be small. To find a compromise between these two pathological situation, the principle of the L-curve criterion is to plot the residual norm and the solution against each other. More precisely, it is the logarithm of the residual norm and of the solution norm that are plotted against each others. This allows one to be invariant to the scale. The resulting curve is called the L-curve. If we note $\hat{\rho}$ and $\hat{\eta}$ the two functions such that $\hat{\rho}(\lambda) = \log(\rho(\lambda))$ and $\hat{\eta}(\lambda) = \log(\eta(\lambda))$, then the L-curve is the set:

$$\left\{(\hat{\rho}(\lambda), \hat{\eta}(\lambda)) \in \mathbb{R}_+^2 | \lambda \in [0, 1[\right\}. \tag{4.64}$$

The name L-curve comes from the usual shape of this curve. Indeed, it often resembles to the letter L. The extremities of the L correspond to the underfitting and overfitting situations. The compromise selected with the L-curve criterion corresponds to the corner of the L. This corner has often been selected manually by actually plotting the L-curve. It is possible to make this approach automatic by saying that the corner of the L is the point of the curve which as the highest curvature. Let κ be the curvature of the L-curve:

$$\kappa(\lambda) = 2\frac{\hat{\rho}'\hat{\eta}'' - \hat{\rho}''\hat{\eta}'}{(\hat{\rho}'^2 + \hat{\eta}'^2)^{3/2}}, \tag{4.65}$$

where the symbols $'$ and $''$ denote respectively the first and the second derivatives of the functions to which it is applied. The L-curve criterion is thus defined as:

$$\lambda^* = \arg \max_{\lambda \in [0,1[} \kappa(\lambda). \tag{4.66}$$

An example of L-curve is shown in figure 4.13. Figure 4.13 also shows the typical aspect of the L-curve criterion. In this example, the L-curve has actually the shape of the letter L. In this case, the L-curve criterion is well defined in the sense that its maximum indeed corresponds to a good regularization parameter. However, selecting a regularization parameter with the L-curve criterion is not always that simple. Indeed, the L-curve is often not well defined in the sense that there are a lot of local maxima. This is a problem if we want a fully automatic approach to choose the regularization parameter. Besides, when the values of the maxima are close to each other, it is not always clear to decide which local maxima is the best (it is not necessarily *the* highest one: it can be *one* of the highest). This pathological situation is illustrated in figure 4.14.

4.2.3 The L-Tangent Norm Criterion

The L-Tangent Norm (LTN) is a new criterion that we have designed to choose the regularization parameter in range surface fitting. It was partly inspired by the idea of 'stability' developed in (Poggio et al., 2004). It also relies on the L-curve.

[7]The word *norm* is a bit abusive here: $\|\mathsf{B}^{1/2}\mathbf{p}\|$ is not an actual norm of $\mathbf{p}$ since the matrix $\mathsf{B}^{1/2}$ is rank deficient.

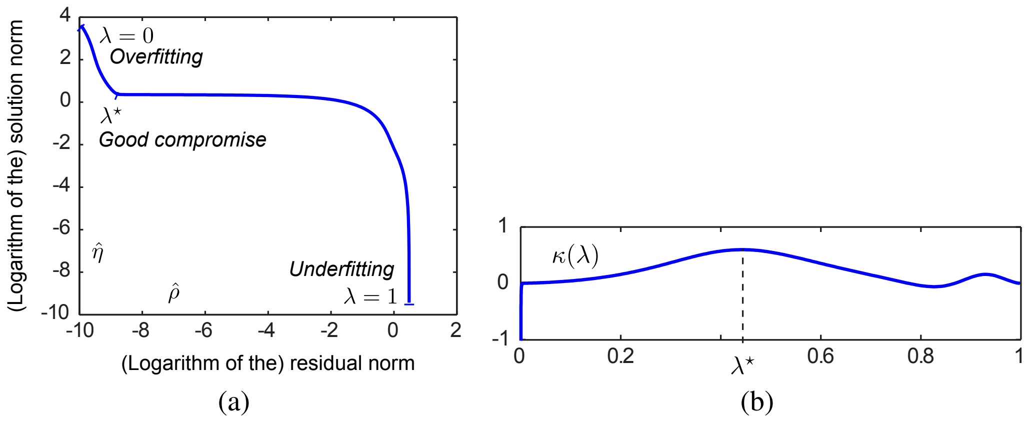


Figure 4.13: (a) An example of L-curve that has the typical shape of the letter L. The extremities of the L corresponds to the worst cases: underfitting and overfitting. (b) Curvature of the L-curve shown in (a). The highest curvature corresponds to the regularization parameter determined with the L-curve criterion.

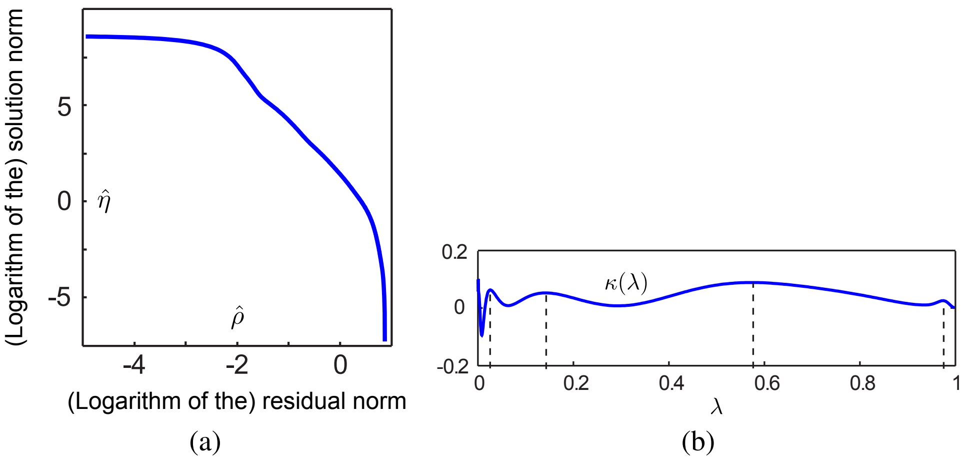


Figure 4.14: An example of pathological case for the L-curve criterion.

4.2.3.1 The Proposed Criterion

One thing that can easily be noticed when dealing with L-curves is that their parametrization is not uniform. In particular, one can observe that there exists a range of values for λ where the tangent vector norm is significantly smaller than elsewhere. Our new criterion is based on this observation. The regularization parameter is chosen as the one for which the L-curve tangent norm is minimal. Intuitively, such a regularization parameter is the one for which a small variation of the regularization parameter has the lowest impact in the trade-off between the goodness of fit and the surface smoothness.

The L-Tangent Norm (LTN) criterion can be written as:

$$\lambda^* = \arg\min_{]0,1[} L(\lambda) \tag{4.67}$$

$$\text{with } L(\lambda) = \left\| \left(\overline{\eta}'_\lambda, \overline{\rho}'_\lambda \right) \right\|_2^2 . \tag{4.68}$$

$\overline{\rho}'_\lambda$ and $\overline{\eta}'_\lambda$ are the derivatives with respect to λ of the normalized residual and solution norms:

$$\overline{\rho}_\lambda = \frac{\rho_\lambda - \rho_\varepsilon}{\rho_{1-\varepsilon} - \rho_\varepsilon}, \qquad \overline{\eta}_\lambda = \frac{\eta_\lambda - \eta_{1-\varepsilon}}{\eta_\varepsilon - \eta_{1-\varepsilon}} \tag{4.69}$$

for ε a small positive constant (10^{-6}, for instance).

4.2.3.2 Properties of the L-Tangent Norm

A typical example of the L-tangent norm criterion is shown in figure 4.15. Even if our criterion is not convex, it is continuous and smooth enough to make it interesting from the optimization point of view. Moreover, neglecting the values of λ very close to 1, our criterion often has a unique minimum, which is not the case of the L-curve criterion.

It sometimes happens that there are two minima. In such cases, it seems that these two local minima are both meaningful. The smaller one (*i.e.* the global minimum) corresponds to the regularization parameter giving the best of the two 'explanations' of the data. The second one seems to appear when the data contains, for instance, a lot of small oscillations. In this case, it is not clear (even for a human being) whether the surface must interpolate the data or approximate them, considering the oscillations as some kind of noise. This situation is illustrated in figure 4.16.

The evaluation of the LTN criterion requires only the computation of the residual and solution norm derivatives. This makes our new criterion faster to compute than, for instance, cross-validation. In particular, our criterion allows one to improve the computation time when the surface model leads to sparse collocation and regularization matrices (as it is the case with the B-spline model). This is not possible with the cross-validation because the influence matrix is generally not sparse.

Another advantage of the LTN criterion is that it would still be efficient with a non-linear surface model. While cross-validation needs a non-iterative formula to achieve acceptable computational time (which does not necessarily exists for such surface models), our criterion just needs the computation of the residual and the solution norms.

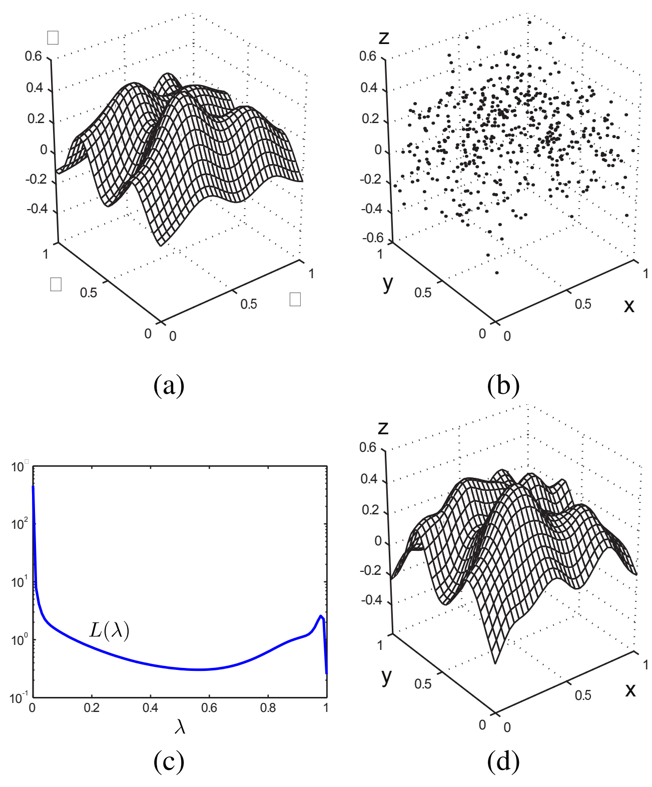


Figure 4.15: Example of the L-Tangent Norm criterion. (a) An initial surface. (b) The initial surface sampled on a set of 500 points with additive normally-distributed noise. (c) The L-Tangent Norm criterion. (d) The reconstructed surface using the optimal regularization parameter found with the LTN.

4.2.4 Experimental Results

4.2.4.1 Data

Synthetic data. The first type of data we have used in these experiments are generated by taking sample points (with added noise) of surfaces defined by:

$$g(x,y) = \sum_{i=1}^{8} \frac{2(1-d)c_1}{5} g_1(x,y) + \frac{dc_2}{5} g_2(x,y) \tag{4.70}$$

$$g_1(x,y) = \exp\left(- \frac{20\left(a_1(x-a_2)^2 + a_3(y-a_4)^2\right)}{a_5} \right) \tag{4.71}$$

$$g_2(x,y) = \sin 4\pi \left(b_1(x+2b_2)^{\frac{1}{2}+b_3} + b_4(y+2b_5)^{\frac{1}{2}+b_6} \right) \tag{4.72}$$

where $a_1, \ldots, a_5, b_1, \ldots, b_6, c_1, c_2$ are randomly chosen in $[0,1]$ and where d is randomly chosen in $\{0,1\}$. Examples of generated surfaces are given in figure 4.17.

Real data. The second type of data we have used in these experiments are real depth maps obtained by stereo imaging means. Figure 4.18 shows the data. The range images we have used in these experiments are large: their size is approximately 400×600 pixels. Therefore, with the approach presented in this section, it is difficult (even impossible) to reconstruct a surface from the original datasets. This is the reason why the range images have been subsampled over a regular grid of size 30×45. However, the full resolution image is used when comparing a reconstructed surface to the initial dataset.

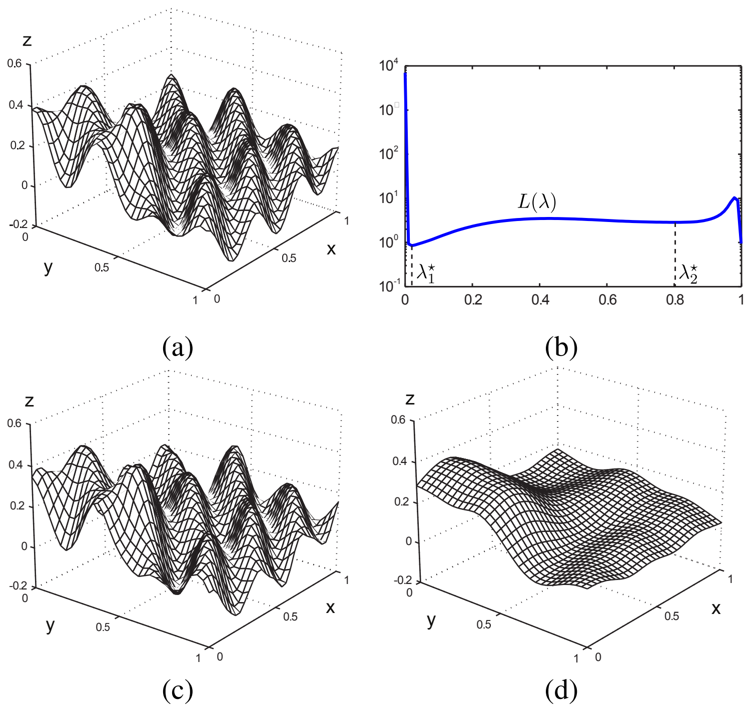


Figure 4.16: An example of the LTN criterion presenting two meaningful minima. (a) An initial surface containing a lot of small oscillations. (b) The L-tangent norm criterion presents two minima (excluding the one reached for λ close to 1). (c) The reconstructed surface using the first minimum ($\lambda_1^* = 0.0189$). (d) The reconstructed surface using the second minimum ($\lambda_2^* = 0.8073$).

4.2.4.2 Computation Timings

Single point evaluation. We intend to compare the computation time of the evaluation for a single value of the regularization parameter of the cross-validation score and the L-tangent norm. To do so, we take a surface and we sample it for several numbers of points. The timings reported in figure 4.19 have been obtained with the cputime function of Matlab and for the (arbitrary) regularization parameter $\lambda = \frac{1}{2}$. Note that the timing for each distinct number of points has been repeated multiple times in order to get stable results. Not surprisingly, figure 4.19 tells us that the evaluation for a single point with the L-tangent norm is far much faster than with cross-validation. This comes from the fact that the inversion of a matrix is needed in the computation of the cross-validation score while only multiplications between sparse matrices and vectors are involved in the L-tangent norm computation.

Optimization of the criterion. In this experiment, we are interested in the computation time of the whole optimization process for both the L-tangent norm and cross-validation. We have taken 300 examples of randomly generated surfaces known through a noisy sampling. The cross-validation optimization process is performed using a golden section search (implemented in the fminbnd function of Matlab). The results are shown in figure 4.20. As in the previous experiment, the optimization of the L-tangent norm is faster than for cross-validation.

Reconstruction of whole surfaces. Figure 4.21 shows the computation times needed to the whole surface reconstruction problem with the three range images presented in figure 4.18. Timings for both the L-tangent norm criterion and the cross-validation score are given in figure 4.21. As expected, using the L-tangent norm is faster than using cross-validation.

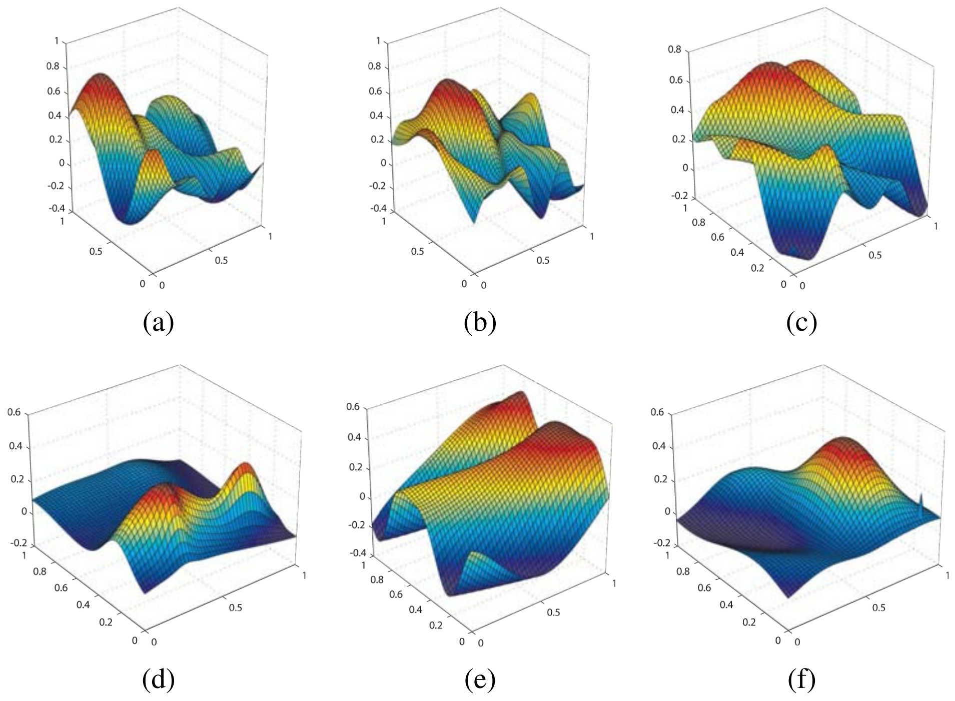


Figure 4.17: Examples of randomly generated surfaces for the experiments.

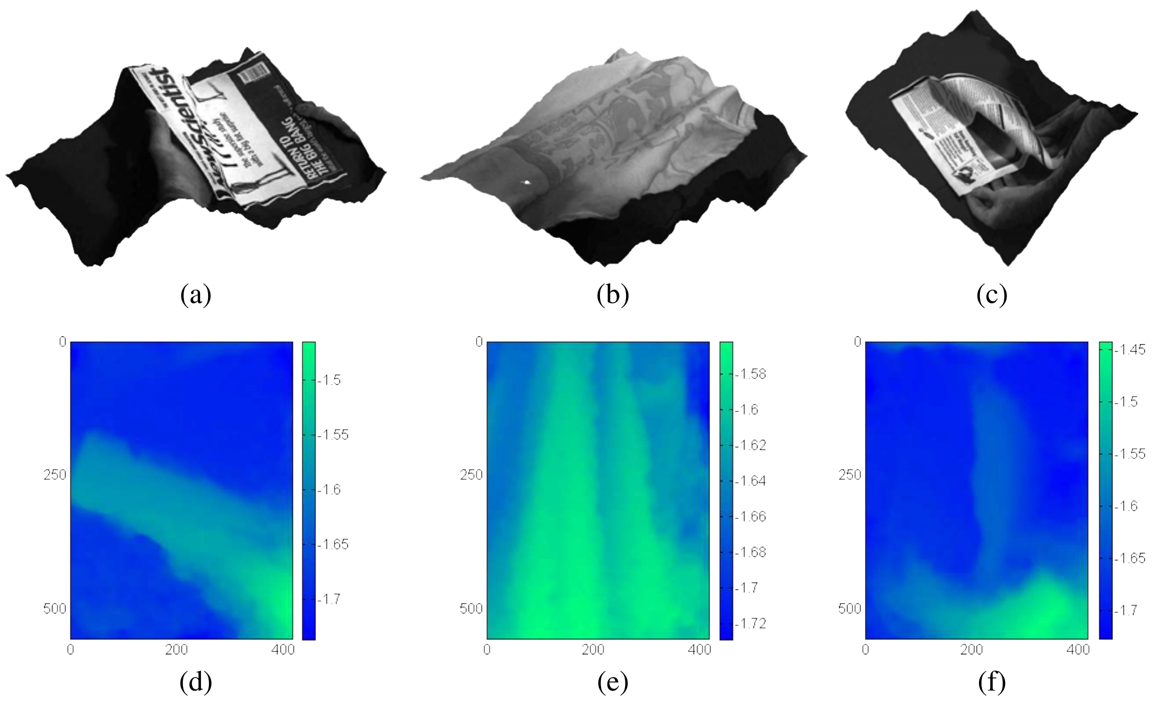


Figure 4.18: Real range data used in the experiments of this section (Courtesy of Toby Collins). Top row: data represented as a textured 3D surface. Bottom row: same data as in the top row but represented with depth maps.

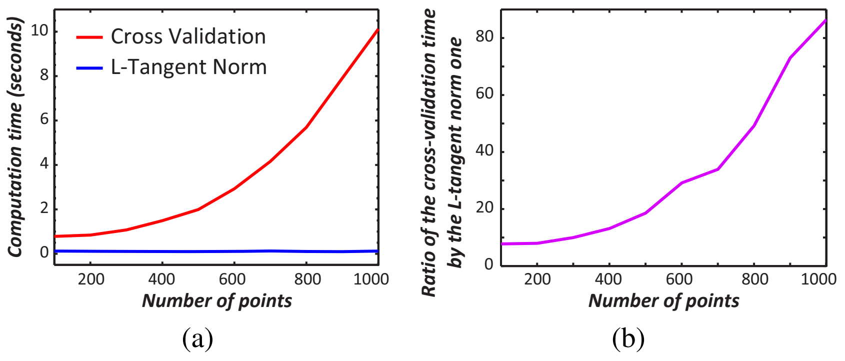


Figure 4.19: Comparison of the cross-validation versus the L-tangent norm computation time for the evaluation of the criteria at a single value. (a) Plot for both cross-validation and the L-Tangent Norm criteria. (b) Ratio of the timings (cross-validation timings divided by the L-Tangent Norm timings).

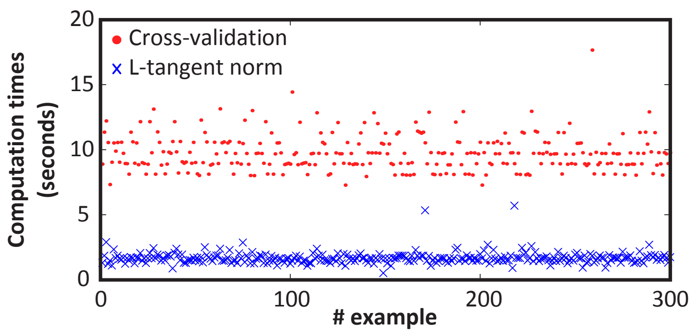


Figure 4.20: Computation time needed to optimize the L-tangent norm and the cross-validation.

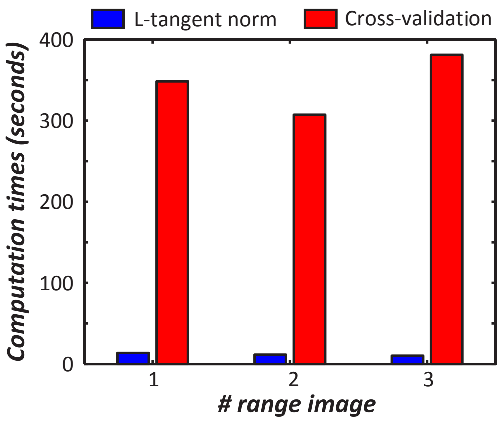


Figure 4.21: Computation time needed to reconstruct the whole surfaces from the range data of figure 4.18 using the L-tangent norm and cross-validation.

4.2.4.3 Is the L-Tangent Norm an Approximation of Cross-Validation?

This experiment aims at comparing the regularization parameter obtained with our L-tangent norm and with the cross-validation criterion. To do so, we have taken noisy samples of randomly generated surfaces. Then, the regularization parameters obtained with cross-validation (the $\lambda_c^\star$) for each dataset are plotted versus the regularization parameter determined with the L-tangent norm (the $\lambda_l^\star$). The results are reported in figure 4.22. We see from figure 4.22 that the regularization parameters obtained with the L-tangent norm are often close to the ones obtained with cross-validation. One can observe that the L-tangent norm tends to slightly under-estimate large regularization parameters. However, large regularization parameters are usually obtained for datasets with a lot of noise or badly constrained. In such cases, the accuracy of the regularization parameter does not matter so much.

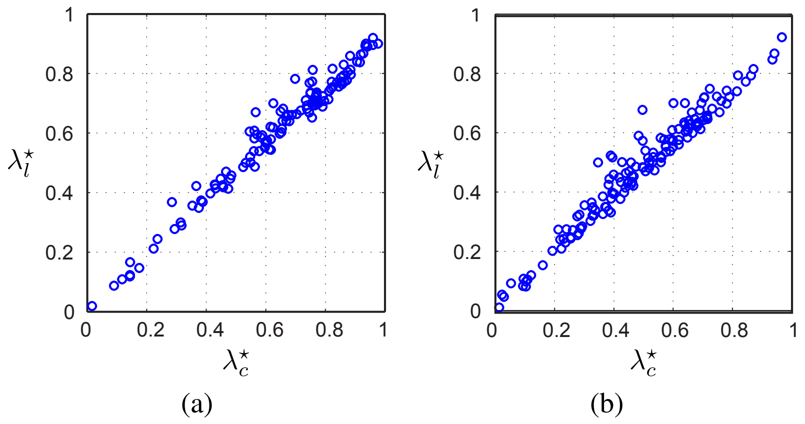


Figure 4.22: Comparison of the regularization parameters obtained with the L-tangent norm ($\lambda_l^\star$) and with the ones obtained with cross-validation ($\lambda_c^\star$). (a) Data with normally-distributed noise. (b) Data with uniformly-distributed noise.

4.2.4.4 Reconstructed Surfaces

Synthetic data. In this experiment, we compare the surfaces reconstructed from data obtained as noisy discretizations of randomly generated surfaces. Let us denote f the original randomly generated surface, f_c, f_l and f_n the surfaces reconstructed using respectively cross-validation, the L-curve criterion and our L-tangent norm. The difference between the original surface and the reconstructed ones is measured with the *Integral Relative Error* (IRE). If the functions f, f_c, f_l and f_n are all defined over the domain Ω, then the IRE is given by:

$$e(f,g) = \frac{\iint_\Omega |g(x,y) - f(x,y)| \,\mathrm{d}x\mathrm{d}y}{\left| \max\limits_{(x,y)\in\Omega} f(x,y) - \min\limits_{(x,y)\in\Omega} f(x,y) \right|}, \tag{4.73}$$

where the function g is either f_c, f_l or f_n. The results of this experiment are reported in figure 4.23. This figure tells us that the reconstruction errors are small and similar for cross-validation and the L-tangent norm. The IRE for surfaces reconstructed using the L-curve criterion are much larger than with the two other criteria. Moreover, only IREs lower than 1 are reported for the L-curve criterion: the IRE was greater than 1 for 48 test surfaces. These large IREs are mainly due to a failure in the maximization of the L-curve criterion.

Range images. In this last experiment, we intend to compare the surfaces reconstructed from real range images. To do so, we take again the three range images of figure 4.18. Let f_i be the original range image (before subsampling). Let f_l and f_c be the reconstructed surfaces using respectively the L-tangent norm and the cross-validation criterion to choose the regularization parameter. The results of this experiment are presented

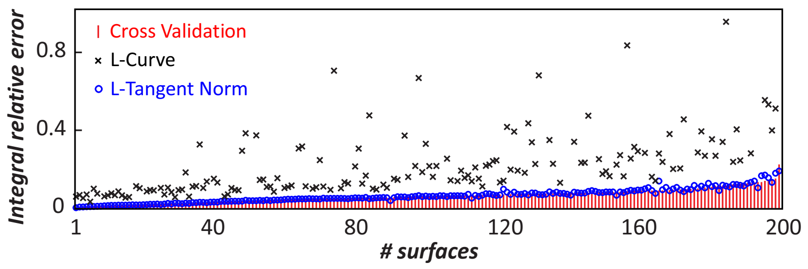


Figure 4.23: Integral relative errors for 200 randomly generated surfaces sampled over 500 points with additive normally-distributed noise.

in the form of *Relative Error Maps* (REM). The REM for the surface reconstructed using the L-tangent norm is a picture such that each pixels (x, y) is associated to a colour $C_l(x, y)$ proportional to the difference of depth between the reconstructed surface and the original one. This is written as:

$$C_l(x,y) = \frac{|f_i(x,y) - f_l(x,y)|}{\left| \max\limits_{(u,v)} f_i(u,v) - \min\limits_{(u,v)} f_i(u,v) \right|}. \tag{4.74}$$

The REM for the surface reconstructed using cross-validation, C_c, is defined similarly to equation (4.74)) except that f_l is replaced by f_c. We also define the *Difference Error Map* (DEM) by $C_{l,c}(x, y) = |C_c(x, y) - C_l(x, y)|$. The results of the comparison between surfaces reconstructed from range images using the L-tangent norm and the cross-validation are reported in figure 4.24. On this figure, only the error map for the L-tangent norm is reported. Indeed, as it is shown in figure 4.24 (d-f), the two reconstructed surfaces are very similar (which is the point of main interest in this experiment). Even if the reconstruction errors are not negligible (figure 4.24 (a-c)), they are still small. The main reason for these errors is the subsampling of the initial datasets.

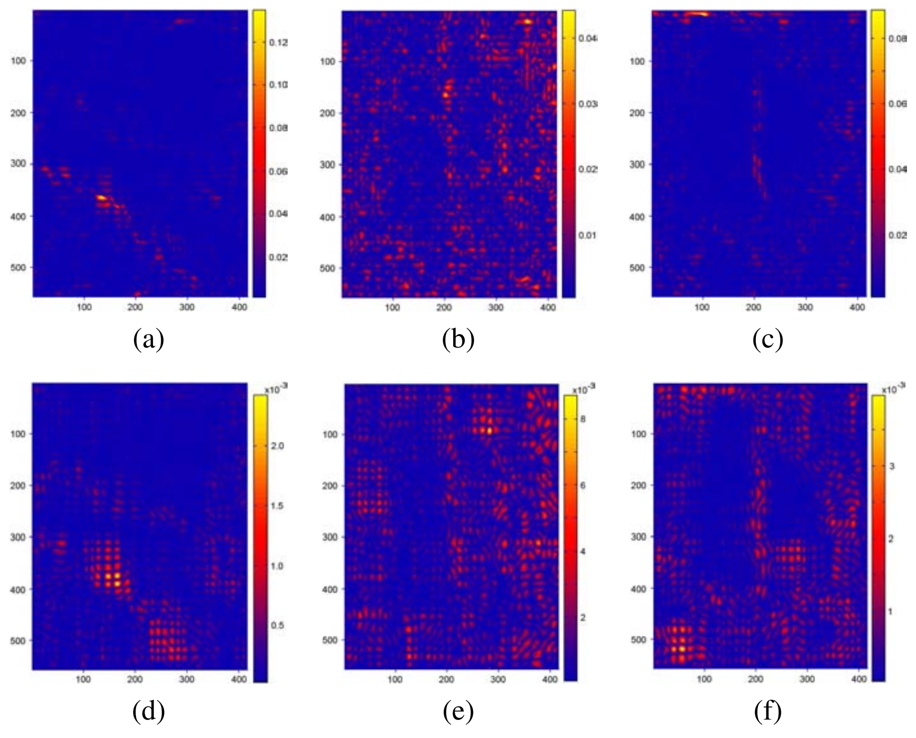


Figure 4.24: (a-c) Relative error maps (REM) for the surfaces reconstructed using the L-tangent norm criterion for the three range images of figure 4.18. (d-f) Difference error maps (DEM) between the surfaces reconstructed using the L-tangent norm criterion and the cross-validation criterion.

4.3 Range Surface Fitting with Heteroskedastic Noise

In this section, we present some other contributions we made in the topic of reconstructing range surface. In the previous sections, we considered arbitrary range data in the sense that the locations of the data points did not follow any particular pattern. In this section, we consider data sets with points arranged on a regular grid. More precisely, we consider data sets in form of depth maps produced by TOF cameras. This type of data is particularly challenging for several reasons. Firstly, there is an important amount of data. With the equipment we used, there was typically around 20 thousands points per depth map. Secondly, there is usually an important amount of noise in the TOF data. Besides, this noise is heteroskedastic which means that the variance of the noise is not identical for all the pixels of the depth map. Thirdly, as TOF cameras can capture data from arbitrary parts of the environment, the resulting depth maps are likely to contain discontinuities. The contributions proposed in this section aim at coping with these different aspects of TOF data.

This section is organized as follows. First, we specialize the general approach to range surface fitting with B-splines to the particular case of data arranged on a regular grid. In this first part, we will just consider an homoskedastic noise. In addition to the presentation of the general way of handling grid data, we make two contributions: a new regularization term that approximate the bending energy while being compatible with the grid approach and a new manner of selecting the regularization parameter. Next, we present our multi-step approach for handling heteroskedastic noise and discontinuities. Finally, the efficiency of our approach is illustrated with a set of experiments.

4.3.1 Fitting a B-spline on Mesh Data

4.3.1.1 Properties of the Kronecker Product

As it will be shown later, fitting a B-spline on mesh data leads to formulas that heavily relies on the Kronecker product. We thus give here a list of interesting properties linked to the Kronecker product. The following identities are taken from (Dierckx, 1993):

$$\mathbf{I}_p \otimes \mathbf{I}_q = \mathbf{I}_{pq} \tag{4.75}$$

$$(\mathsf{A} \otimes \mathsf{B})^\mathsf{T} = \mathsf{A}^\mathsf{T} \otimes \mathsf{B}^\mathsf{T} \tag{4.76}$$

$$(\mathsf{A} \otimes \mathsf{B})^{-1} = \mathsf{A}^{-1} \otimes \mathsf{B}^{-1} \tag{4.77}$$

$$(\mathsf{AB}) \otimes (\mathsf{CD}) = (\mathsf{A} \otimes \mathsf{C})(\mathsf{B} \otimes \mathsf{D}) \tag{4.78}$$

$$(\mathsf{A} + \mathsf{B}) \otimes (\mathsf{C} + \mathsf{D}) = \mathsf{A} \otimes \mathsf{C} + \mathsf{B} \otimes \mathsf{C} + \mathsf{A} \otimes \mathsf{D} + \mathsf{B} \otimes \mathsf{D} \tag{4.79}$$

$$\mathrm{vect}(\mathsf{A}_{p\times s}\mathsf{X}_{s\times t}) = (\mathbf{I}_t \otimes \mathsf{A})\mathrm{vect}(\mathsf{X}) \tag{4.80}$$

$$\mathrm{vect}(\mathsf{X}_{s\times t}\mathsf{B}_{t\times q}) = (\mathsf{B}^\mathsf{T} \otimes \mathbf{I}_s)\mathrm{vect}(\mathsf{X}) \tag{4.81}$$

$$\mathrm{vect}(\mathsf{A}_{p\times s}\mathsf{X}_{s\times t}\mathsf{B}_{t\times q}) = (\mathsf{B}^\mathsf{T} \otimes \mathsf{A})\mathrm{vect}(\mathsf{X}) \tag{4.82}$$

Here comes additional properties that will be useful in the derivations of the principles for fitting a B-spline to mesh data:

$$\mathrm{vect}(\mathsf{A}_{m\times n}) = \mathrm{vect}(\mathsf{B}_{m\times n}) \Leftrightarrow \mathsf{A}_{m\times n} = \mathsf{B}_{m\times n} \tag{4.83}$$

$$\mathrm{trace}(\mathsf{A}_m \otimes \mathsf{B}_n) = \mathrm{trace}(\mathsf{A}_m)\mathrm{trace}(\mathsf{B}_n) \tag{4.84}$$

$$\mathrm{vect}(\mathsf{A}) - \mathrm{vect}(\mathsf{B}) = \mathrm{vect}(\mathsf{A} - \mathsf{B}) \tag{4.85}$$

$$\|\mathrm{vect}(\mathsf{A})\|^2 = \|\mathsf{A}\|_{\mathcal{F}}^2 \tag{4.86}$$

$$\|\mathsf{A}\|_{\mathcal{F}}^2 = \mathrm{trace}(\mathsf{A}^\mathsf{T}\mathsf{A}) \tag{4.87}$$

$$\mathrm{trace}(\mathbf{I}_n - \mathsf{A}_{n\times n}) = n - \mathrm{trace}(\mathsf{A}) \tag{4.88}$$

4.3.1.2 Mesh Data

In this section, we consider that the data set is made of range data points organized as a regular grid of size $m \times n$. This type of data naturally arises with ToF cameras since this device produces depth maps. This particular data structure leads to write the whole data set in the following way:

$$\left\{\mathbf{q}_{i,j} = (x_i, y_j) \leftrightarrow z_{i,j} \mid i \in [\![1, m]\!], j \in [\![1, n]\!]\right\}. \tag{4.89}$$

The depths $z_{i,j}$ are grouped in a matrix $\mathsf{Z} \in \mathbb{R}^{m\times n}$.

4.3.1.3 Fitting a B-spline to Mesh Data

The general principle for fitting a B-spline to mesh range data is the same as the one used in the previous sections. It is still formulated as a minimization problem with a cost function composed of two terms. However, for the needs of this section, this optimization problem is formulated in a slightly different way:

$$\min_{\mathsf{P}} \mathcal{D}(\mathsf{P}) + \mathcal{R}_\lambda(\mathsf{P}), \tag{4.90}$$

where $\mathcal{D}$ and $\mathcal{R}_\lambda$ are respectively the data term and the regularization term. Here, we consider that the regularization parameter λ is part of the regularization term itself.

Data term. The data term is again defined as the mean squared residual:

$$\mathcal{D}(\mathsf{P}) = \frac{1}{mn}\left\|\mathsf{N}\mathrm{vect}(\mathsf{P}) - \mathrm{vect}(\mathsf{Z})\right\|^2. \tag{4.91}$$

The fact that the data are arranged on a grid is exploited in the collocation matrix N. Indeed, since we use uniform cubic B-splines, this matrix is initially defined as:

$$\mathsf{N} = \begin{pmatrix} N_{-3}(x_1)N_{-3}(y_1) & \cdots & N_{-3}(x_1)N_{h-1}(y_1) & \cdots\cdots & N_{g-1}(x_1)N_{-3}(y_1) & \cdots & N_{g-1}(x_1)N_3(y_1) \\ \vdots & & \vdots & & \vdots & & \vdots \\ N_{-3}(x_1)N_{-3}(y_n) & \cdots & N_{-3}(x_1)N_{h-1}(y_n) & \cdots\cdots & N_{g-1}(x_1)N_{-3}(y_n) & \cdots & N_{g-1}(x_1)N_3(y_n) \\ \vdots & & \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots & & \vdots \\ N_{-3}(x_m)N_{-3}(y_1) & \cdots & N_{-3}(x_m)N_{h-1}(y_1) & \cdots\cdots & N_{g-1}(x_m)N_{-3}(y_1) & \cdots & N_{g-1}(x_m)N_3(y_1) \\ \vdots & & \vdots & & \vdots & & \vdots \\ N_{-3}(x_m)N_{-3}(y_n) & \cdots & N_{-3}(x_m)N_{h-1}(y_n) & \cdots\cdots & N_{g-1}(x_m)N_{-3}(y_n) & \cdots & N_{g-1}(x_m)N_3(y_n) \end{pmatrix}. \tag{4.92}$$

From equation (4.92), it is easy to see that the collocation matrix N can be written as the Kronecker product of two matrices:

$$\mathsf{N} = \mathsf{E} \otimes \mathsf{F}, \tag{4.93}$$

where:

$$\mathsf{E} = \begin{pmatrix} N_{-3}(x_1) & \dots & N_{g-1}(x_1) \\ \vdots & & \vdots \\ N_{-3}(x_m) & \dots & N_{g-1}(x_m) \end{pmatrix}, \tag{4.94}$$

and

$$\mathsf{F} = \begin{pmatrix} N_{-3}(y_1) & \dots & N_{h-1}(y_1) \\ \vdots & & \vdots \\ N_{-3}(y_n) & \dots & N_{h-1}(y_n) \end{pmatrix}. \tag{4.95}$$

Consequently, the data term can be written as:

$$\mathcal{D}(\mathsf{P}) = \|(\mathsf{E} \otimes \mathsf{F})\mathrm{vect}(\mathsf{P}) - \mathrm{vect}(\mathsf{Z})\|^2 \tag{4.96}$$

Regularization term. In order to obtain the best efficiency in terms of computational complexity, we propose a new regularization term. It is a variation of the classical bending energy used so far. We propose a new regularization term because the classical bending energy cannot be expressed with a tensor product expression. We define it as:

$$\begin{aligned} \mathcal{R}_\lambda(\mathsf{P}) = \lambda \int_{k_0^y}^{k_h^y}\int_{k_0^x}^{k_g^x} \left(\frac{\partial^2 f}{\partial x^2}(x, y; \mathsf{P})\right)^2 \mathrm{d}x\mathrm{d}y \\ + \lambda \int_{k_0^y}^{k_h^y}\int_{k_0^x}^{k_g^x} \left(\frac{\partial^2 f}{\partial y^2}(x, y; \mathsf{P})\right)^2 \mathrm{d}x\mathrm{d}y \\ + \lambda^2 \int_{k_0^y}^{k_h^y}\int_{k_0^x}^{k_g^x} \left(\frac{\partial^4 f}{\partial x^2 \partial y^2}(x, y; \mathsf{P})\right)^2 \mathrm{d}x\mathrm{d}y. \end{aligned} \tag{4.97}$$

The main advantage of the regularization term of equation (4.97) is that, as the data term, it can be written using the Kronecker product. Indeed, we have that:

$$R_\lambda(\mathsf{P}) = \|\mathsf{R}_\lambda \mathrm{vect}(\mathsf{P})\|^2 \qquad \text{with} \qquad \mathsf{R}_\lambda = \begin{bmatrix} \lambda(\mathsf{S}_x \otimes \mathsf{F}) \\ \lambda(\mathsf{E} \otimes \mathsf{S}_y) \\ \lambda^2(\mathsf{S}_x \otimes \mathsf{S}_y) \end{bmatrix}, \tag{4.98}$$

where S_x and S_y are the square roots of the monodimensional bending matrices along the x and y directions respectively.

Solution to the problem. The main advantage of having a data term and a regularization term that can be written using the Kronecker product of matrices is that it leads to efficient computations. Indeed, given the expressions of the data term and the regularization term, the solution of problem (4.90) is given by:

$$\mathsf{P}^\star = \left(\mathsf{F}^\mathsf{T}\mathsf{F} + \lambda \mathsf{B}_y\right)^{-1} \mathsf{F}^\mathsf{T}\mathsf{Z}\mathsf{E}\left(\mathsf{E}^\mathsf{T}\mathsf{E} + \lambda \mathsf{B}_x\right)^{-1}, \tag{4.99}$$

where $\mathsf{B}_x = \mathsf{S}_x^\mathsf{T}\mathsf{S}_x$ and $\mathsf{B}_y = \mathsf{S}_y^\mathsf{T}\mathsf{S}_y$. The solution given by equation (4.99) is derived with the following reasoning:

$$
\begin{aligned}
&\min_{\mathsf{P}} \left\| \begin{bmatrix} \mathsf{E}\otimes\mathsf{F} \\ \lambda(\mathsf{S}_x\otimes\mathsf{F}) \\ \lambda(\mathsf{E}\otimes\mathsf{S}_y) \\ \lambda^2(\mathsf{S}_x\otimes\mathsf{S}_y) \end{bmatrix} \mathrm{vect}(\mathsf{P}) - \begin{bmatrix} \mathrm{vect}(\mathsf{Z}) \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \right\|^2 \\
\Leftrightarrow\quad & \begin{bmatrix} \mathsf{E}\otimes\mathsf{F} \\ \sqrt{\lambda}\mathsf{S}_x\otimes\mathsf{F} \\ \sqrt{\lambda}\mathsf{E}\otimes\mathsf{S}_y \\ \lambda\mathsf{S}_x\otimes\mathsf{S}_y \end{bmatrix}^\mathsf{T} \begin{bmatrix} \mathsf{E}\otimes\mathsf{F} \\ \sqrt{\lambda}\mathsf{S}_x\otimes\mathsf{F} \\ \sqrt{\lambda}\mathsf{E}\otimes\mathsf{S}_y \\ \lambda\mathsf{S}_x\otimes\mathsf{S}_y \end{bmatrix} \mathrm{vect}(\mathsf{P}) = \begin{bmatrix} \mathsf{E}\otimes\mathsf{F} \\ \sqrt{\lambda}\mathsf{S}_x\otimes\mathsf{F} \\ \sqrt{\lambda}\mathsf{E}\otimes\mathsf{S}_y \\ \lambda\mathsf{S}_x\otimes\mathsf{S}_y \end{bmatrix}^\mathsf{T} \begin{bmatrix} \mathrm{vect}(\mathsf{Z}) \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \\
\Leftrightarrow\quad & \big\{(\mathsf{E}^\mathsf{T}\otimes\mathsf{F}^\mathsf{T})(\mathsf{E}\otimes\mathsf{F}) + \lambda(\mathsf{S}_x^\mathsf{T}\otimes\mathsf{F}^\mathsf{T})(\mathsf{S}_x\otimes\mathsf{F}) + \lambda(\mathsf{E}^\mathsf{T}\otimes\mathsf{S}_y^\mathsf{T})(\mathsf{E}\otimes\mathsf{S}_y) \\
&\qquad + \lambda^2(\mathsf{S}_x^\mathsf{T}\otimes\mathsf{S}_y^\mathsf{T})(\mathsf{S}_x\otimes\mathsf{S}_y)\big\}\mathrm{vect}(\mathsf{P}) = (\mathsf{E}^\mathsf{T}\otimes\mathsf{F}^\mathsf{T})\mathrm{vect}(\mathsf{Z}) \\
\Leftrightarrow\quad & (\mathsf{E}^\mathsf{T}\mathsf{E}\otimes\mathsf{F}^\mathsf{T}\mathsf{F})\mathrm{vect}(\mathsf{P}) + \lambda(\mathsf{S}_x^\mathsf{T}\mathsf{S}_x\otimes\mathsf{F}^\mathsf{T}\mathsf{F})\mathrm{vect}(\mathsf{P}) + \lambda(\mathsf{E}^\mathsf{T}\mathsf{E}\otimes\mathsf{S}_y^\mathsf{T}\mathsf{S}_y)\mathrm{vect}(\mathsf{P}) \\
&\qquad + \lambda^2(\mathsf{S}_x^\mathsf{T}\mathsf{S}_x\otimes\mathsf{S}_y^\mathsf{T}\mathsf{S}_y)\mathrm{vect}(\mathsf{P}) = (\mathsf{E}^\mathsf{T}\otimes\mathsf{F}^\mathsf{T})\mathrm{vect}(\mathsf{P}) \\
\Leftrightarrow\quad & \mathsf{F}^\mathsf{T}\mathsf{F}\mathsf{P}\mathsf{E}^\mathsf{T}\mathsf{E} + \lambda\mathsf{F}^\mathsf{T}\mathsf{F}\mathsf{P}\mathsf{S}_x^\mathsf{T}\mathsf{S}_x + \lambda\mathsf{S}_y^\mathsf{T}\mathsf{S}_y\mathsf{P}\mathsf{E}^\mathsf{T}\mathsf{E} + \lambda^2\mathsf{S}_y^\mathsf{T}\mathsf{S}_y\mathsf{P}\mathsf{S}_x^\mathsf{T}\mathsf{S}_x = \mathsf{F}^\mathsf{T}\mathsf{Z}\mathsf{E} \\
\Leftrightarrow\quad & \left(\mathsf{F}^\mathsf{T}\mathsf{F}\mathsf{P} + \lambda\mathsf{B}_y\mathsf{P}\right)\left(\mathsf{E}^\mathsf{T} + \mathsf{E} + \lambda\mathsf{B}_x\right) = \mathsf{F}^\mathsf{T}\mathsf{Z}\mathsf{E} \\
\Leftrightarrow\quad & \left(\mathsf{F}^\mathsf{T}\mathsf{F} + \lambda\mathsf{B}_y\right)\mathsf{P}\left(\mathsf{E}^\mathsf{T}\mathsf{E} + \lambda\mathsf{B}_x\right) = \mathsf{F}^\mathsf{T}\mathsf{Z}\mathsf{E} \\
\Leftrightarrow\quad & \mathsf{P} = \left(\mathsf{F}^\mathsf{T}\mathsf{F} + \lambda\mathsf{B}_y\right)^{-1}\mathsf{F}^\mathsf{T}\mathsf{Z}\mathsf{E}\left(\mathsf{E}^\mathsf{T}\mathsf{E} + \lambda\mathsf{B}_x\right)^{-1}
\end{aligned}
$$

4.3.1.4 Choice of the Regularization Parameter

At this point, we have an efficient approach to fit a surface to range data (with homoskedastic noise) for a given regularization parameter. In this section, we propose a new approach to determine in an automatic and fast manner the regularization parameter. The contribution we propose here is not completely brand new. In fact, it is an adaptation to our problem of a quite old principle: *Morozov's discrepancy principle* (Morozov, 1966). In this section, we explain the principles of this criterion in the context of range surface fitting. Extended details on the Morozov's principle may be found in (George and Nair, 1994; Pereverzev and Schock, 1999; Schock, 1984; Shiguemori et al., 2004; Tautenhahn and Hämarik, 1999).

Again, we emphasize that for now, we just consider a noise with the same distribution over all the dataset. This stems from the fact that fitting a surface to range data with homoskedastic noise is the basic building block of the whole algorithm that we propose in this section. The case of the heteroskedastic noise will be handle in section 4.3.2. Let us consider that the dataset is a noisy sample of a reference function $\underline{f}$, *i.e.* $z_{i,j} = \underline{f}(j,i) + e_{i,j}$ where the $e_{i,j}$ are random variables that represent the noise. Let us further assume that the noise has standard deviation σ. For a given dataset Z, let $\mathcal{S}$ be the set of all the B-splines control points satisfying equation (4.99) for all possible regularization parameter λ, *i.e.* :

$$\mathcal{S} = \{f(\bullet;\mathsf{P}_\lambda) \mid \lambda \in \mathbb{R}^+\}. \tag{4.100}$$

The surface fitting problem is equivalent to find the function $f \in \mathcal{S}$ such that:

$$f = \arg\min_{s \in \mathcal{S}} \|s - \underline{f}\|, \tag{4.101}$$

where $\|\bullet\|$ is some kind of function norm. The discrepancy principle states that the function f should be the one that results in a standard deviation for the errors ($\mathrm{std}(\mathsf{S} - \mathsf{Z})$ with $s_{i,j} = f(j,i)$) that is as close as possible to the true one (σ). Since $\underline{f}$ is not necessarily a member of $\mathcal{S}$, the strict equality between σ and $\mathrm{std}(\mathsf{S} - \mathsf{Z})$ is not always possible. The regularization parameter is thus chosen as the one that minimizes the following problem:

$$\min_{\lambda \in \mathbb{R}_+} \left(\mathrm{std}(\mathsf{S}^\star(\lambda) - \mathsf{Z})\right)^2, \tag{4.102}$$

with $s_{i,j}(\lambda) = s_\lambda(j,i)$ and s_λ is the B-spline obtained by solving equation (4.99) for a given value of λ. Since the quantity $\mathrm{std}(\mathsf{S}^\star - \mathsf{Z})$ increases with λ, the criterion optimized in equation (4.102) has only one global minimum. Problem (4.102) may be optimized using, for instance, the golden section search algorithm described in section 2.2.2.12.

Of course, the ground truth standard deviation of the errors is generally unknown in practice. In place of that, we propose a simple but efficient approach to estimate the standard deviation of the noise directly from the dataset. It relies on a local linear approximation of the data. Let $\mathcal{P}_{i,j}$ be the function representing the plane obtained by linear regression from the dataset $\{(v,u) \leftrightarrow z_{u,v} \mid u \in [\![i-1, i+1]\!] \text{ and } v \in [\![j-1, j+1]\!]\}$. The approximate standard deviation σ_e for all the dataset Z is given by:

$$\sigma_e^2 = \frac{1}{(m-2)(n-2)} \sum_{i=2}^{m-1} \sum_{j=2}^{n-1} (\mathcal{P}_{i,j}(j,i) - z_{i,j})^2. \tag{4.103}$$

The efficiency of this standard deviation estimator for range data will be tested in the experimental part of this section (see section 4.3.3.1). Note that this estimator is clearly more efficient with data that represent a smooth surface than with data that have discontinuities.

4.3.1.5 The FGA Algorithm

The combination of equation (4.99) and of the discrepancy principle (including the standard deviation estimator) is called the *Fast Grid Approach* algorithm (FGA). Note that this algorithm is designed to work on datasets with homoskedastic noise. It will be used as a basic building block of our more general approach to handle heteroskedastic data.

4.3.2 Our Approach to Handle Heteroskedastic Noise and Discontinuities

In section 4.3.1, we presented the FGA method which allows one to quickly fit a B-spline to range data organized as a regular mesh with homoskedastic noise. In this section, we present our approach to fit a B-spline on range data with heteroskedastic noise and discontinuities. The approach we propose to handle such data is a multi-step approach. The different steps of the algorithm are summarized below and detailed after that.

Segmentation. The dataset is segmented in order to first, separate the different level of noise and second, get rid of the discontinuities.

Bounding boxes. The data of each segment is embedded in a rectangular domain (the *bounding box*) so that the FGA algorithm can be used.

Local depth maps. A *local depth map* is created for each segment by filling the corresponding bounding box.

Local fittings. A surface is fitted to each one of the local depth maps using the FGA algorithm.

Merging. A single global surface is finally constructed from the local fittings using the fact that the control points of a B-spline have only a localized influence.

Figure 4.25 illustrates the steps of our approach.

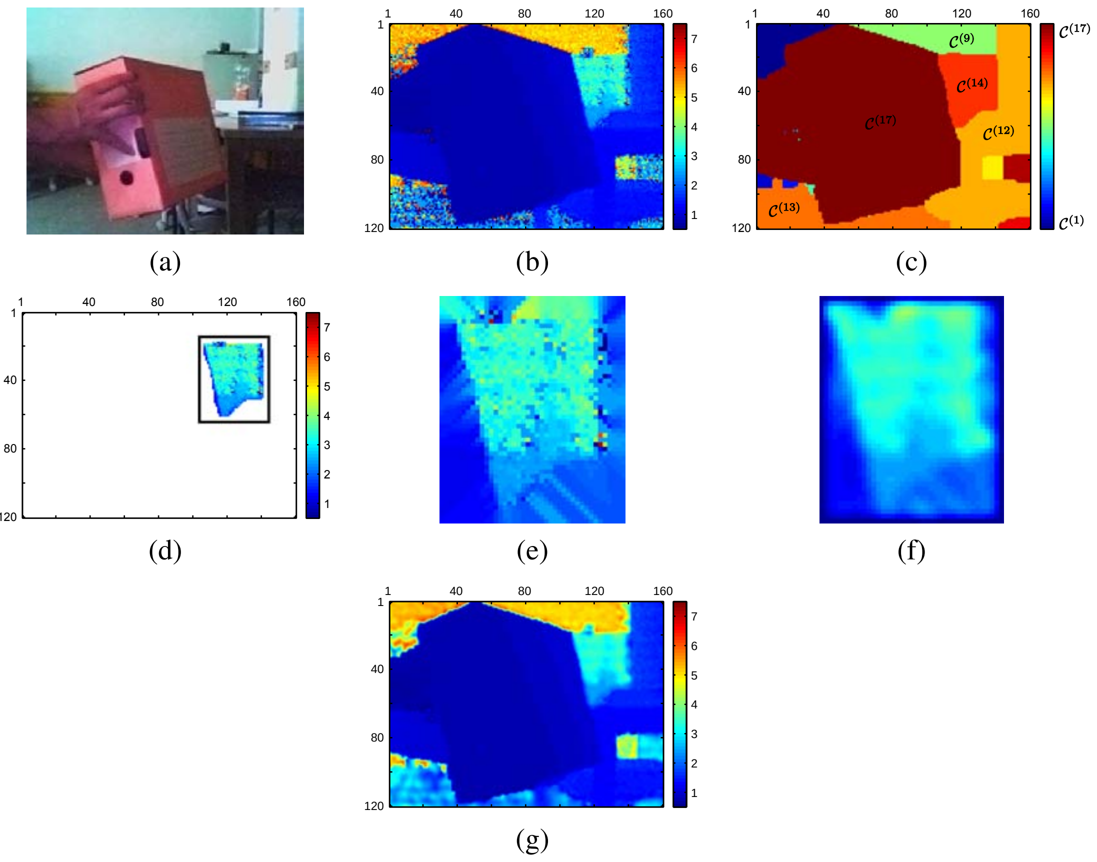


Figure 4.25: Illustration of our multistep algorithm to fit a B-spline to range data with discontinuities and heteroskedastic noise. (a) The scene viewed with an usual sensor (CMOS sensor). (b) Depth map acquired with a ToF camera (PMDTec 19k). (c) Depth map segmentation. (d) Bounding box for the segment $\mathcal{C}^{(14)}$. (e) Local depth map obtained by extrapolating the data from $\mathcal{C}^{(14)}$. (f) Local fitting. (g) Depth map sampled from the global surface obtained by merging all the local fittings. Compared to the initial depth map (b), the amount of noise has been reduced while preserving the edges of the objects.

4.3.2.1 Segmentation

The first step of our approach consists in segmenting the initial dataset according to the depth of the points. Using this criterion is motivated by the fact that the amount of noise depends mainly on the distance between the sensor and the scene. Besides, this approach makes the discontinuities coincident with the borders of the segments which is a desirable behaviour in the sense that it is easier to fit a surface on a dataset without discontinuities.

The segmentation is achieved with the *α-expansion* algorithm of (Boykov et al., 2001; Zabih and Kolmogorov, 2004). The implementation details are given below. The result of the segmentation step is a set of r labels $E = [\![1, r]\!]$ and a collection of r segments $\mathcal{C}^{(t)}$ for $t \in [\![1, r]\!]$. The segment $\mathcal{C}^{(t)}$ is the set of pixels $(i, j) \in [\![1, m]\!] \times [\![1, n]\!]$ labelled with the value t. Note that each segment is guaranteed to be 8-connected. We denote $\mathsf{C}^{(t)}$ the matrix such that $c_{i,j}^{(t)} = 1$ if $(i, j) \in \mathcal{C}^{(t)}$ and 0 otherwise.

Segmentation details. Let $\mathcal{J} = [\![1, m]\!] \times [\![1, n]\!]$ be the set of pixels of the depth map Z. The segmentation consists in classifying each pixel of $\mathcal{J}$ in one of the r segments $\{\mathcal{C}^{(t)}\}_{t\in[\![1,r]\!]}$. Besides, we want the segmentation to be spatially consistent and the segments to be sets of 8-connected pixels. To do so, we use the α-expansion algorithm proposed in Boykov et al. (2001); Zabih and Kolmogorov (2004). With this approach, the segmentation is the solution of an optimization problem:

$$\min_{\{\mathcal{C}^{(1)},\dots,\mathcal{C}^{(s)}\}} E_{\text{data}} + E_{\text{reg}}, \tag{4.104}$$

where s is an integer such that $s \leq r$. E_{data} is the data term, *i.e.* a term that measures the quality of allocating a certain label to certain pixels. E_{reg} is a term promoting the spatial consistency of the segmentation.

We chose as data term the likelihood that a depth $z_{i,j}$ belongs to the segment $\mathcal{C}^{(t)}$. If we consider a normal distribution of parameters $\boldsymbol{\theta}^{(t)} = (\mu^{(t)}, \sigma^{(t)})$ for the noise in the tth segment, E_{data} is given by:

$$E_{\text{data}} = \sum_{t\in[\![1,s]\!]} \sum_{(i,j)\in\mathcal{J}} D^{(t)}(i,j), \tag{4.105}$$

$$\text{with} \quad D^{(t)}(i,j) = -\log\left(L((i,j) \in \mathcal{C}^{(t)} | \boldsymbol{\theta}^{(t)}))\right), \tag{4.106}$$

$$\text{and} \quad L((i,j) \in \mathcal{C}^{(t)} | \boldsymbol{\theta}^{(t)}) \propto \exp\left(-\frac{(z_{i,j} - \mu^{(t)})^2}{2(\sigma^{(t)})^2}\right). \tag{4.107}$$

The parameters $\{\boldsymbol{\theta}^{(t)}\}_{t\in[\![1,s]\!]}$ are computed using the *k-means* algorithm on the depth map Z. The number s is determined manually. The regularization term E_{reg} is defined using the Potts model ($\mathcal{N}(\mathbf{p})$ denotes the 8-neighbourhood of the pixel $\mathbf{p}$):

$$E_{\text{reg}} = \sum_{\mathbf{p}\in\mathcal{J}} \sum_{\mathbf{q}\in\mathcal{N}(\mathbf{p})} V(\mathbf{p}, \mathbf{q}), \tag{4.108}$$

$$\text{with } V(\mathbf{p}, \mathbf{q}) = \begin{cases} 0 & \text{if } \mathbf{p} \in \mathcal{C}^{(t)} \text{ and } \mathbf{q} \in \mathcal{C}^{(t)} \\ c & \text{if } \mathbf{p} \in \mathcal{C}^{(t)} \text{ and } \mathbf{q} \in \mathcal{C}^{(t')}, t \neq t'. \end{cases}$$

The scalar $c \in \mathbb{R}_+$ determines the weight given to the spatial consistency: we use $c = 25$. The final touch of the segmentation consists in taking the r connected components of the s segments $\{\mathcal{C}^{(t)}\}_{t\in[\![1,s]\!]}$.

4.3.2.2 Bounding Boxes

Usually, the data contained in the segments $\left\{\mathcal{C}^{(t)}\right\}_{t=1}^{r}$ does not form a complete rectangle. However, the data must be arranged in a full grid so that the FGA algorithm can be used. To do so, we define for each segment $\mathcal{C}^{(t)}$ a bounding box $\mathcal{B}^{(t)}$ which is, intuitively, a rectangular subdomain of $[k^x_{-k}, k^x_{g+k+1}] \times [k^y_{-l}, k^y_{h+l+1}]$ that includes the data contained in $\mathcal{C}^{(t)}$ and of minimal size. In other words, $\mathcal{B}^{(t)}$ is the smallest rectangular domain $[\![\beta^{(t)}, \delta^{(t)}]\!] \times [\![\gamma^{(t)}, \eta^{(t)}]\!]$ (with $\beta^{(t)}, \delta^{(t)} \in M$ and $\gamma^{(t)}, \eta^{(t)} \in N$) that contains the non-rectangular domain $\mathcal{I}^{(t)}$ defined as follow:

$$\mathcal{I}^{(t)} = \bigcup_{\substack{(i,j) \text{ s.t.} \\ (u,v)\in\text{infl}(p_{i,j}) \\ \text{for } (u,v)\in\mathcal{C}^{(t)}}} \text{infl}(p_{i,j}) \qquad \text{for } t \in [\![1, r]\!], \tag{4.109}$$

where $\text{infl}(p_{i,j})$ is the domain of influence of the weight $p_{i,j}$, *i.e.* :

$$\text{infl}(p_{i,j}) = [k^x_j, k^x_{j+k+1}] \times [k^y_i, k^y_{i+l+1}]. \tag{4.110}$$

4.3.2.3 Local Depth Maps

Except for some exceptional cases, the data contained in the segment $\mathcal{C}^{(t)}$ does not fill completely the bounding box $\mathcal{B}^{(t)}$. Having a dataset organized as a complete grid is however a requirement to use the FGA algorithm. We thus define for each segment $\mathcal{C}^{(t)}$ a *local depth map*, denoted $\mathsf{Z}^{(t)}$, which is a rectangular completion of the data contained in $\mathcal{C}^{(t)}$. For all $t \in [\![1, r]\!]$, the local depth map $\mathsf{Z}^{(t)} \subset \mathbb{R}^{m \times n}$ is defined as:

$$z_{i,j}^{(t)} = \begin{cases} z_{i,j} & \text{if } (i,j) \in \mathcal{C}^{(t)} \\ 0 & \text{if } (i,j) \notin \mathcal{B}^{(t)} \\ \hat{z}_{i,j}^{(t)} & \text{otherwise.} \end{cases} \tag{4.111}$$

The values $\hat{z}_{i,j}^{(t)}$ are computed by recursively propagating the depths located on the borders of the class $\mathcal{C}^{(t)}$. The details of this operation are given by algorithm 6. Note that the local depth maps $\mathsf{Z}^{(t)}$ all have the same size than the initial depth map Z but they are non-zero only over $\mathcal{B}^{(t)}$. This makes the merging step easier.

Algorithm 6: Computation of the local depth map $\hat{\mathsf{Z}}^{(t)}$

```
input  : $\mathcal{B}^{(t)}$, $\mathsf{Z}$, $\mathsf{C}^{(t)}$
output : $\hat{\mathsf{Z}}^{(t)}$
data   : $\mathsf{S} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
1  begin
2      $\mathsf{A} \leftarrow \mathsf{C}^{(t)}$
3      $\mathsf{T} \leftarrow (\mathsf{A} \oplus \mathsf{S}) \setminus \mathsf{A}$
4      $\mathcal{T} \leftarrow \{(i,j) \in \mathcal{B}^{(t)} \mid \mathsf{T}_{i,j} = 1\}$
5      while $\mathcal{T} \neq \varnothing$ do
6          for $(i,j) \in \mathcal{T}$ do
7              $\mathcal{V} \leftarrow \{(u,v) \mid |u-i| \leq 1$ et $|v-j| \leq 1\}$
8              $z \leftarrow 0$
9              $n \leftarrow 0$
10             for $(u,v) \in \mathcal{V} \cap \mathcal{C}^{(t)}$ do
11                 $z \leftarrow z + z_{u,v}$
12                 $n \leftarrow n + 1$
13             $\hat{z}_{i,j}^{(t)} \leftarrow z/n$
14         $\mathsf{A} \leftarrow \mathsf{A} + \mathsf{T}$
15         $\mathsf{T} \leftarrow (\mathsf{A} \oplus \mathsf{S}) \setminus \mathsf{A}$
16         $\mathcal{T} \leftarrow \{(i,j) \in \mathcal{B}^{(t)} | t_{i,j} = 1\}$
```

4.3.2.4 Local Fittings

The fourth step of our approach consists in fitting a surface to each one of the r depth maps $\mathsf{Z}^{(t)}$. This is achieved using the FGA algorithm of section 4.3.1. The use of the FGA algorithm is possible because all the requirements are fulfilled: the local depth maps are complete rectangular grids and, moreover, the noise is homogeneous inside each one of the segments. We use the following parameters for the FGA algorithm:

$$\bar{M}^{(t)} = M \cap [\beta^{(t)}, \delta^{(t)}], \qquad \bar{A}^{(t)} = A \cap [\beta^{(t)}, \delta^{(t)}],$$
$$\bar{L}^{(t)} = L \cap [\gamma^{(t)}, \eta^{(t)}], \qquad \bar{B}^{(t)} = B \cap [\gamma^{(t)}, \eta^{(t)}].$$

The result of this step is a collection of r matrices of control points $\mathsf{P}^{(t)} \in \mathbb{R}^{(g+k+1)\times(h+l+1)}$. Note that $p_{i,j}^{(t)} = 0$ if there is no pixel $(u, v) \in \mathcal{B}^{(t)}$ such that $(u, v) \in \text{infl}(p_{i,j})$. The regularization parameter for the tth depth map, denoted $\alpha^{(t)}$, is computed using the data contained in $\mathcal{C}^{(t)}$ only, not all the data from $\mathsf{Z}^{(t)}$.

4.3.2.5 Merging

The last step of our approach consists in merging all the local fittings in order to get a surface that fits the whole depth map Z. To do so, we build a B-spline by taking the control points computed during the previous step. For $t \in [\![1, r]\!]$, let $\mathsf{Q}^{(t)} \in \mathbb{R}^{(g+k+1)\times(h+l+1)}$ be the matrix that indicates the control points of 'interest' for the segment $\mathcal{C}^{(t)}$, *i.e.* the ones that have an influence region intersecting with the data contained in $\mathcal{C}^{(t)}$:

$$q_{i,j}^{(t)} = \begin{cases} 1 & \text{if } \exists(u,v) \in \mathcal{C}^{(t)} \text{ s.t. } (u,v) \in \text{infl}(p_{i,j}) \\ 0 & \text{otherwise.} \end{cases}$$

The set of interesting control points is then reduced so that there are no conflicts between the control points coming from different local fittings. This is achieved by eroding the binary matrix $\mathsf{Q}^{(t)}$:

$$\hat{\mathsf{Q}}^{(t)} = \mathsf{Q}^{(t)} \ominus \mathsf{S}_2, \tag{4.112}$$

where S_2 is the following structuring element:

$$\mathsf{S}_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.113}$$

Finally, the surface that fits the whole depth map Z is the B-spline having for control points the matrix $\mathsf{P}^\star$:

$$\mathsf{P}^\star = \sum_{t=1}^{r} \bar{\mathsf{P}}^{(t)}, \tag{4.114}$$

$$\text{with } \bar{p}_{i,j}^{(t)} = \begin{cases} p_{i,j}^{(t)} & \text{if } \hat{q}_{i,j}^{(t)} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.115}$$

4.3.3 Experiments

4.3.3.1 Standard Deviation Estimator

In this experiment, we assess the precision of the standard deviation estimator described in section 4.3.1.4. To do so, we randomly generate some surfaces as linear combination of elementary functions (sine waves, exponential, polynomials). Depth maps are then obtained by sampling these surfaces on a grid of size 120×160. An additive normally-distributed noise with standard deviation σ is added to the depth maps. The estimated standard deviation σ_e is finally compared to the expected value σ by computing the relative error between them. The estimator is tested for several levels of noise ranging from 1% to 21% of the maximal data amplitude. The estimation is repeated for 100 surfaces for each level of noise. We obtain a relative error that is **consistently smaller than 6%**.

4.3.3.2 Grid Approach

This experiment shows the interest of exploiting the tensorial properties of the B-spline model instead of the direct approach in order to fit a surface on data arranged as an orthogonal grid. Figure 4.26 shows the computation times required by the two approaches in function of the number of data points (with slightly less data points than control points). It is obvious from this illustration that the grid approach is much faster than the direct approach. We can even say that the grid approach is compulsory if we want to fit surfaces in a reasonable amount of time for large datasets. Moreover, the memory space required by the direct approach is much more important than for the grid approach. On our test machine, the computations were intractable for the standard approach with more than 1500 data points.

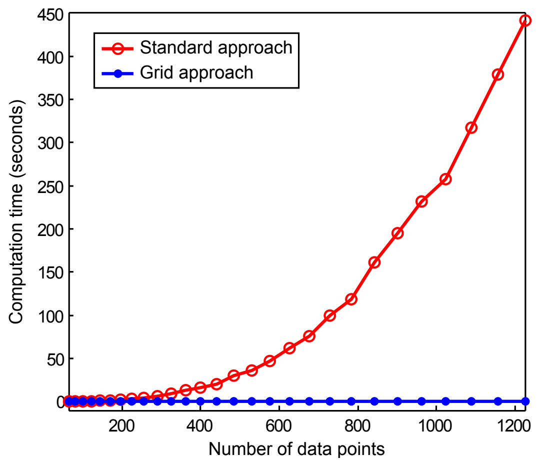


Figure 4.26: Illustration of the performance gain obtained when using a grid approach instead of the standard one to fit surfaces on range data.

4.3.3.3 Qualitative Results

Figure 4.27 shows some representative results of our surface fitting algorithm. These results are compared to the ones obtained with a global regularization scheme. Since the discrepancy principle is only usable on data with homogeneous noise, we used the generalized cross validation principle (Wahba, 1990) to compute the regularization parameter with the global regularization scheme. It is clear from figure 4.27 that our approach is able to simultaneously use the proper amount of regularization while preserving the sharp edges present in the observed scene. On the contrary, a global regularization scheme leads to an overly smoothed surface that does not reflect well the scene geometry.

4.3.3.4 Quantitative Results

In this experiment, we compare the quality of the surfaces fitted using our algorithm described in section 4.3.2 with the ones obtained using a global regularization scheme. This is achieved by comparing the fitted surface with a ground truth surface. To do so, we used a set of 214 range images publicly available at (CESAR laboratory (Oak Ridge National Laboratory)). These images were acquired using an *Odetics LADAR camera* and, as a consequence, the amount of noise is quite limited. These datasets can thus be used as ground truth surfaces. Let $\bar{Z}$ be a range image acquired with the LADAR camera. The different algorithms are tested using datasets obtained by adding a normally-distributed noise to the LADAR range images with a standard deviation proportional to the depth. Let Z be the noisy dataset:

$$z_{i,j} = \bar{z}_{i,j} + e_{i,j}, \tag{4.116}$$

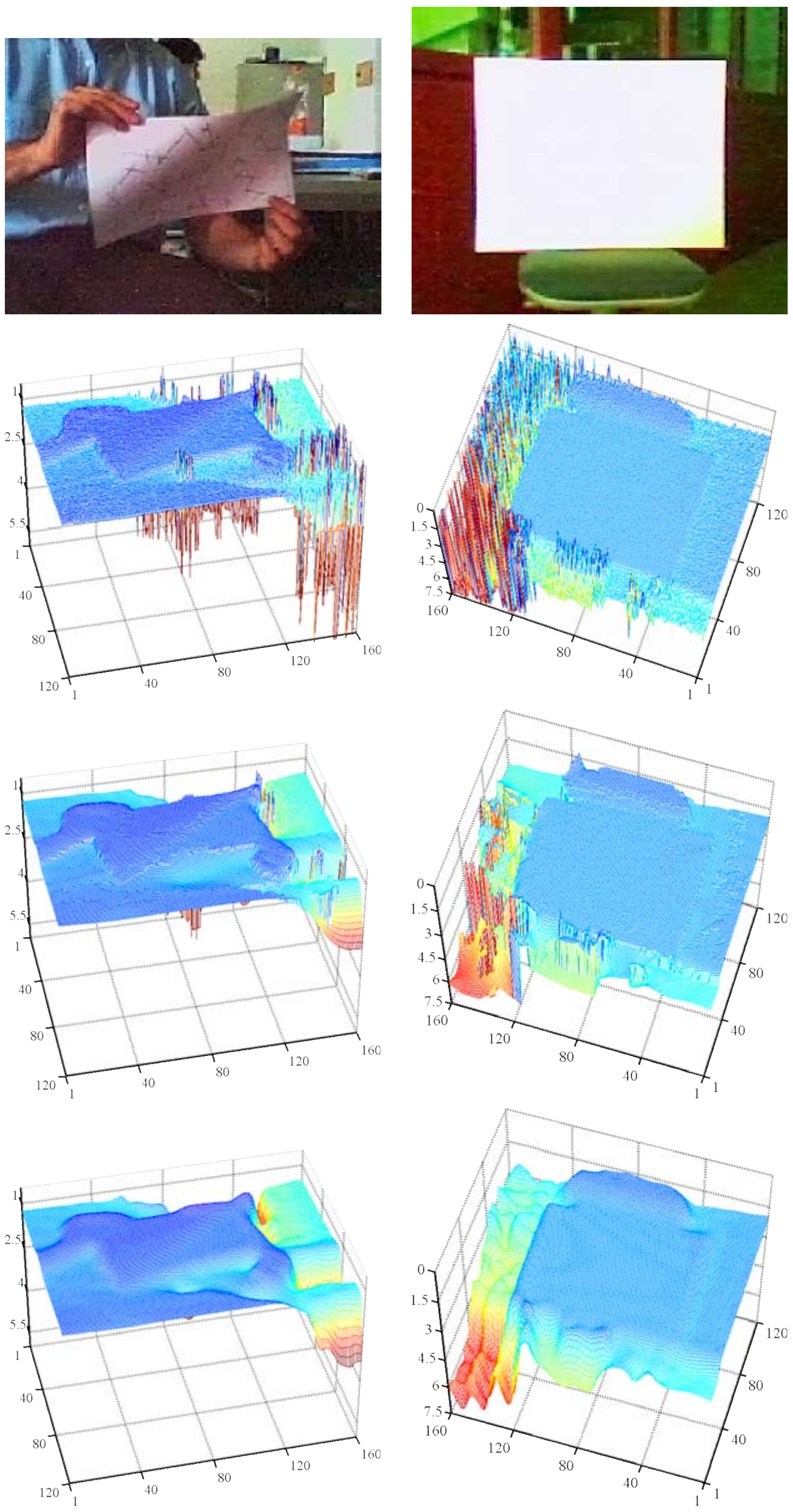


Figure 4.27: First row: a view of the scene taken with a CMOS sensor. Second row: raw depth maps. Notice the high level of noise present in the background. Third row: surface fitted using our algorithm. Last row: surface fitted using a global approach. With this approach, we can see that the object boundaries are over-smoothed compared to the results obtained with our algorithm.

where $e_{i,j}$ is a random variable drawn from a normal distribution $\mathcal{N}(0, \sigma_{i,j})$ with $\sigma_{i,j} \propto z_{i,j}$. Let f_o and f_g be the surfaces fitted to the range image Z with our algorithm and a global regularization scheme respectively. Let $\hat{\mathsf{Z}}^o$ and $\hat{\mathsf{Z}}^g$ be the depth maps predicted with f_o and f_g respectively. The quality of the fitted surface is assessed by comparing the norm between the predicted depth maps $\hat{\mathsf{Z}}^o$ and $\hat{\mathsf{Z}}^g$ with the ground truth one Z. The results, averaged for the 214 range images of the dataset, are reported in figure 4.28 for different levels of noise. It shows that our approach is, in average, 20% better than the one using a global regularization scheme whatever the noise intensity is.

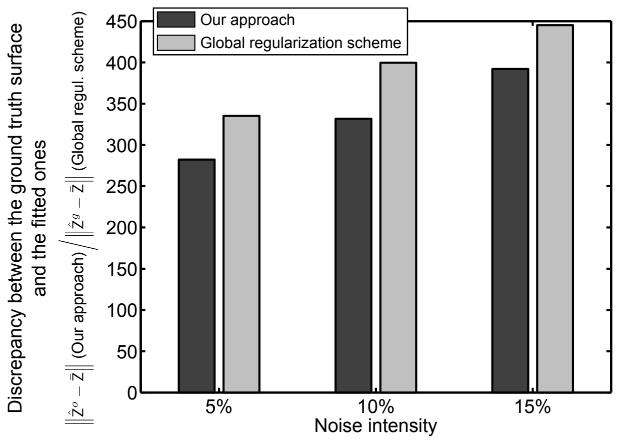


Figure 4.28: Discrepancy between ground truth range images and the ones predicted with the fitted surface using our algorithm (dark grey) and a global regularization scheme (light grey). The indicated level of noise corresponds to the standard deviation of a normally-distributed noise for the deepest point and relative to the total depth of the scene. The surfaces fitted with our approach are better than the ones fitted using a global regularization scheme.

Chapter 5

Image Registration

Image registration is one of the most fundamental problems in computer vision. It consists in finding the transformation that aligns two or more images. Many techniques have been proposed to solve this problem. We start this chapter by reviewing the state-of-the-art approaches to image registration. In particular, the two fundamental approaches for parametric image registration, *i.e.* the feature-based and the direct approaches, will be introduced. Then, we will present two of our contributions related to image registration. The first one is a method that allows one to use a direct approach to image registration without needing a region of interest. This first contribution uses a robust framework for direct image registration based on M-estimators. It has been published in (Brunet et al., 2009c, 2010c,d). The second one is a new method to automatically tune the hyperparameters that naturally arise when using a feature-based approach to register images. The novelty of this approach lies in the fact that it uses the pixel information in addition to the features. It has been published in (Brunet et al., 2010a,b).

Note chapter 6 will also deal with image registration. The main difference of this next chapter is that it will be about defining a parametric image deformation model instead of the parameter and hyperparameter estimation as the current chapter.

5.1 General Points on Image Registration

5.1.1 Background

The problem of registering two or more images is to determined a transformation that aligns the input images (Irani and Anandan, 1999; Torr and Zisserman, 1999). Note that in thesis, we will mainly consider the case with only two images. Image registration is a problem that impacts numerous fields such as, to cite a few, computer vision (Bartoli, 2008c,c; Bartoli and Zisserman, 2004; Szeliski, 2006; Zitová and Flusser, 2003), medical imaging (Aouadi and Sarry, 2008; Groher et al., 2010; Hahn et al., 2010, 2006; Modersitzki, 2004; Zikic et al., 2010a,b), or metrology (Corpetti et al., 2002; Gendrich and Koochesfahani, 1996; Heitz et al., 2008; Papadakis and Mémin, 2008).

Broadly speaking, two types of transformation may be estimated (Gay-Bellile, 2008):

A geometric transformation that modifies the position of the pixels in the pictures. This transformation results in a dense deformation field between a reference image and a target image. Our contributions related to image registration mainly deal with this kind of transformations, especially with non-rigid ones.

A photometric transformation that modifies the colour of the pixels. This transformation models the illumination changes. This type of transformation is not always used when registering two images. This is especially the case when the images are registered using features which are invariant to illumination changes.

Parametric image registration is a classical parameter estimation problem in the sense that, as always, it consists in fixing some parametric model and then determining its parameters from the data by minimizing some criterion. The parametric model that explains the deformation between the images to register is typically called a *warp*. The general principle of image registration is illustrated in figure 5.1.

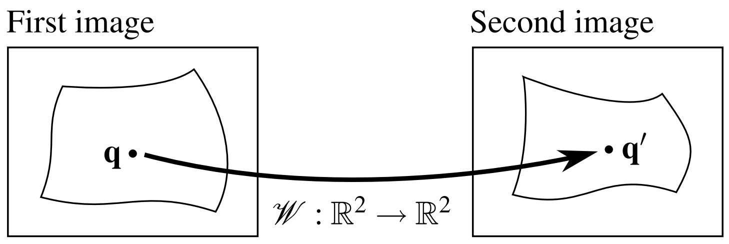


Figure 5.1: General principle of image registration. Image registration consists in estimating the transformations that modify an image so that it matches another one. Several type of transformations may be considered. In this thesis, we are mainly interested in geometric transformations.

There are two main approaches for parametric image registration: the *direct approach* and the *feature-based approach*. As its name may indicate, the direct approach relies on the information which is directly available such as the colour of the pixels. With this approach, the criterion to be minimized measures the colour discrepancy between the warped source image and the target image. In the feature-based approach, the warp parameters are estimated from a finite set of features. The features are distinctive elements of the images which are first extracted independently in each image to register. Then, they are matched across the images. Both approaches have their own advantages and disadvantages. Other approaches such as cross-validation and frequential approaches may be used to register images. However, we are not interested in these approaches in this document.

Properties of the direct approach. The main advantage of this approach is the high quantity of data used to estimate the warp parameters. Another good point is that it does not require one to extract and match features. On the other side, the direct approach is sensitive to changes of appearance. Besides, it is not suited for registering images with important displacements since the colour information is correlated to the displacement only locally.

Properties of the feature-based approach. Contrarily to the direct approach, the feature-based approach is better suited for large displacements (wide-baseline registration). Another advantage of the feature-based approach is that they may be robust to illumination changes. The main drawback of the feature-based approach is that it requires the extraction and the matching of features. Depending on the nature of the images, there may be only a few feature correspondences which make difficult the estimation of the warp parameters. This is especially true for badly textured images. The proportion of false point correspondences (outliers) may also grow quickly when, for instance, the images contains repetitive and undistinguishable patterns.

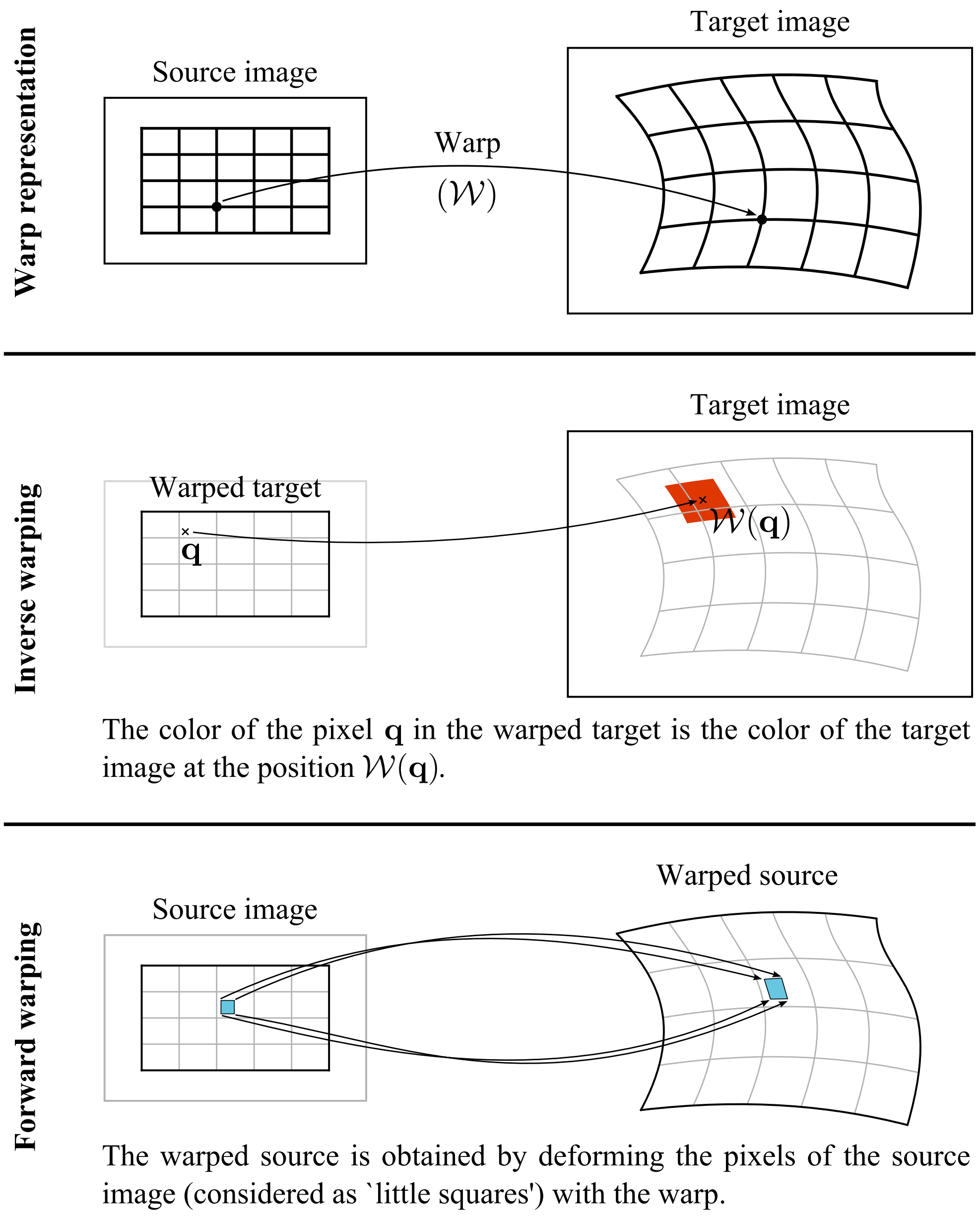
What is deformed? Once a warp has been estimated between two images, it is natural to deform one of the images in order to align them. This is something which may be a bit confusing. Indeed, although the warp is a function from the source image to the target image[1], it is generally easier to deform the target image. We call this technique the *inverse warping*. It can be achieved without having to compute the inverse warp. This is realized as follows. We take an empty picture of the same size than the source image. The colour of each pixel of this picture (the warped target) is the colour of the corresponding pixel in the target image determined by applying the estimated warp. With this approach, the target image is not completely warped. Indeed, the pixels of the target image which do not have an antecedent in the domain of the source image will not be warped. This can be obviated using the same principle except that we consider an initial domain for the warped target image larger than the one of the source image. The principle used to warp the target image is illustrated in figure 5.2.

If one is ready to make some approximations, it is also possible to make a *forward* warping of the source image. Therefore, the source image is aligned to the target image. This can be achieved by considering the pixels as 'little squared area'. The corners of these squares can then be transformed using the estimated warp. Even if we use deformable warps, the pixels are transformed in a rigid way. Practically, the warped source image can be represented as a vectorial image. It can then be converted to a bitmap using a standard graphical library such as Cairo[2]. Although quite natural, this approach has the major drawback of being much heavier in terms of computational time than the inverse warping. The forward warping is illustrated in figure 5.2.

Derivatives of an image. As it will become clearer later in this chapter, most of the approaches to image registrations requires one to compute the derivatives of images. Indeed, since image registration is formulated as an optimization problem, the optimization algorithms often need derivative information such as the gradient or the Hessian matrix. Let $\mathcal{I}$ be an image. It is a common assumption to consider that $\mathcal{I}$ is defined over a rectangular bounded subdomain of $\mathbb{R}^2$. However, in practice, this is not quite true. Indeed, a digital image is defined over a rectangular bounded subdomain of $\mathbb{Z}^2$. Therefore, one of the most simple technique to compute the derivatives of an image consists in using *finite differences*. Finite differences come in different flavours. For instance, if we consider the derivative of the image $\mathcal{I}$ with respect to x, *i.e.* the first free variable of the function, then we have:

[1]The fact that a warp is naturally a function from the source image to the target image is revealed when we use a visualisation grid to display the warp.

[2]`http://cairographics.org/`



The color of the pixel $\mathbf{q}$ in the warped target is the color of the target image at the position $\mathcal{W}(\mathbf{q})$.

The warped source is obtained by deforming the pixels of the source image (considered as `little squares') with the warp.

Figure 5.2: Illustration of the inverse and forward warping.

- Forward finite differences:
$$\frac{\partial \mathcal{I}}{\partial x}(\mathbf{q}) \approx \mathcal{I}\left(\mathbf{q} + \begin{pmatrix}1\\0\end{pmatrix}\right) - \mathcal{I}(\mathbf{q}) \tag{5.1}$$
- Backward finite differences:
$$\frac{\partial \mathcal{I}}{\partial x}(\mathbf{q}) \approx \mathcal{I}(\mathbf{q}) - \mathcal{I}\left(\mathbf{q} - \begin{pmatrix}1\\0\end{pmatrix}\right) \tag{5.2}$$
- Central finite differences:
$$\frac{\partial \mathcal{I}}{\partial x}(\mathbf{q}) \approx \frac{\mathcal{I}\left(\mathbf{q} + \binom{1}{0}\right) - \mathcal{I}\left(\mathbf{q} - \binom{1}{0}\right)}{2} \tag{5.3}$$

In this document, the default choice is the forward finite differences. Note that there exists other more complex techniques to evaluate the derivatives of a discrete function (see, for instance, (Brunet, 2007; Malgouyres et al., 2008).

5.1.2 Problem Statement

5.1.2.1 Image Registration as an Optimization Problem

Typical cost function. Image registration may be cast into a parameter estimation problem. In its most generic form, the non-rigid image registration problem is written as:

$$\min_{\mathbf{p}} \mathcal{E}_d(\mathbf{p}) + \lambda \mathcal{E}_s(\mathbf{p}), \tag{5.4}$$

where $\mathbf{p}$ is the vector containing the parameters of the warp (such as the control points of a B-spline). $\mathcal{E}_d$ is the *data term* which compares the information between the source and the target images. The term $\mathcal{E}_s$ is the smoothing term that regularizes the warp. The hyperparameter λ controls the relative importance of the data and smoothing terms. We now give some details of the different terms presented in equation (5.4).

Direct approach and least-squares. The most common data term for the direct approach to image registration is the *sum of squared difference* (SSD). It is computed between the source image and the target image deformed with the estimated warp:

$$\mathcal{E}_d(\mathbf{p}) = \sum_{\mathbf{q} \in \mathfrak{R}} \|\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathcal{W}(\mathbf{q}; \mathbf{p}))\|^2, \tag{5.5}$$

where $\mathfrak{R}$ is the region of interest (ROI). It is a set of pixels of the source image that should be carefully chosen. More details on this region of interest along with some of our contributions related to this matter will be given in section 5.2. The principle of the SSD term for direct image registration is illustrated in figure 5.3.

Mutual Information. The *Mutual Information* (MI) is another common principle to image registration that raises a different data term. It has been independently introduced in (Collignon et al., 1995; Viola, 1995). Note that we do not use this kind of criterion in the rest of this document. The MI is the quantity of information of the first image which is contained in the second image. The MI is maximal when the two images are identical. There exists several definitions. We report here the resulting data term presented in (Gay-Bellile, 2008):

$$\mathcal{E}_d(\mathbf{p}) = \zeta(\mathcal{S}) + \zeta(\mathcal{T}(\mathcal{W}(\bullet; \mathbf{p}))) - \zeta(\mathcal{S}, \mathcal{T}(\mathcal{W}(\bullet; \mathbf{p}))), \tag{5.6}$$

where ζ is the Shannon entropy (Shannon, 1948). It measures the quantity of information contained in a sequence of events. The entropy of an image is related to its complexity. A completely untextured image has

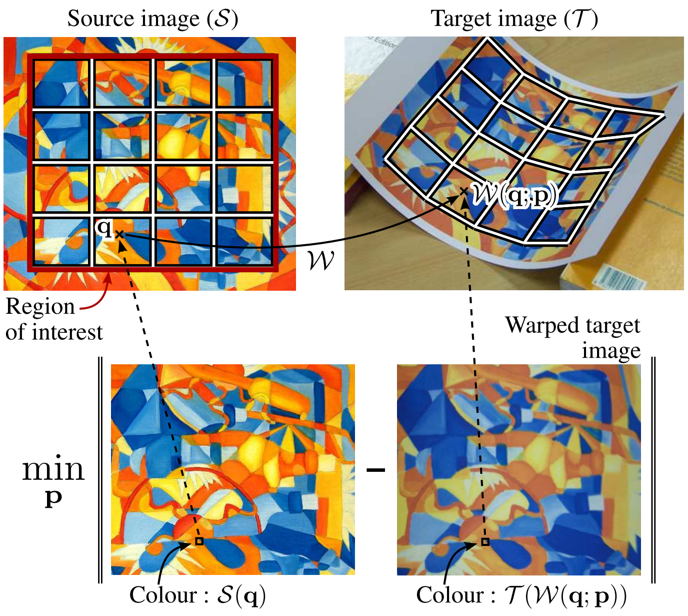

Source image ($\mathcal{S}$)
Target image ($\mathcal{T}$)
$\mathbf{q}$
$\mathcal{W}(\mathbf{q};\mathbf{p})$
$\mathcal{W}$
Region of interest
Warped target image
$\min_{\mathbf{p}}$
Colour : $\mathcal{S}(\mathbf{q})$
Colour : $\mathcal{T}(\mathcal{W}(\mathbf{q};\mathbf{p}))$

Figure 5.3: Illustration of the principle of the SSD term for direct image registration. The SSD term is the sum of the difference of colours of the pixels in the ROI (in the source image) and their correspondent in the target image (computed using the warp induced by the parameters $\mathbf{p}$).

a low entropy while a highly textured image has a higher value. The joint entropy $\zeta(\mathcal{S}, \mathcal{T}(\mathcal{W}(\bullet;\mathbf{p})))$ measures the quantity of information shared by the images $\mathcal{S}$ and $\mathcal{T}(\mathcal{W}(\bullet;\mathbf{p}))$. If these images are identical then their joint entropy is minimal. It may be computed from the joint histogram of the two images. Maximizing the MI is thus equivalent to searching for the transformation that results in a maximum of information in the images while having the images as similar as possible. Contrarily to the other types of approaches, the MI criterion does not rely directly on the colour information. It makes the MI criterion particularly well-suited for multi-modal image registration. This is the reason why the MI criterion is very popular in medical image registration (Modersitzki, 2004; Pluim et al., 2003; Rueckert et al., 1999).

Feature-based criterion. Typically, the data term used in feature-based image registration measures the (squared) Euclidean distance between the features of the source image transformed with the warp and the features of the target image. This may be written as:

$$\mathcal{E}_d(\mathbf{p}) = \sum_{\{\mathbf{f}\leftrightarrow\mathbf{f}'\}\in\Theta} \left\|\mathcal{W}(\mathbf{f};\mathbf{p}) - \mathbf{f}'\right\|^2, \tag{5.7}$$

where Θ is a set of feature correspondences that was previously detected and matched in the images.

M-estimators and robustness. A simple yet powerful extension of the above-mentioned data term consists in using an M-estimator instead of a squared Euclidean distance. This allows one to get estimation procedures more robust to erroneous data. Erroneous data are very common and almost unavoidable with the two most common approaches to image registration. For instance, in the feature-base approach, the are often false feature correspondences. This generally comes from a poor extraction of the features or from a bad matching of the features. The standard non-robust data term for feature-based image registration is modified with an M-estimator in the following way:

$$\mathcal{E}_d(\mathbf{p}) = \sum_{\{\mathbf{f}\leftrightarrow\mathbf{f}'\}\in\Theta} \rho\left(\mathcal{W}(\mathbf{f};\mathbf{p}) - \mathbf{f}'\right), \tag{5.8}$$

In the case of the direct approach, outliers are due to phenomena such as occlusions and specularities.

Indeed, if such phenomena arise then the colour of two corresponding pixels may be quite different. The use of an M-estimator in the direct approach has been utilized in, for instance, (Arya et al., 2007; Odobez and Bouthemy, 1995). With an M-estimator, the initial data term of the direct approach is transformed into the following one:

$$\mathcal{E}_d(\mathbf{p}) = \sum_{\mathbf{q}\in\mathfrak{R}} \rho\left(\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathcal{W}(\mathbf{q};\mathbf{p}))\right). \tag{5.9}$$

Regularization term. As we said in equation (5.4), the cost function for non-rigid image registration generally includes a smoothing term (also known as regularization term). Many such terms have been proposed in the literature. For instance, one may use the smoothing term of order 1 proposed in (Horn and Schunck, 1981):

$$\mathcal{E}_s(\mathbf{p}) = \iint_{\Omega} \left\| \frac{\partial \mathcal{W}}{\partial x}(\mathbf{q};\mathbf{p}) \right\|^2 + \left\| \frac{\partial \mathcal{W}}{\partial y}(\mathbf{q};\mathbf{p}) \right\|^2 \mathrm{d}\mathbf{q}, \tag{5.10}$$

where Ω is the definition domain of the warp $\mathcal{W}$. Alternatively, one may use a smoothing term of second order : the bending energy. It is similar to the bending energy used in the previous chapter for range data fitting. With a warp, which is a function from $\mathbb{R}^2$ to $\mathbb{R}^2$, the bending energy is defined by:

$$\mathcal{E}_s(\mathbf{p}) = \sum_{k=1}^{2} \iint_{\Omega} \left\| \frac{\partial^2 \mathcal{W}^k}{\partial x^2}(\mathbf{q};\mathbf{p}) \right\|^2 + 2\left\| \frac{\partial^2 \mathcal{W}^k}{\partial x \partial y}(\mathbf{q};\mathbf{p}) \right\|^2 + \left\| \frac{\partial^2 \mathcal{W}^k}{\partial y^2}(\mathbf{q};\mathbf{p}) \right\|^2 \mathrm{d}\mathbf{q}, \tag{5.11}$$

where $\mathcal{W}^1$ and $\mathcal{W}^2$ are respectively the first and the second coordinates of $\mathcal{W}$.

5.1.2.2 Direct Methods

Here, we present some more advanced topics on the direct methods for image registration. We keep talking about general elements.

Handling Illumination Changes The basic criterion of equation (5.5) for the direct approach to image registration is extremely sensitive to illumination changes. Indeed, there may be some important discrepancies in the colour of the corresponding pixels between images taken under different circumstances (lighting conditions, cameras, *etc*.) There exists several approaches to obviate this problem. For instance, the illumination changes can be explicitly modelled by adding a photometric transformation in the criterion. The criterion thus becomes:

$$\mathcal{E}_d(\mathbf{p},\mathbf{u}) = \sum_{\mathbf{q}\in\mathfrak{R}} \left\| \mathcal{S}(\mathbf{q}) - \mathcal{P}(\mathcal{T}(\mathcal{W}(\mathbf{q};\mathbf{p}));\mathbf{u}) \right\|^2. \tag{5.12}$$

The transformation $\mathcal{P}$ parametrized by the vector $\mathbf{u}$ represents the illumination changes. Therefore, the optimization takes place simultaneously over $\mathbf{p}$ and $\mathbf{u}$. This type of approach has been used in, for instance, (Bartoli, 2008a; Silveira and Malis, 2007).

Another solution to overcome the problems related to illumination changes is to estimate the geometric transformation on data invariant to such changes. For instance, a normalization of the images may introduce some kind of robustness to global illumination changes (such global transformation might appear when the images to register have been taken with different cameras). In (Pizarro and Bartoli, 2007), the images are converted into a different colour space which is invariant to shadows.

Multi-scale approaches One major drawback of the direct approaches to image registration is the fact that they are usually able to estimate the deformation only if it is not too wide. A partial remedy to this problem is the use of a *multi-scale approach* (Lindeberg and ter Haar Romeny, 1994). The underlying idea of such approaches is to first estimate the geometric deformation on a low resolution version of the images and then propagating the estimated deformation to higher resolutions. A pyramid of images is thus constructed along with the registration process. Several techniques may be used to construct a low resolution version of an image. For instance, one may compute the value of a pixel at the level k as a weighted mean of the pixels at the level $k-1$. The weights are typically determined with a bell-shaped Gaussian curve. Note that in image registration, the multi-scale approach works quite well for large translations. For other types of geometric transformation, it is generally less efficient.

5.1.2.3 Feature-Based Approach

We now give some supplementary details on the feature-based approach to image registration. In particular, we quickly review some very common types of features. We also say a word on the important step of matching the features. Detecting and matching feature is the first step in feature-based image registration, nonetheless essential to successfully register images.

Features. Features are distinctive elements of an image. They can be viewed as a high level representation of an image since they abstract the content of an image. There exists many types of features. In a nutshell, features can be classified into two main categories : the point-wise features (keypoints) and the area-based one. An algorithm in charge of extracting features is named a *detector*. In addition to a spatial information, a descriptor is often associated to a *descriptor*. The descriptor can be viewed as a characteristic signature of the feature. They are used to couple the features across several images relying on the principle that features with similar descriptors are likely to represent the same point.

A desirable property for any feature is to be *invariant*, *i.e.* to have a descriptor that does not depend too much on certain transformations. For instance, a feature can be invariant to a change of illumination or to a geometric transformation such as a rotation.

We now give some explanations on the most commonly used features in image registration. Supplementary general informations on features may be found in, for instance, (Szeliski, 2006; Tuytelaars and Mikolajczyk, 2007).

SIFT. SIFT (Scale Invariant Feature Transform) is one of the most popular keypoint detectors. It has been introduced in (Lowe, 1999). The SIFT feature descriptors are invariant to scale, orientation, and partially invariant to illumination changes (Lowe, 2004). It has also been shown to perform quite well under non-rigid deformations (Doshi et al., 2008).

We now give the general idea behind the SIFT detector. We do not give here the detailed algorithm since it is not the core of this work. In the rest of this document, features detectors are just tools we sometimes use in experiments. With SIFT, the locations of the keypoints are the minima and the maxima of the result of Difference Of Gaussians applied to a pyramid of images built from the initial image by smoothing and resampling. Using a pyramid of images makes SIFT invariant to the scale. The points obtained in this way are then filtered: points with low contrast and points along edges are discarded. The dominant orientations of the resulting keypoints are then computed. The description of the keypoints are computed relatively to the dominant orientations. This allows one to be invariant to the orientation. The SIFT descriptor (typically a sequence of 128 bytes) is finally computed by considering the pixels around a radius of the keypoint location. This descriptor encodes an orientation histogram computed for all the pixels in the neighbourhood of the keypoint.

SURF. SURF (Speeded Up Robust Features) is another keypoint detector which shares many similarities with SIFT. It was first introduced in (Bay et al., 2006). Supplementary details may be found in (Bay et al., 2008). The main advantage of SURF is that it is much faster than SIFT. SURF is built on the same concepts than SIFT but with some radical approximations to speed up the detection process (Schweiger et al., 2009). Instead of using a the difference of Gaussian operator to detect the maxima and the minima in the pyramid of images, SURF uses the determinant of the Hessian. The descriptor associated to the keypoints are also similar to the one produced by SIFT. It is computed from orientation histograms in a square neighbourhood centred at the keypoint location. The resulting descriptor is a vector of 64 elements.

MSER. MSER (Maximally Stable Extremal Region) is one of the most common method to detect area-based regions in images. It was developed by (Matas et al., 2002) to find correspondences between image areas from two images with different points of views. Additional information may be found in (Mikolajczyk and Schmid, 2005; Mikolajczyk et al., 2005; Tuytelaars and Mikolajczyk, 2007). It relies on the concept of *extremal region*. Let $\mathcal{I}$ be an image and let Q be a contiguous region of $\mathcal{I}$. Let us denote ∂Q the contour of the region Q. Q is an extremal region if either $\forall p \in Q, \forall q \in \partial Q : \mathcal{I}(p) > \mathcal{I}(q)$ (maximum intensity region) or $\forall p \in Q, \forall q \in \partial Q : \mathcal{I}(p) < \mathcal{I}(q)$ (minimum intensity region). A maximally stable extremal region is defined as follows. Let $Q_1 \subset \ldots \subset Q_i$ be a sequence of nested extremal regions. The region $Q_{i^\star}$ is maximally stable if and only if the quantity $f(i) = |Q_{i+\delta} \setminus Q_{i-\delta}|/|Q_i|$ has a local minimum at $i^\star$. δ is a parameter of the method.

MSER has many interesting properties. For instance, it is invariant to affine transformation of image intensities. It is also stable: only the regions whose support is nearly the same over a range of thresholds are selected. MSER automatically detects regions at different scales without having to explicitly build an image pyramid. Computing the MSER regions can be achieved quite rapidly (Nistér and Stewénius, 2008). Besides, MSER has been compared to other region detectors in (Mikolajczyk et al., 2005) and it has been shown to consistently give the best results for many tests. MSER is therefore a reliable region detector.

5.2 Direct Image Registration without Region of Interest

As a reminder of section 5.1.2.1, direct image registration in its most basic form amounts to solve the following minimization problem:

$$\min_{\mathbf{p}} \sum_{\mathbf{q} \in \mathfrak{R}} \|\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathcal{W}(\mathbf{q}; \mathbf{p}))\|^2 . \tag{5.13}$$

For the sake of simplicity, we forget about the regularization term in equation (5.13) since it is not of central importance in the contribution proposed in this section. We also remind here the classical variant of equation (5.13) which consists to use an M-estimator in order to get a robust registration algorithm:

$$\min_{\mathbf{p}} \sum_{\mathbf{q} \in \mathfrak{R}} \rho \left(\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathcal{W}(\mathbf{q}; \mathbf{p}))\right) . \tag{5.14}$$

This formulation is important in this section since the proposed contribution relies on it.

In this section, we present a contribution we made concerning direct image registration. More precisely, this contribution deals with the problems related to the region of interest (ROI). As it will be explained later in this section, the ROI may represent a major difficulty when doing direct image registration. Here, we propose a new solution that allows one to use the direct approach to register images without needing a ROI. This is made possible by a wise use of the robust framework to direct image registration that relies on saturated M-estimators.

5.2.1 Introduction

As said previously in this chapter, standard direct image registration consists in estimating the geometric warp between a source and a target images by maximizing the photometric similarity for the pixels of a Region of Interest (ROI). The ROI must be included in the real overlap between the images otherwise standard registration algorithms fail. As it will be shown later, determining a proper ROI is a hard 'chicken-and-egg' problem since the overlap is only known after a successful registration. Almost all algorithms in the literature consider that the ROI is given. This is generally either inconvenient or unreliable.

In this section we propose a new method that registers two images without using a ROI. The key idea of our method is to consider the off-target pixels as outliers. We define the off-target pixels as those pixels of the source image mapped outside the target image by the current warp. We use the classical robust M-estimation framework to handle both the off-target pixels and the usual outliers caused, for instance, by occlusions. With our formulation, the true image overlap is defined as the set of inliers.

Experiments on synthetic and real data with the homography and Free-Form Deformation based on B-splines show that our method outperforms standard approaches in terms of accuracy and robustness while precisely retrieving the overlap in the source and target images.

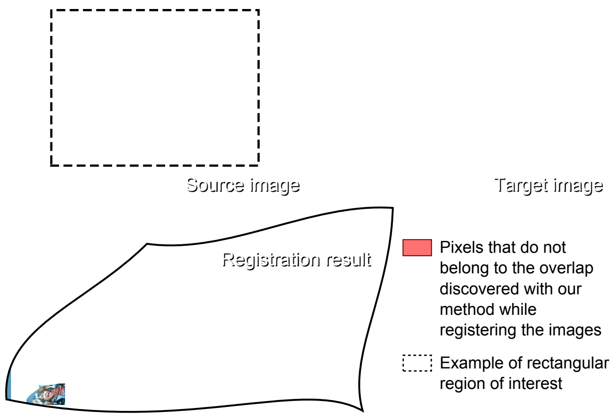

Source image
Target image
Registration result
Pixels that do not belong to the overlap discovered with our method while registering the images
Example of rectangular region of interest

Figure 5.4: We propose a new algorithm that does not require one to define a region of interest (ROI). Our algorithm discovers the exact overlap between two images while registering them. Using the rectangular ROI in dashed line defeats classical methods since it contains pixels that do not belong to the overlap.

The overlap and the RoI. The direct approach to image registration is interesting because it does not rely on feature correspondences. However, standard registration algorithms require a ROI $\mathfrak{R}$ included in the overlap of the images. This is a difficult 'chicken-and-egg' problem since the overlap can only be determined after a successful registration. There is no known satisfactory solution to this problem.

Let $\mathfrak{O}_\mathcal{S}$ be the image *overlap*, *i.e.* the set of pixels of the source image that are also seen in the target image:

$$\mathfrak{O}_\mathcal{S} = \left\{ \mathbf{q} \in \Omega_\mathcal{S} \mid \underline{\mathbf{q}}' \in \Omega_\mathcal{T} \text{ and } \mathcal{S}(\mathbf{q}) \approx \mathcal{T}(\underline{\mathbf{q}}') \right\}, \tag{5.15}$$

where $\underline{\mathbf{q}}'$ is the pixel $\mathbf{q}$ transformed with the true deformation between $\mathcal{S}$ and $\mathcal{T}$. It is obvious that the cost function equation (5.13) or equation (5.14) cannot be evaluated at those pixels that do not belong to $\mathfrak{O}_\mathcal{S}$. As a consequence, $\mathfrak{R}$ must be included in $\mathfrak{O}_\mathcal{S}$, otherwise the registration algorithms based on equation (5.13) or equation (5.14) will fail. Besides, it is better to have a ROI as large as possible in order to have the greatest quantity of information to estimate the warp. The problem here is that the real overlap $\mathfrak{O}_\mathcal{S}$ is known only after a successful registration of the images.

Previous work. As we will review in section 5.2.2.1, the ROI is often a polygonal region in the source image defined either by the user or by some *ad hoc* means (Bartoli, 2008a). This lacks automatism and may be unreliable. The adaptive ROI (Pires and Aguiar, 2004) is another approach. It considers the entire domain of the source image as an initial ROI and updates it during the optimization process. As we will review in section 5.2.2.2, the cost function of (Pires and Aguiar, 2004) is extremely hard to minimize and has global minima that do not correspond to the correct solution (see figure 5.5).

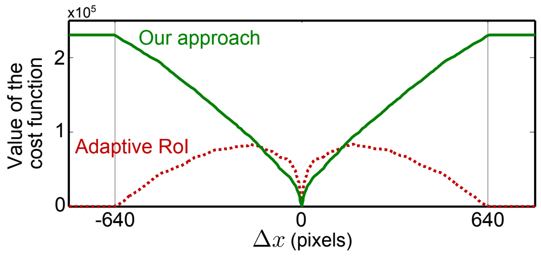


Figure 5.5: Profile of the cost functions of the adaptive ROI approach of (Pires and Aguiar, 2004) (red dashed curve) and our approach (green solid curve). The source and the target images are 640 pixels wide. The simulated warp is a translation along the x-axis parametrized by Δx (more details in section 5.2.2.2). The cost function of (Pires and Aguiar, 2004) vanishes for a warp that creates no overlap while our cost function has only one global minima that corresponds to the true translation (*i.e.* $\Delta x = 0$).

Contribution. We propose a novel approach to direct image registration. It is fundamentally different from standard approaches in that it does not need a ROI. This is made possible by considering the off-target pixels as outliers; the theoretical foundations of this principle are explained in section 5.2.3. The cost function we propose to optimize takes into account *all* the pixels of the source image. A fixed penalty that corresponds to the one given to usual outliers is associated to the off-target pixels. We then use the standard robust M-estimation framework of equation (5.14) to handle both the usual outliers and the off-target pixels in a unified way. Our new approach has several advantages. First, the proposed cost function does not have trivial minima (see figure 5.5). Second, it solves all the above-mentioned problems related to the ROI. Third, the overlap is automatically obtained as the set of inliers.

5.2.2 Region of Interest: State of the Art

5.2.2.1 Rectangular Region of Interest

A common approach used to define the ROI consists in guessing a maximal per-pixel displacement. The ROI is then chosen as a rectangle obtained by removing to the source image domain a margin of width larger than the hypothesized maximal displacement. Ideally, the width of this margin should be as close as possible to the actual maximal displacement, rarely known before registration. The margin width is commonly overestimated so that the optimization algorithm will not fail. Nonetheless, a large ROI provides more information to estimate the warp accurately. Moreover, the size of the ROI affects the profile of the cost function in equation (5.5). A simple experiment inspired by (Pires and Aguiar, 2004) illustrates this phenomenon. Figure 5.6 shows, for different margin sizes, the evolution of the cost function *versus* a single shift parameter Δx (the amplitude of a translation along the x-axis). The source and the target images are identical except for a Gaussian noise with standard deviation equal to 5% of the maximal pixel value. Figure 5.6 shows that a small margin (a large ROI) results in a smooth cost function but has dramatically restricted range of admissible translations. Using a larger margin (a smaller ROI) increases the range of possible translations but creates a lots of local minima in the cost function.

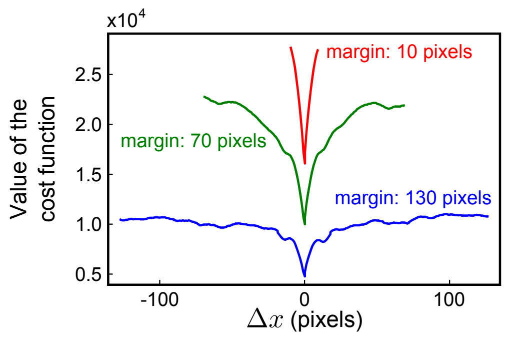


Figure 5.6: Profile of the data term of equation (5.5) for rectangular ROI with margins ranging from 10 to 130 pixels (for images of size 640×480).

	Large margin (small RoI)	Small margin (large RoI)
Range of admissible transformations	+	−
Quantity of information available for the registration	−	+

Table 5.1: Respective advantages and disadvantages of the large and small margins. Note that neither of them has all the advantages.

5.2.2.2 Adaptive Region of Interest

An alternative to the rectangular ROI has been proposed in (Pires and Aguiar, 2004). In this approach, the fixed ROI $\mathfrak{R}$ is replaced by an adaptive ROI $\mathfrak{R}_A(\mathbf{p})$. The minimization problem thus becomes:

$$\min_{\mathbf{p}} \sum_{\mathbf{q} \in \mathfrak{R}_A(\mathbf{p})} \left(\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathcal{W}(\mathbf{q};\mathbf{p}))\right)^2 . \tag{5.16}$$

For a given set of parameters $\mathbf{p}$, $\mathfrak{R}_A(\mathbf{p})$ contains all the pixels (except for a 1-pixel margin used to compute the target image derivatives by finite differences) from the source image that, once warped, belongs to the domain of the target image, *i.e.* $\mathfrak{R}_A(\mathbf{p}) = \{\mathbf{q} \in \Omega_\mathcal{S} \mid \mathbf{q}' \in \Omega_\mathcal{T}\}$ with $\mathbf{q}' = \mathcal{W}(\mathbf{q};\mathbf{p})$. Although this method does not require one to define a ROI, it is not fully satisfactory. First, problem (5.16) is badly posed in the sense that there exists an infinite number of minima that do not correspond to the correct warp parameters. These minima appear when there is no overlap between the source and the warped target images. This fact is illustrated with an experiment similar to the one used in section 5.2.2.1. We observe in figure 5.5 that the cost function of problem (5.16) is null (and thus minimal) as soon as the domains do not overlap ($|\Delta x| > 640$). Second, the fact that $\mathfrak{R}_A(\mathbf{p})$ depends on $\mathbf{p}$ makes problem (5.16) hard to solve rigorously. The authors of (Pires and Aguiar, 2004) propose to neglect the dependency on $\mathbf{p}$ and alternate the estimation of $\mathfrak{R}_A$ and $\mathbf{p}$. Third, the adaptive ROI algorithm is not robust to outliers and, as such, it cannot properly handle occlusions and specularities.

5.2.3 Direct Image Registration without Region of Interest

We propose a new method to direct image registration that does not need a ROI. It thus avoids the above mentioned problems related to the ROI. Our new cost function uses all the pixels of the source image, as the adaptive ROI of (Pires and Aguiar, 2004). However, as the example of figure 5.6 shows, our cost function has no trivial minima. We will show that it is also much easier to optimize rigorously. The key idea of our method is to penalize the off-target pixels with a fixed cost. The cost associated to the other pixels remains the usual robust colour discrepancy of equation (5.14). To some extent, this maximizes the size of the overlap between the two images. We use the same penalty for the off-target pixels and the outlying pixels, for reasons explained below.

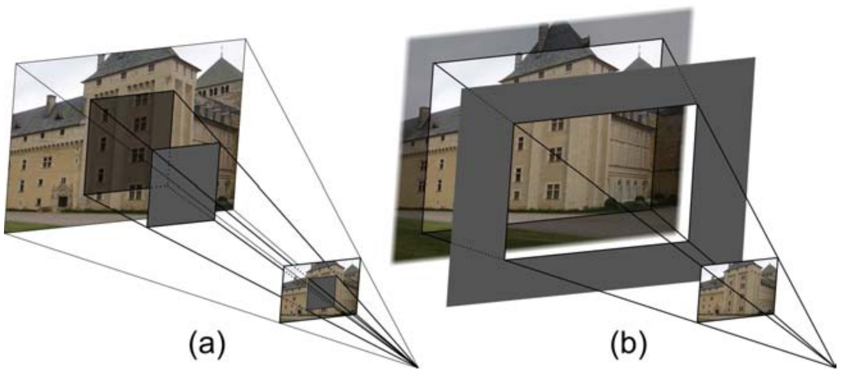


Figure 5.7: Pixels out of the field of view (b) can be considered as usual outliers (a).

Derivation. Imagine a target camera with an unbounded field of view. Such a camera would produce images with an infinite domain. Imagine now that a plane with a rectangular hole is placed between the camera and the observed scene, as figure 5.7 (b) illustrates. The part of the scene visible through the hole corresponds to the actual target image $\mathcal{T}$. The rest of the scene is not seen because it is *occluded* by the plane, exactly as for the pixels hidden by an external occluder, as shown in figure 5.7 (a). With this reasoning, it becomes natural for one to handle off-target pixels as usual outliers.

A direct yet incomplete mathematical statement of our idea is the following minimization problem:

$$\min_{\mathbf{p}} \sum_{\substack{\mathbf{q}\in\Omega_{\mathcal{S}}\\ \mathbf{q}'\in\Omega_{\mathcal{T}}}} \rho\big(\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathbf{q}')\big) + \sum_{\substack{\mathbf{q}\in\Omega_{\mathcal{S}}\\ \mathbf{q}'\notin\Omega_{\mathcal{T}}}} \frac{c^2}{6}. \tag{5.17}$$

Here, we consider that the M-estimator is the Tukey's M-estimator, detailed in section 3.1.2.2. As a reminder, the ρ-function of this M-estimator is:

$$\rho(x) = \begin{cases} \frac{c^2}{6}\left(1 - \left(1 - \frac{x^2}{c}\right)^3\right) & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{otherwise,} \end{cases} \tag{5.18}$$

where c is a constant tuning the sensitivity of the M-estimator to outliers. We use this particular M-estimator because it has the interesting property of being saturated, *i.e.* it is constant after a certain threshold (c). Note that the second term in equation (5.17) corresponds to the value given to outliers by the Tukey's M-estimator. Solving problem (5.17) is difficult since two sums are mixed, with a number of terms varying as a function of $\mathbf{p}$ since $\mathbf{q}' = \mathcal{W}(\mathbf{q};\mathbf{p})$. First of all, we rewrite the fixed penalty term:

$$\min_{\mathbf{p}} \sum_{\substack{\mathbf{q}\in\Omega_{\mathcal{S}}\\ \mathbf{q}'\in\Omega_{\mathcal{T}}}} \rho\big(\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathbf{q}')\big) + \sum_{\substack{\mathbf{q}\in\Omega_{\mathcal{S}}\\ \mathbf{q}'\notin\Omega_{\mathcal{T}}}} \rho(x_0), \tag{5.19}$$

where x_0 is any value saturating the M-estimator: $\rho(x_0) = \frac{c^2}{6}$. With the bisquare ρ-function, any value x_0 such that $|x_0| \geq c$ is suitable (see equation (5.18)). Problem (5.19) can be rewritten:

$$\min_{\mathbf{p}} \sum_{\mathbf{q} \in \Omega_{\mathcal{S}}} \rho\Big([\mathbf{q}' \in \Omega_{\mathcal{T}}]\big(\mathcal{S}(\mathbf{q}) - \mathcal{T}(\mathbf{q}')\big) + [\mathbf{q}' \notin \Omega_{\mathcal{T}}] x_0\Big), \tag{5.20}$$

where [ ] is the operator such that $[a] = 1$ if a is true and $[a] = 0$ otherwise. We rewrite problem (5.20) by introducing the image $\mathcal{T}_\infty$:

$$\mathcal{T}_\infty(\mathbf{q}) = \begin{cases} \mathcal{T}(\mathbf{q}) & \text{if } \mathbf{q} \in \Omega_{\mathcal{T}} \\ \alpha & \text{otherwise} \end{cases} \qquad \text{and} \qquad \Omega_{\mathcal{T}_\infty} = \mathbb{R}^2, \tag{5.21}$$

where α is any value such that $\|\mathcal{S}(\mathbf{q}) - \alpha\| > x_0$. Finally, our method is to solve:

$$\min_{\mathbf{p}} \sum_{\mathbf{q} \in \Omega_{\mathcal{S}}} \rho\big(\mathcal{S}(\mathbf{q}) - \mathcal{T}_\infty(\mathbf{q}')\big). \tag{5.22}$$

Problem (5.22) is solved with standard Iteratively Reweighed Least-Squares (see section 2.2.2.11) or with a more generic algorithm such as Newton's method (see section 2.2.2.4).

M-estimator and overlap. An interesting property of our approach is that it automatically discovers the overlap. For instance, with Tukey's bisquare M-estimator, a pixel $\mathbf{q}$ such that $\left(\left|\rho\big(\mathcal{S}(\mathbf{q}) - \mathcal{T}_\infty(\mathbf{q}')\big) - \frac{c^2}{6}\right| \leq \varepsilon\right)$ can be considered as an outlier (with ε a small constant, *e.g.* 10^{-4}). The overlap in the source image is the set of source pixels satisfying this condition. The overlap in the target image is the warped source overlap. Recovered overlaps are illustrated in figure 5.4 and in section 5.2.4.2.

5.2.4 Experimental Results

5.2.4.1 Synthetic Data

Data generation. We generated synthetic data in the following manner. First, a warp (homography or B-spline) is determined by interpolating some randomly generated point correspondences. The source image is obtained by unravelling a texture image with the previously computed warp and the texture image is used as the target image. The average distance between the point correspondences controls the warp magnitude γ (in pixels). A proportion α of the source and target images is then replaced with data from a different image to simulate occlusions. Last, Gaussian noise with standard deviation σ is added to the images. We used colour images with intensities coded with real values between 0 and 1. The images are 320×240 pixels wide. Figure 5.8 gives an illustration of the generation process.

Experimental setup. The influence of several factors is studied: the transformation magnitude γ, the amount of noise σ and the proportion of erroneous data α. Each one of these factors is studied independently with default values: $\gamma = 8$ pixels, $\alpha = 10\%$ and $\sigma = 0.1$ (10% of the maximal pixel intensity value). Several algorithms are compared: rectangular ROI (RECT), the adaptive ROI of Pires and Aguiar (2004) (ADAP) and our approach (MAXC). Different variants of the RECT algorithm are considered: narrow (10%) and large (25%) margins without M-estimator (RECTN, RECTL) and with M-estimator (RECTNM, RECTLM). The reported results are averages over 100 trials.

Optimization failures. As explained in section 5.2.2.1, a ROI of fixed size can lead to a failure of the optimization process. Figure 5.9 shows in which proportion such failures occur for the experiments of the next 3

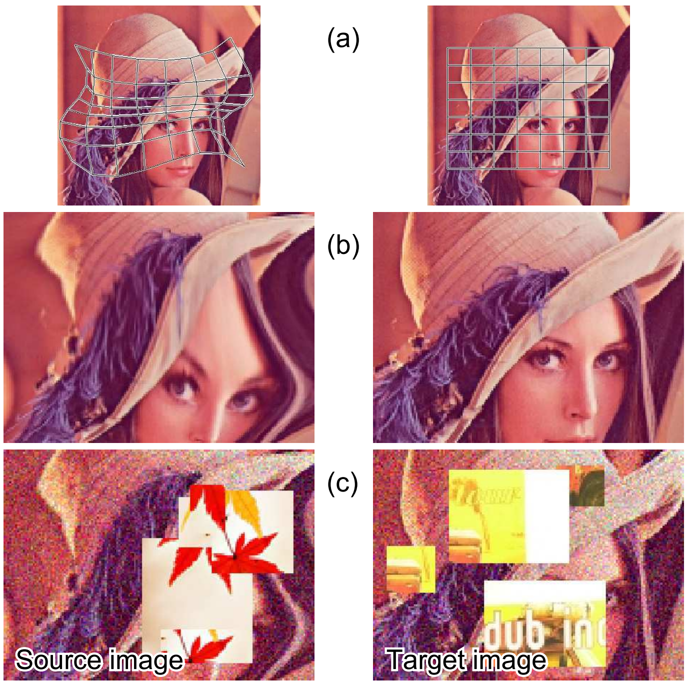


Figure 5.8: Synthetic data generation. (a) Texture image and deformation used to generate the source and the target images. (b) The warp is unravelled to generate the source image. (c) Noise and outliers are added to the images.

paragraphs and for the default values. Note that convergence towards a false solution (local minimum) is not counted as a failure. We observe that there are more failures with a wide rectangular ROI (RECTN) than with a small one (RECTL). There are less failures with an M-estimator (RECTNM, RECTLM) than without because the steps of the optimization algorithms tend to be smaller. In the sequel, when an algorithm fails to converge, the reported measurements are from the last valid iteration.

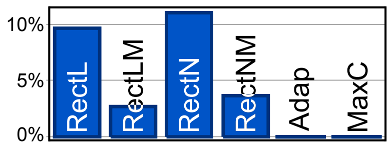


Figure 5.9: Failure rates. ADAP and MAXC never fail because they do not rely on a fixed ROI.

Number of iterations. Figure 5.10 shows the number of iterations. Overall, the convergence is faster with the homographic than with the B-spline warp. This comes from the fact that the homographic warp is global. The apparent rapidity of the algorithms relying on a rectangular ROI stems from the fact that these algorithms can fail before convergence when the given ROI is not valid. Our approach, MAXC is generally better than ADAP which is the only other method that does not require a ROI. However, MAXC takes more iterations to converge when the transformation magnitude is large. This is explained by the fact that many pixels from the source image, once transformed, do not belong to the target image domain. The convergence is slightly slowed down since these pixels are penalized with our approach.

Geometric error. Figure 5.11 shows the geometric error, the discrepancy in pixels between the estimated and the ground truth transformations. We observe that the amount of noise does not influence much the performance of the algorithms. On the contrary, the geometric error is influenced by the transformation magnitude and by the proportion of outliers. This is especially true for the approaches that do not include an M-estimator. Compared to the other methods, our approach is the one that gives the best results. We can see that, with our approach,

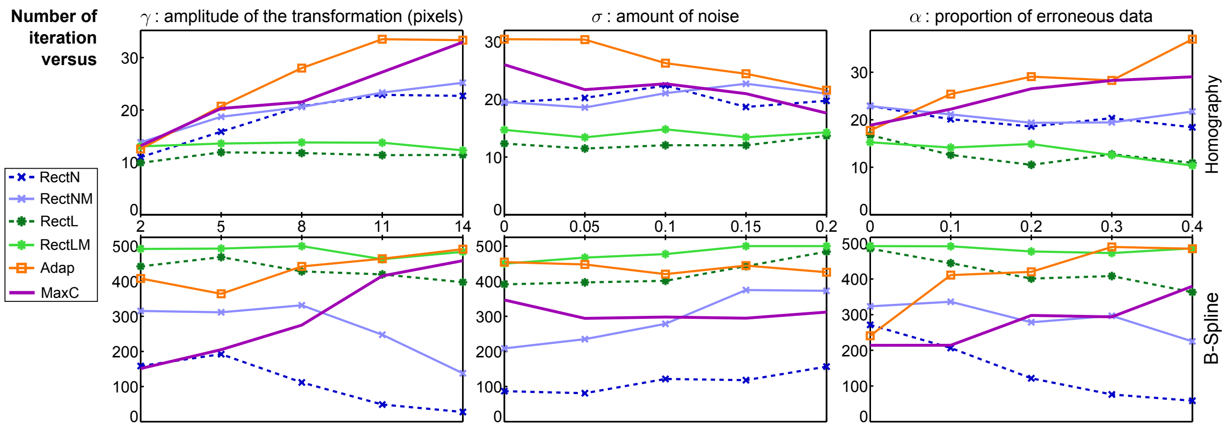


Figure 5.10: Influence of several factors on the the number of iterations. The number of iterations done by the algorithms based on a rectangular ROI is relatively low because these methods can stop prematurely (fail) as soon as the ROI is not valid.

the geometric error is often less than one pixel. This result is particularly important because it shows that our approach is not biased by the penalty term used for the pixels which are warped outside of the target domain.

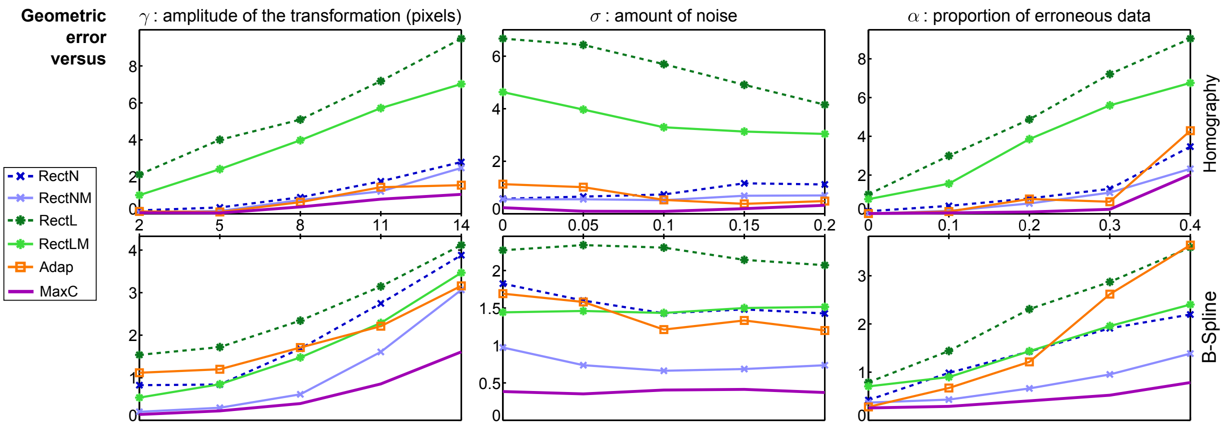


Figure 5.11: Influence of several factors on the geometric error. Our approach (MAXC) gives the best results. Globally, the approaches relying on M-estimators are the best ones.

Photometric error. The average photometric error obtained after the last iteration of the studied algorithms is reported in Figure 5.12. The smallest errors are always obtained with our approach whatever the varying factor and the geometric transformation are.

5.2.4.2 Real Data

Overlap. We consider a source and a target images of a planar scene taken from two different view points and with an occlusion in the target image. Under these conditions, the warp between the two images is a homography. Figure 5.13 shows the ROI used during the last iteration of four different algorithms. This ROI is shown in both the source and the target images. The difference image between the warped target and the source images is also shown. It shows that our approach, MAXC, is the only one to estimate the correct homography. The main point of figure 5.13 is that the final ROI determined with MAXC corresponds exactly to the true overlap between the images. The ROI used by ADAP at convergence does not take into account the occluder. Consequently, ADAP is not able to recover correctly the homography. The ROI utilized by RECTLM does not contain enough pixels making this approach unable to determine precisely the homography. Finally, the algorithm RECTNM fails to converge since its ROI contains pixels that do not belong to the overlap (figure 5.13 shows the last valid iteration).

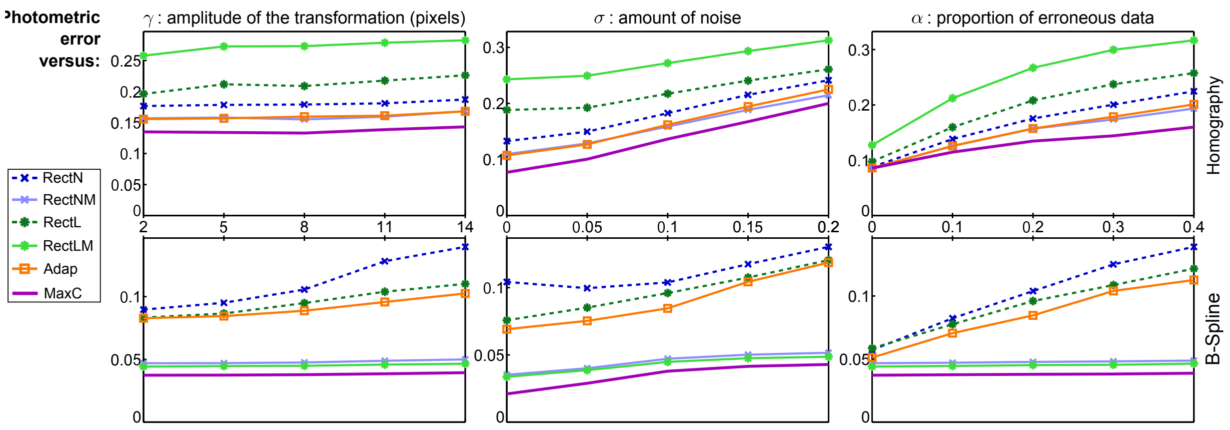


Figure 5.12: Influence of several factors on the photometric error. The best results are always obtained with our approach whatever the transformation model and whatever the studied factor.

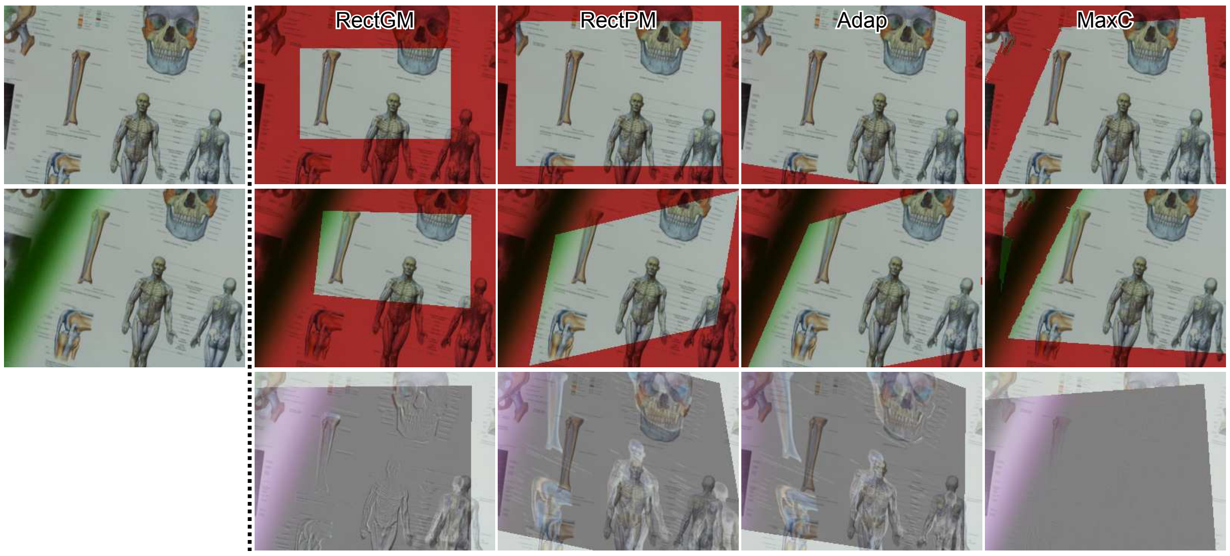


Figure 5.13: Examples of registration results for different algorithms. The first row corresponds to the source image, the second row to the target image, and the last row to the difference between the source and the warped target image. The red pixels are the pixels not included in the ROI during the last iteration of the algorithms. Note that the ROI computed by our approach (MaxC) corresponds to the true overlap (taking into account both the field of view and the occluder). Our approach is the only one that successfully registers this pair of images.

The widest panorama. We consider a video captured by a camera that rotates around its optical centre with a uniform movement from left to right. Consequently, the successive images are linked with homographies. The goal of this experiment is to build a panorama as wide as possible by taking the first image of the video and the furthest image for which the registration is successful. As shown in figure 5.14, the widest panoramas are obtained with ADAP and MAXC. For this video, there are no occluders and, thus, the results of ADAP and MAXC are similar. The algorithms RECTN and RECTL get the smallest panoramas since the maximal displacements are dictated by margin sizes.

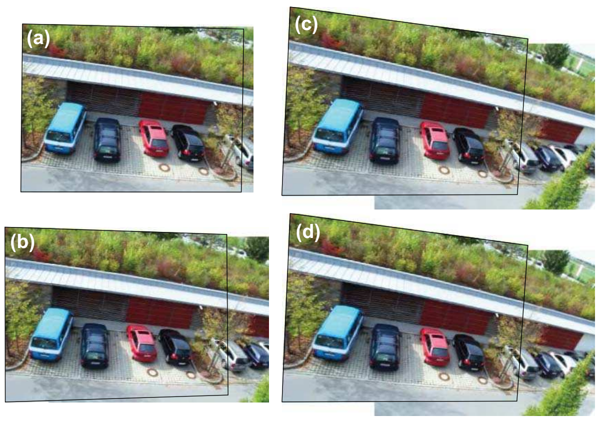


Figure 5.14: Panorama calculated with (a) RECTN, (b) RECTL, (c) ADAP and (d) MAXC. The widest panoramas are obtained with ADAP and our approach: MAXC.

Deformable mosaic. An example of deformable registration using our method is given in figure 5.15. This figure illustrates that our approach automatically retrieves the true overlap in both the source and the target images. Note that a video corresponding to that example is provided as supplemental material.

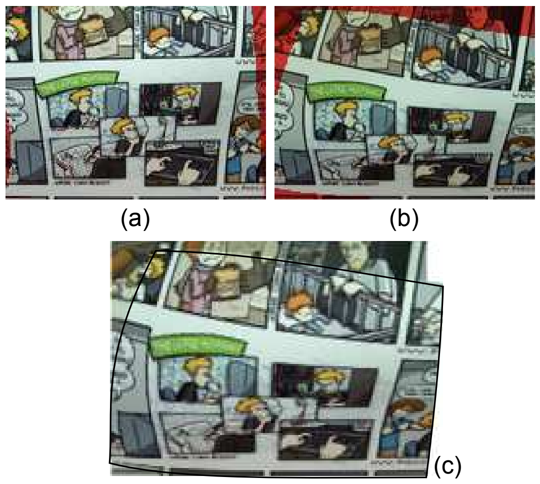


Figure 5.15: Example of deformable mosaic. (a): source image ; (b): target image ; (c) mosaic. The red pixels in (a) and (b) are the pixels that do not belong to the overlap determined with our approach.

Pattern tracking. Figure 5.16 illustrates the tracking of a pattern in a video sequence. Three approaches are compared: our approach, and two approaches using a fixed rectangular ROI (defined with either a large margin or a narrow margin). The object to track is a deforming banknote. We thus use a FFD warp with 5×5 control points. The pattern (*i.e.* the source image) to track is defined as a part of the first image in the video sequence. The pattern is registered in each new image (which plays the role of the target image) using as an initial solution the registration determined for the previous image. Figure 5.16 shows that the approaches relying on fixed ROI fail as soon as a part of the pattern is not visible in the target image. Such problems cannot happen with our approach. Figure 5.16 also illustrates that, with our approach, the true overlap is correctly determined in both the source (pattern) and the target images. The fourth and fifth columns of figure 5.16 shows that our approach handles erroneous data (occlusions and specularities) and the overlap in a unified manner.

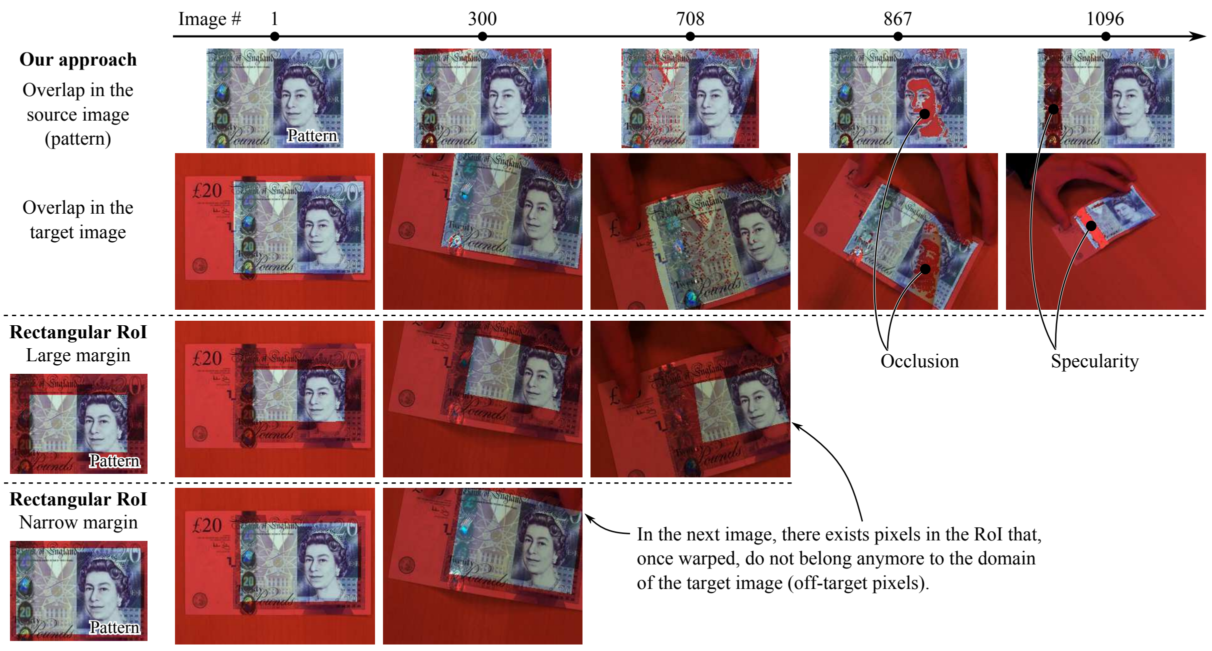


Image #
1
300
708
867
1096
Our approach
Overlap in the source image (pattern)
Pattern
Overlap in the target image
Rectangular RoI
Large margin
Pattern
Rectangular RoI
Narrow margin
Pattern
Occlusion
Specularity
In the next image, there exists pixels in the RoI that, once warped, do not belong anymore to the domain of the target image (off-target pixels).

Figure 5.16: Pattern tracking in a video sequence. Only a few frames of the video are shown here (the complete video is available as supplemental material). For our method (first and second rows), we systematically show the pattern (*i.e.* the source image) in order to illustrate the automatic discovery of the true overlap. For the methods that rely on a fixed rectangular ROI (third and fourth rows), the pattern is shown only once since it does not vary with time. The approaches relying on a fixed ROI fails prematurely because some pixels of the ROI are warped outside of the target image domain (frame #300 with a large margin and #708 with a narrow margin). The frames #867 and #1096 shows how our approach handles occlusions and specularities.

5.2.5 Conclusion

We proposed a new approach to image registration that does not need a ROI. It relies on a theoretical foundation stating that it is possible to consider the off-target pixels as outliers. This new point of view of direct image registration resulted in a slight but elegant modification of the cost function usually optimized. An interesting consequence of our approach is that the true overlap between the images is simply the set of inlying pixels. Compared to previous approaches, ours solves the problems related to the ROI and to the optimization of the cost function. The efficiency of our approach was illustrated with extensive experiments. In particular, we showed that our approach was better than the previous methods in term of accuracy and robustness.

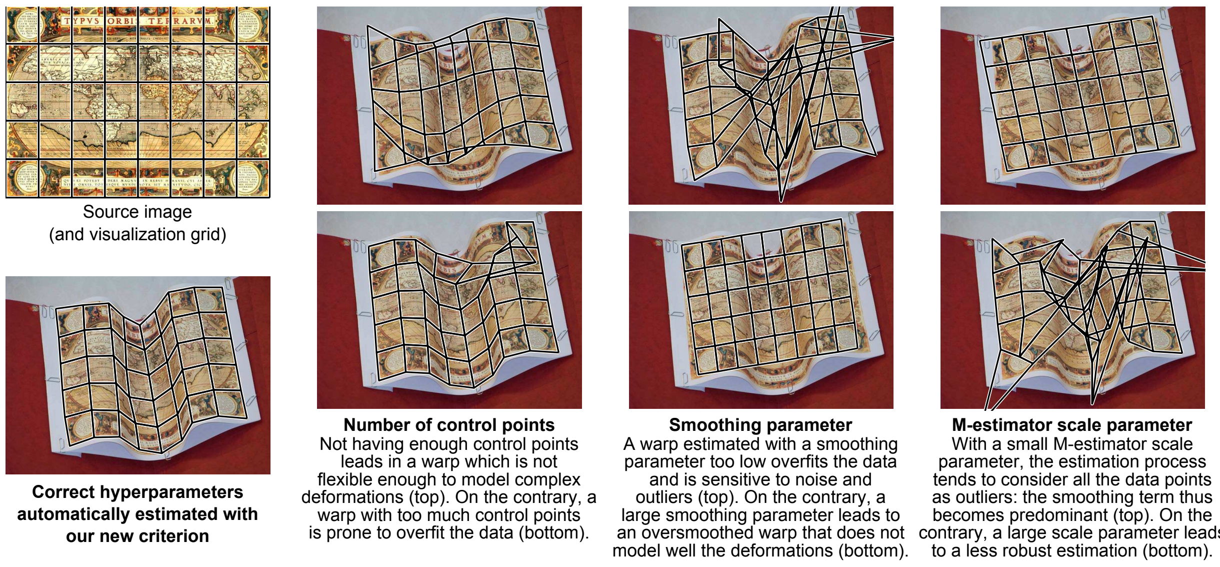
5.3 Pixel-Based Hyperparameter Selection for Feature-Based Image Registration

5.3.1 Introduction

In this section, we deal with parametric image registration from point correspondences in deformable environments. In this problem, it is essential to determine correct values for hyperparameters such as the number of control points of the warp, a smoothing parameter weighting a term in the cost function, or an M-estimator threshold. This is usually carried out either manually by a trial-and-error procedure or automatically by optimizing a criterion such as the Cross-Validation score (see section 3.2.2). In this section, we propose a new criterion that makes use of all the available image photometric information. We use the point correspondences as a training set to determine the warp parameters and the photometric information as a test set to tune the hyperparameters. Our approach is fully robust in the sense that it copes with both erroneous point correspondences and outliers in the images caused by, for instance, occlusions or specularities.

Parametric image registration is the problem of finding the (natural) parameters of a warp such that it aligns a source image to a target image. In addition to these natural parameters, one also has to determine correct values for the problem *hyperparameters* in order to get a proper registration. The hyperparameters are either additional parameters of the warp itself (*warp hyperparameters*) or parameters included in the cost function to optimize (*cost hyperparameters*). As illustrated in figure 5.17, the hyperparameters greatly influence the quality of the estimated warp. As reviewed previously, there are two main approaches to image registration: the feature-based and the pixel-based (or direct) approaches. They both have their own drawbacks and advantages but neither of them directly enables one to automatically tune the hyperparameters. In this section, we propose a new method to automatically set the hyperparameters by combining the advantages of the feature-based and the pixel-based approaches.

As just said, some hyperparameters are linked to the warp. Let $\mathcal{W} : \mathbb{R}^2 \times \mathbb{R}^l \rightarrow \mathbb{R}^2$ be a warp. It is primarily parametrized by a set of l parameters arranged in a vector $\mathbf{s} \in \mathbb{R}^l$. The homography (Hartley and Zisserman, 2003a; Szeliski, 2006) is an example of warp, often parametrized by the 8 independent coefficients of the homography matrix. Another example of warp is the Free-Form Deformation (FFD) (Rueckert et al., 1999) parametrized by $l/2$ two-dimensional control points. Examples of hyperparameters linked to the warps include, but are not limited to, the number of control points of an FFD or the kernel bandwidth of a Radial Basis Function (Bookstein, 1989).



Source image
(and visualization grid)

Correct hyperparameters automatically estimated with our new criterion

Number of control points
Not having enough control points leads in a warp which is not flexible enough to model complex deformations (top). On the contrary, a warp with too much control points is prone to overfit the data (bottom).

Smoothing parameter
A warp estimated with a smoothing parameter too low overfits the data and is sensitive to noise and outliers (top). On the contrary, a large smoothing parameter leads to an oversmoothed warp that does not model well the deformations (bottom).

M-estimator scale parameter
With a small M-estimator scale parameter, the estimation process tends to consider all the data points as outliers: the smoothing term thus becomes predominant (top). On the contrary, a large scale parameter leads to a less robust estimation (bottom).

Figure 5.17: Illustration of how some typical hyperparameters influence image registration. The contribution of this paper is a method able to select the proper hyperparameters by combining the advantages of the feature-based and of the pixel-based approaches to image registration. In this example, the data points were automatically detected and matched with SIFT Lowe (2004); Vedaldi and Fulkerson (2008). There was approximately 200 point correspondences (not shown in the figure) uniformly spread across the source image. Among these points, around 10% were gross outliers.

In this section, we use a slightly different formalism than the one used in the introductory section of this chapter. This is motivated by the fact that we now explicitly consider the hyperparameters bundled in the feature-based approach to image registration. Let $\{\mathbf{p}_i \leftrightarrow \mathbf{q}_i\}_{i=1}^n$ be the point correspondences. We now write the optimization problem of the feature-based image registration as:

$$\min_{\mathbf{s}} \mathcal{E}(\mathbf{s}; \boldsymbol{\theta}), \tag{5.23}$$

where $\boldsymbol{\theta}$ is a vector containing all the hyperparameters and $\mathbf{s}$ is the vector of the natural parameters of the warp. $\mathcal{E}$ is the cost function that may be defined as, for instance:

$$\mathcal{E}(\mathbf{s}; \boldsymbol{\theta}) = \sum_{i=1}^{n} \rho\big(\mathcal{W}(\mathbf{p}_i; \mathbf{s}) - \mathbf{q}_i; \gamma\big) + \lambda \mathcal{R}(\mathbf{s}), \tag{5.24}$$

with ρ an M-estimator, γ its scale parameter, $\mathcal{R}$ a smoothing term such as the classical bending energy term discussed in section 5.1.2.1 (equation (5.11)) and λ a smoothing term controlling the trade-off between goodness-of-fit and smoothing. In equation (5.24), γ and λ are two examples of cost hyperparameters. Note that other hyperparameters can appear in the cost function if one decides, for instance, to use more terms. The main advantages of the feature-based approach to image registration are that it copes with large deformations and it is efficient in terms of computational complexity (this is particularly true when using an efficient keypoint detector such as SIFT (Lowe, 2004) or SURF (Bay et al., 2008) combined with a good matching algorithm such as the improved nearest neighbour algorithm suggested in (Lowe, 2004) and implemented in (Vedaldi and Fulkerson, 2008)). However, the feature-based approach by itself does not enable one to determine correct hyperparameters. It is *not possible* to determine proper values for the hyperparameters by including them directly in the optimization problem (5.23), *i.e.* $\min_{\mathbf{s},\boldsymbol{\theta}} \mathcal{E}(\mathbf{s}; \boldsymbol{\theta})$. This comes from the general arguments that were developed in section 3.2.

The other approach to image registration is the *direct approach* (Baker et al., 2004; Irani and Anandan, 1999). In this case, the warp parameters are estimated by minimizing the pixel-wise dissimilarities between the source image and the warped target image. One advantage of this approach is that the data used for the parameter estimation is denser than with the feature-based approach. As in the feature-based approach, it is not possible to estimate the hyperparameters with the direct approach.

Since the hyperparameters cannot be trivially estimated, they are often fixed once and for all according to some empirical (and often unreliable) observations. It is also possible to choose them manually with some kind of trial-and-error procedure. This technique is obviously not satisfactory because of its lack of automatism and of foundations. Several approaches have been proposed to tune the hyperparameters in an automatic way. None of them is specific to image registration. They generally minimize a criterion that depends on the hyperparameters and that assesses the 'quality' of the estimated parameters by measuring the ability of the current estimate to generalize to new data. They include, but are not limited to, Akaike Information Criterion (Cetin and Erar, 2002), Mallow's C_P (Ronchetti and Staudte, 1994), Minimum Description Length criterion, and the techniques relying on Cross-Validation scores (Bartoli, 2008b; Brabanter et al., 2003; Wahba and Wold, 1975). These types of approach were reviewed in section 3.2.2.

The common characteristic of these approaches to automatically select the hyperparameters is that they are problem generic and, as a consequence, they all rely on the point correspondences only. In the particular context of feature-based image registration, another type of data is available: the *photometric information*. We thus propose a new criterion, named the *photometric criterion*, that uses the point correspondences as a training set and the pixel colours as a test set. Another way to put it is to say that our approach combines the two classical approaches to image registration: roughly speaking, the feature-based approach is used to estimate the natural parameters while the pixel-based approach is used for the hyperparameters. Our photometric criterion is more flexible than the previous approaches in the sense that it can handle simultaneously several hyperparameters of different types (for instance, discrete and continuous hyperparameters can be mixed together). Besides, our approach is much more robust to erroneous data (noise and outliers) than previous approaches based on Cross-Validation. Also, it still works when there are only a few point correspondences. Our new criterion is explained in section 5.3.3 and its ability to properly tune several hyperparameters simultaneously is experimented in section 5.3.4 with B-spline warps and the Cauchy M-estimator.

5.3.2 Reminder and Complementary Elements on Automatic Hyperparameter Selection

We presented several hyperparameters in the introduction of this section. It is important to understand that inconsistent results would arise if one tries to estimate the hyperparameters by including them in the optimization problem (5.23). For instance, with such an approach, the best way to minimize the contribution of the regularization term would be to set $\lambda = 0$ which is obviously not the desired value. All the same way, making the M-estimator scale parameter γ tend to 0 would 'artificially' decrease the value of the cost function because it would be equivalent to consider that almost all the point correspondences are outliers (and the cost assigned to outliers tends to zero when $\gamma \to 0$).

As we explained in section 3.2, the classical approach to build an automatic procedure for selecting the hyperparameters consists in designing a criterion $\mathcal{C}$ that assesses the 'quality' of a given set of hyperparameters (Bartoli, 2008b; Wahba, 1990). The minimizer of this criterion should be the set of hyperparameters to use. The complete problem thus consists in solving the following nested optimization problem:

$$\min_{\mathbf{s}} \mathcal{E}(\mathbf{s}; \arg\min_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta})). \tag{5.25}$$

Note that the introduction of the criterion $\mathcal{C}$ makes the problem (5.25) completely different from the inconsistent problem $\min_{\mathbf{s},\boldsymbol{\theta}} \mathcal{E}(\mathbf{s}; \boldsymbol{\theta})$.

5.3.2.1 Cross-Validation

Cross-Validation (hereinafter abbreviated CV) is a general principle used to tune the hyperparameters in parameter estimation problems (Wahba, 1990). Broadly speaking, a CV procedure consists in minimizing a score function that measures how well a set of estimated parameters will generalize to new data. This is achieved by dividing the whole data set into several subsets. Each one of these subsets is then alternatively used as a training set or as a test set to build the CV score function. The use of CV to select the hyperparameters for spline parameter estimation has been introduced in (Wahba and Wold, 1975). It has been successfully applied for deformable warp estimation from point correspondences in (Bartoli, 2008b). We now give a reminder of two variants of CV which allow us to fix the notation used in this section: the *Ordinary* CV and the *V-fold* CV.

Ordinary CV (OCV). For a given set of hyperparameters $\boldsymbol{\theta}$, let $\mathbf{s}_{\boldsymbol{\theta}}^{(k)}$ be the warp parameters estimated from the data with the k-th point correspondence left out. The OCV score, denoted $\mathcal{C}_{OCV}$, is defined by:

$$\mathcal{C}_{OCV}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{k=1}^{n} \left\| \mathbf{q}_k - \mathcal{W}\left(\mathbf{p}_k; \mathbf{s}_{\boldsymbol{\theta}}^{(k)}\right) \right\|^2 . \tag{5.26}$$

Tuning the hyperparameters using the OCV consists in minimizing $\mathcal{C}_{OCV}$ with respect to $\boldsymbol{\theta}$. This approach has several drawbacks. First, computing $\mathcal{C}_{OCV}$ is prohibitive: evaluating $\mathcal{C}_{OCV}$ for a single $\boldsymbol{\theta}$ with equation (5.26) requires to estimate each one of the n vectors $\{\mathbf{s}_{\boldsymbol{\theta}}^{(k)}\}_{k=1}^{n}$. There exists some close approximations of equation (5.26) resulting in a significant improvement in terms of computational time. However, these approximations are only usable in a least-squares framework for parameter estimation (see, for instance, (Bartoli, 2008a; Farenzena et al., 2008)). Second, the score $\mathcal{C}_{OCV}$ is not robust to false point correspondences. And last, but not least, the OCV score is not reliable when there are not enough point correspondences (Wahba, 1990).

V-Fold Cross-Validation (V-fold CV). An alternative to the OCV score is the V-fold CV score. A complete review of the V-fold CV is given in Brabanter et al. (2003). It consists in splitting the set of point correspondences into V disjoint sets of nearly equal sizes (with V usually chosen as $V = \min(\sqrt{n}, 10)$). Let $\mathbf{s}_{\boldsymbol{\theta}}^{[v]}$ be the warp parameters obtained from the data with the v-th group left out and let m_v be the number of point correspondences in the v-th group. The V-fold CV score, denoted $\mathcal{C}_V$, is defined by:

$$\mathcal{C}_V(\boldsymbol{\theta}) = \sum_{v=1}^{V} \frac{m_v}{n} \sum_{k=1}^{m_v} \frac{1}{m_v} \left\| \mathbf{q}_k - \mathcal{W}\left(\mathbf{p}_k; \mathbf{s}_{\boldsymbol{\theta}}^{[v]}\right) \right\|^2. \tag{5.27}$$

The V-fold CV is not robust to erroneous point correspondences. It can be made robust by replacing the average $\sum_{k=1}^{m_v} \frac{1}{m_v} \left\| \mathbf{q}_k - \mathcal{W}\left(\mathbf{p}_k; \mathbf{s}_{\boldsymbol{\theta}}^{[v]}\right) \right\|^2$ in equation (5.27) with some robust measure such as the trimmed mean (Brabanter et al., 2003). Besides, the V-fold CV score is not more reliable than the OCV score when there are only a few point correspondences.

5.3.2.2 Other Approaches

Other approaches such as Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Mallow's C_P, Minimum Description Length (MDL) have been used to tune hyperparameters (see, for instance, (Brabanter et al., 2003; Cetin and Erar, 2002)). Some robust versions also exist for these criteria ; for instance a robust Mallow's C_P is developed in (Ronchetti and Staudte, 1994). However, these criteria have usually been developed to choose one model among a finite set of given models and, as such, approaches based on CV are better suited to tune continuous hyperparameters (Bartoli, 2008b).

5.3.3 Our Contribution: the Photometric Error Criterion

The common characteristic of the approaches reviewed in section 3.2.2 is that both the parameters and the hyperparameters are estimated using exactly the same data set, *i.e.* the point correspondences. In this section, we propose a new criterion to tune hyperparameters that makes use of all the available information: not only the point correspondences but also the photometric information.

The principle of our approach consists in combining the two standard approaches to image registration:

- given a set of hyperparameters $\boldsymbol{\theta}$, the feature-based approach is used to determine the warp parameters $\mathbf{s}_{\boldsymbol{\theta}}$ from the point correspondences ;
- the cost function of the direct approach is used to assess the correctness of the hyperparameters $\boldsymbol{\theta}$: the proper hyperparameters must be the ones minimizing the pixel-wise photometric discrepancy between the target image and the warped source image.

In other words, we propose to use the point correspondences as the training set and the photometric information as the test set. Dividing the data into a training set and a test set is a classical approach of statistical learning (Hastie et al., 2001). Given a vector of hyperparameters $\boldsymbol{\theta}$ and the corresponding warp parameters $\mathbf{s}_{\boldsymbol{\theta}}$ (estimated from the point correspondences), our criterion, denoted $\mathcal{C}_\star$, is defined as:

$$\mathcal{C}_\star(\boldsymbol{\theta}) = \frac{1}{|\mathfrak{R}|} \sum_{\mathbf{p} \in \mathfrak{R}} \left\| \mathcal{S}(\mathbf{p}) - \mathcal{T}(\mathcal{W}(\mathbf{p}; \mathbf{s}_{\boldsymbol{\theta}})) \right\|^2, \tag{5.28}$$

where $\mathfrak{R}$ is the region of interest and $|\mathfrak{R}|$ its size. $\mathfrak{R}$ can be defined as, for instance, a rectangle obtained by cropping the domain of the source image. More advanced techniques such as the one proposed in the previous section could also be used to deal with the region of interest. However, since the region of interest is not the central point of this section, we prefer to keep it as simple as possible.

Note that the criterion of equation (5.28) is a slight variation of the cost function typically minimized in direct image registration (Irani and Anandan, 1999; Szeliski, 2006) (see section 5.1.2.1). The only difference is the normalizing factor $\frac{1}{|\mathfrak{R}|}$. This factor allows one to compensate the undesirable effects due to the overlap problem (see section 5.2). Using this factor in classical direct image registration would make difficult the optimization process since it makes the cost function non-linear with respect to the warp parameters. However, in the context now considered, the criterion of equation (5.28) is not intended to be minimized with respect to the warp parameters. This is the very difference with direct image registration: here, the criterion of equation (5.28) is considered as a function of the hyperparameters $\boldsymbol{\theta}$, *not* of the warp parameters $\mathbf{s}$.

Robustness. When using photometric information, one should take care of the fact that there can be outliers in the image colours caused, for instance, by occlusions or specularities. The criterion $\mathcal{C}_\star$ can be made robust to these outliers by replacing the squared Euclidean norm in equation (5.28) with a more robust measure such as the trimmed mean. We thus define the *robust photometric error criterion*, denoted $\mathcal{C}'_\star$, as:

$$\mathcal{C}'_\star(\boldsymbol{\theta}) = \frac{1}{\frac{100-\alpha}{100}|\mathfrak{R}|} \sum_{\mathbf{p}\in\mathfrak{R}_\alpha} \left\|\mathcal{S}(\mathbf{p}) - \mathcal{T}(\mathcal{W}(\mathbf{p};\mathbf{s}_{\boldsymbol{\theta}}))\right\|^2, \tag{5.29}$$

where $\mathfrak{R}_\alpha$ is the subset of $\mathfrak{R}$ obtained by removing from $\mathfrak{R}$ the $\alpha\%$ of the pixels that produce the highest values for $\|\mathcal{S}(\mathbf{p}) - \mathcal{T}(\mathcal{W}(\mathbf{p};\mathbf{s}_{\boldsymbol{\theta}}))\|^2$ (see section 3.1.2.3).

5.3.4 Experimental Results

5.3.4.1 Technical Details

The contribution we presented in this section is quite generic in the sense that it may be used in many different setup. However, even though our contribution is generic, we use a particular combination of elements for the experiments of this section.

Warp. We use the uniform cubic B-spline model of function for the warp (see section 2.3.2.3 and section 2.3.2.5). This model is interesting since it is generic and may thus represent the transformations of a deformable environment. Besides, this model includes several hyperparameters. The number of control points is one of these hyperparameters (we note l this number in the sequel of the experiments).

Smoothing Term. In order to compute a smooth warp, we use the classical bending energy term as described in section 5.1.2.1 (equation (5.11)). Using such a regularization scheme introduce another hyperparameter: the regularization parameter, which we denote λ in the rest of the experiments.

M-estimator. In this section we use the Cauchy M-estimator defined by the following ρ function:

$$\rho(x;\gamma) = \log\left(1 + \frac{x^2}{\gamma^2}\right), \tag{5.30}$$

where $\gamma \in \mathbb{R}_+^*$ is an hyperparameter that controls the scale of this M-estimator. More details on this M-estimator may be found in section 3.1.2.2. In particular, it can easily be shown that the Cauchy M-estimator is the negative likelihood with errors following a Cauchy/Lorentz distribution. The inaccuracies of the keypoints' locations detected by SURF and SIFT tend to follow such a distribution. Besides, the probability density function (PDF) of the Cauchy distribution has heavy tails that satisfactorily models the outliers, *i.e.* the false point correspondences. We report in figure 5.18 an illustrative test showing that assuming a Cauchy distribution is consistent with the kind of errors encountered in real cases. In this experiment, we use the source and the target images of figure 5.17 for which the ground truth warp is known (manually determined). Figure 5.18 depicts an histogram of the errors between the location of the 1112 keypoints detected with SIFT in the target image and their expected location (computed by applying the ground truth warp to the keypoints in the source image). It shows that considering a Cauchy distribution is a reasonable choice. In particular, the fact that the tails of the PDF of the Cauchy distribution are heavier than the ones of, for instance, the Gaussian PDF makes the cost function of equation (5.23) robust to outliers.

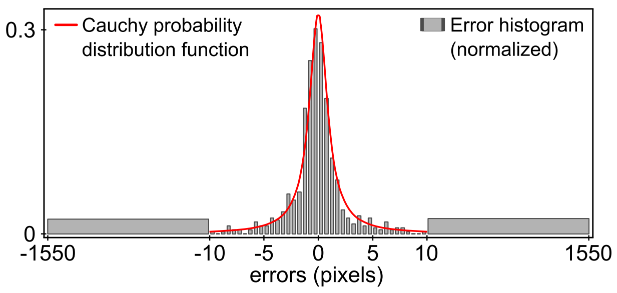


Figure 5.18: Graphical comparison between the probability density function of the Cauchy distribution and the (normalized) histogram of the errors between the expected keypoints in the target image and the keypoints automatically detected with SIFT. Mind the scale of the abscissa axis.

Optimization of the Criteria. All the criteria used in the experiments (including the CV criteria and our new criterion) are minimized using an exhaustive search approach. It consists in evaluating the criteria over a fine grid in order to find the optimum. Although long to compute, this approach has the advantage of being reliable. Besides, we generally optimize over only 2 or 3 hyperparameters, which makes the computational time reasonable.

5.3.4.2 Synthetic Data

In this subsection, several experiments are done on synthetic data. Using such data is interesting since it allows us to know precisely the *ground truth* warp that relates the source and the target images.

Synthetic Data Generation. A pair of images is generated from a texture image (randomly chosen in a stock of 15 different images). A rectangular part of the texture image is used as the source image. The target image is build by deforming another part of the texture image with a ground truth warp $\mathcal{W}^\star$, as illustrated in figure 5.19. The warp $\mathcal{W}^\star$ is a B-spline with 5×5 control points determined randomly and such that the average deformation magnitude is approximately 20 pixels. The sizes of the source and of the target images are 160×160 pixels and 320×240 pixels respectively. A Gaussian noise with standard deviation equal to 5% of the maximal intensity value is added to the pixels of both the source and the target images. A set $\mathcal{P} = \{\mathbf{p}_i \leftrightarrow \mathbf{q}_i\}_{i=1}^n$ of point correspondences is built by randomly picking the points $\mathbf{p}_i$ in the source image and computing their correspondents $\mathbf{q}_i$ in the target image with the warp $\mathcal{W}^\star$. A Cauchy noise with scale parameter $\gamma = 1$ pixel is added to the point correspondences.

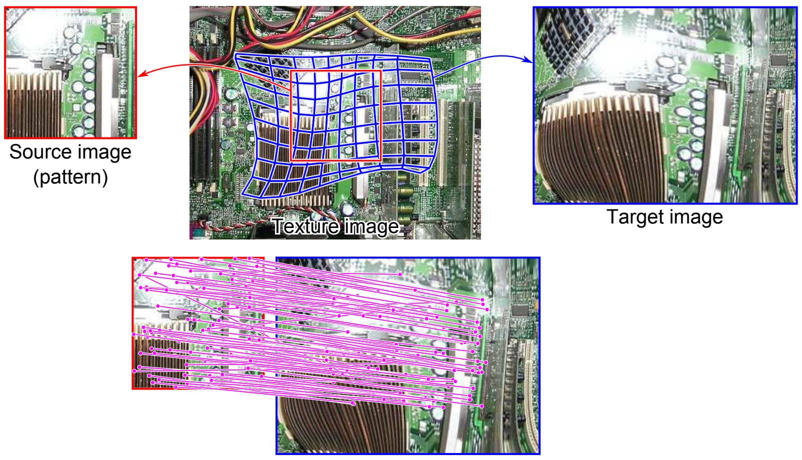


Figure 5.19: Illustration of the generation of synthetic data. First row: a pair of images (the source and the target images) are generated from a texture image using a predetermined warp (used as the ground truth warp). Second row: point correspondences are automatically extracted and matched from the generated images using standard approaches such as SIFT and SURF.

Oracle. We call *oracle* the warp estimated from the point correspondences $\mathcal{P}$ which is as close as possible to the ground truth warp $\mathcal{W}^\star$. It is designed to be the best possible warp given *i)* the available data and *ii)* the warp model. It is preferable to use the oracle instead of the ground truth warp to evaluate an estimated warp. Indeed, en error between an estimated warp and the ground truth warp does not necessarily comes from a bad estimation process (which is the object of our experiments in this paper): it can comes from the fact that the considered warp model is simply not able to fit the ground truth warp (for example, even if the correct hyperparameters are given, a homography will never fit a highly deformed warp). The oracle is defined as the warp induced by the parameters and the hyperparameters $(\mathbf{s}_o, \boldsymbol{\theta}_o)$ estimated by solving the following problem:

$$(\mathbf{s}_o, \boldsymbol{\theta}_o) = \arg\min_{(\mathbf{s},\boldsymbol{\theta})} \iint_{\mathbf{p}\in\Omega_{\mathcal{W}^\star}} \|\mathcal{W}^\star(\mathbf{p}) - \mathcal{W}(\mathbf{p};\mathbf{s})\| \, \mathrm{d}\mathbf{p}. \tag{5.31}$$

Problem (5.31) is numerically solved using an exhaustive search approach.

5.3.4.3 Relative Geometric Error (RGE)

The RGE measures the discrepancy between an estimated warp and the oracle. Let $\boldsymbol{\theta}_\bullet$ be the set of hyperparameters that minimizes the criterion $\mathcal{C}_\bullet$. Let $\mathbf{s}_\bullet$ be the warp parameters estimated from the point correspondences with the hyperparameters $\boldsymbol{\theta}_\bullet$. The RGE is defined as:

$$\iint_{\mathbf{p}\in\Omega_{\mathcal{S}}} \frac{\|\mathcal{W}(\mathbf{p};\mathbf{s}_o) - \mathcal{W}(\mathbf{p};\mathbf{s}_\bullet)\|}{\|\mathcal{W}(\mathbf{p};\mathbf{s}_o)\|} \, \mathrm{d}\mathbf{p}. \tag{5.32}$$

Figure 5.20 compares the RGE obtained by tuning the M-estimator scale parameter γ and the smoothing parameter λ with different approaches:

- our photometric criterion (Photo) and its robust versions with thresholds for the trimmed mean of 25% (PhotoR25) and 50% (PhotoR50) ;
- the V-fold CV criterion (VFold) and its robust versions with thresholds for the trimmed mean of 20% (VFoldR20) and 40% (VFoldR40).

The number of control points of the warp is set to 8×8. 100 point correspondences are used to estimate the warp. The results reported in figure 5.20 are averaged over 100 trials (with different texture images, different point correspondences, and different deformations).

We can observe in figure 5.20 that the smallest RGE are consistently obtained with our photometric criterion. The difference between robust and non-robust versions of our criterion is not as significant as for the CV criteria. This comes from the fact that in the synthetic data used for this experiment, there are outliers in the point correspondences (thus affecting the non-robust CV scores) while the source and the target images are outlier-free.

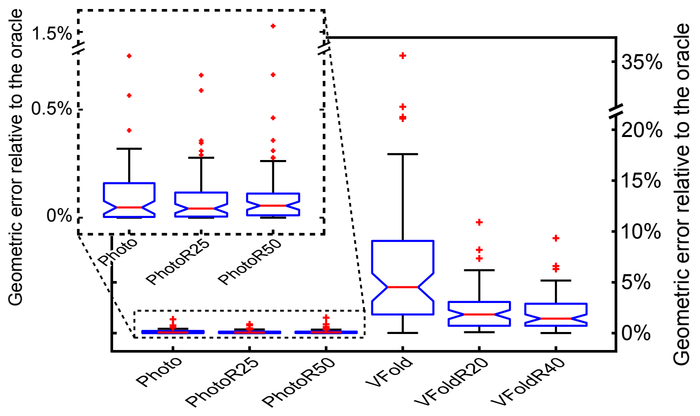


Figure 5.20: Relative geometric errors for several criteria used to determine hyperparameters. Globally, the criteria we propose (Photo, PhotoR25, and PhotoR50) give better results than the ones obtained with criteria relying on Cross-Validation (VFold, VFoldR20, and VFoldR40). The red line is the median over the 100 trials. The limits of the blue box are the 25th and the 75th percentiles. The black 'whiskers' cover approximately 99.3% of the experiment outcomes. The red crosses are the outcomes considered as outliers.

5.3.4.4 Scale Parameter of the Cauchy's M-estimator

Figure 5.21 shows the values determined with several criteria for the Cauchy's M-estimator scale parameter γ. In addition to the criteria used in the previous experiment, we also show the results obtained with the oracle. The data used in this experiment are the same than the one used in the previous experiment. The point correspondences were generated with errors following a Cauchy distribution with scale parameter equals to 1. As a consequence, the criteria are expected to give the value 1 for the scale parameter of the Cauchy's M-estimator. Figure 5.21 shows that the proposed photometric criteria results in values for γ which are close to 1. We observe that the three approaches based on the basic V-fold CV also results in correct values. On the contrary, the robust variants of the V-fold CV gives values farther away from 1 than the other approaches. The fact that the value 1 is not exactly retrieved with our criteria is not really significant since this value is not precisely retrieved with the oracle itself.

5.3.4.5 Noise in the Point Correspondences

In this experiment, we study the influence of the noise in the point correspondences. We use the same data than in the experiments of section 5.3.4.3 except that there are no outliers in the images. The point correspondences are perturbed using an additive Gaussian noise of standard deviation σ varying between 0 and 12 pixels. Therefore, we only test the non-robust methods: VFold and Photo. These methods are used to automatically tune the regularization parameter. Figure 5.22 shows the evolution of the RGE in function of the amount of noise in the point correspondences. It shows that our approach Photo is much more robust to the noise than VFold.

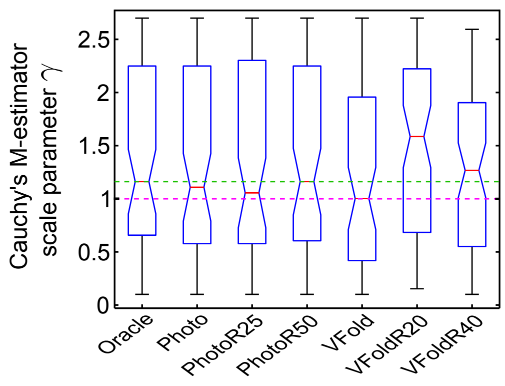


Figure 5.21: Scale parameter of the Cauchy's M-estimator retrieved using several criteria. The pink dashed line represents the expected value for this hyperparameter. The green dashed line represents the value retrieved using the oracle. The use of the criteria we proposed (Photo, PhotoR25, and PhotoR50) results in values close to the expected ones.

This comes from the fact that VFold entirely relies on the noisy point correspondences while our approach also includes colour information.

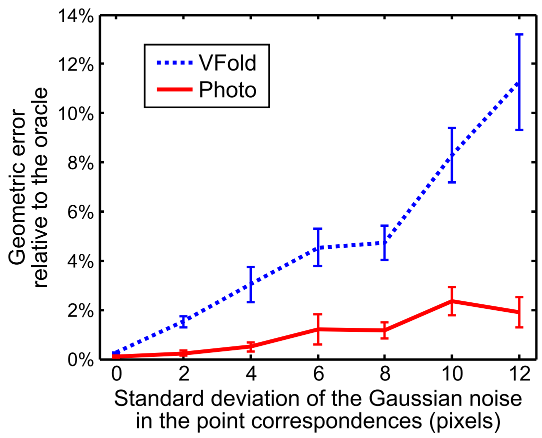


Figure 5.22: Evolution of the relative geometric error in function of the (Gaussian) noise in the point correspondences. Our approach, Photo, is more robust than the approach relying on the CV (VFold).

5.3.4.6 Real Data

The last experiments of this paper are conducted on real data. The source images are digital pictures. The target images are obtained by first printing the source images and second picturing them with a standard camera. Ground truth warps were determined manually by clicking several hundreds of point correspondences in the images. Note that figure 5.17 shows an example of our approach applied to real data.

The cubist image Figure 5.23 shows the registration results obtained by automatically determining the hyperparameters with several criteria. In this experiment, three hyperparameters were considered: the smoothing parameter λ, the M-estimator threshold γ, and the number of control points of the B-spline warp l_x (the number of control points along the x-axis and the y-axis were set to be equal). 314 point correspondences were automatically determined using the SIFT detector and the descriptor matcher implemented in (Vedaldi and Fulkerson, 2008). Approximately 8% of the point correspondences were false matches. We can observe in figure 5.23 that our photometric criterion is the one giving the best results. The standard V-Fold CV criterion is the one leading to the worst results due to the presence of erroneous point correspondences. The robust V-Fold CV criterion performs better than the non-robust one but is not as good as ours, particularly for the bottom right corner of the image: this is due to a lack of point correspondences in this part of the image.

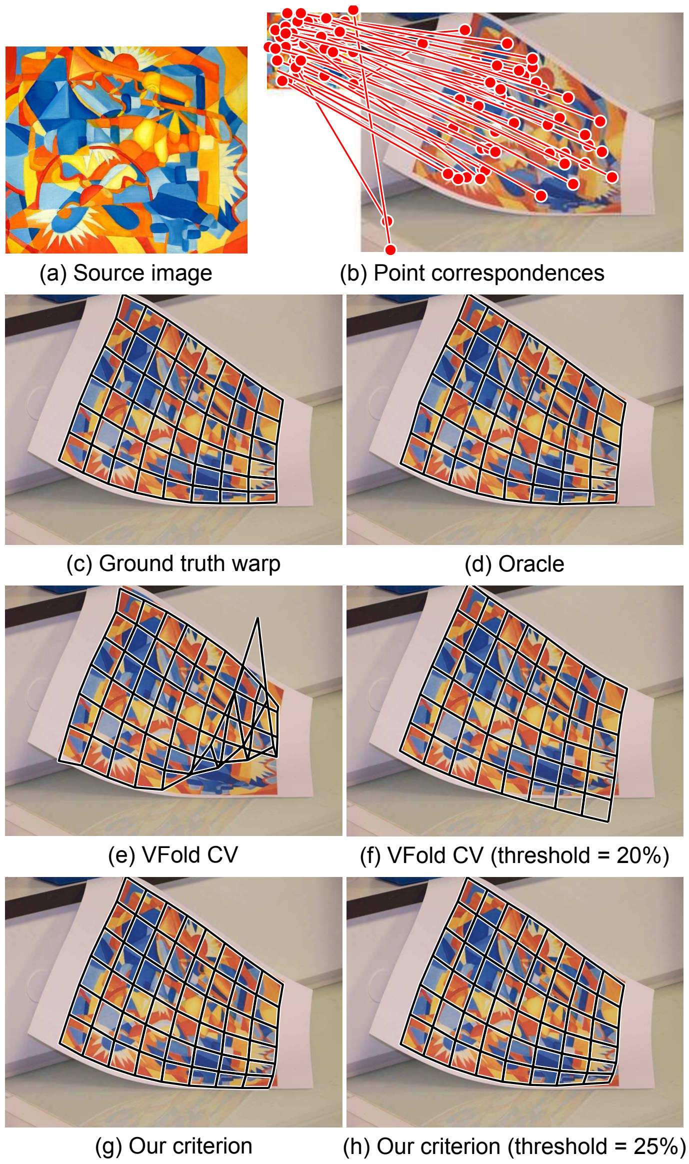

(a) Source image
(b) Point correspondences
(c) Ground truth warp
(d) Oracle
(e) VFold CV
(f) VFold CV (threshold = 20%)
(g) Our criterion
(h) Our criterion (threshold = 25%)

Figure 5.23: Image registered with 3 hyperparameters (γ, λ, and l) automatically determined with several criteria. The point correspondences were obtained with SIFT. The thresholds indicated in (f) and (h) are the thresholds of the trimmed means (see section 5.3.2 and section 5.3.3). In this case, the two variants of our criterion are the ones that lead to the best results.

We report in table 5.2 the RGE as defined in section 5.3.4.3 for the warps estimated in the 'cubist image' experiment.

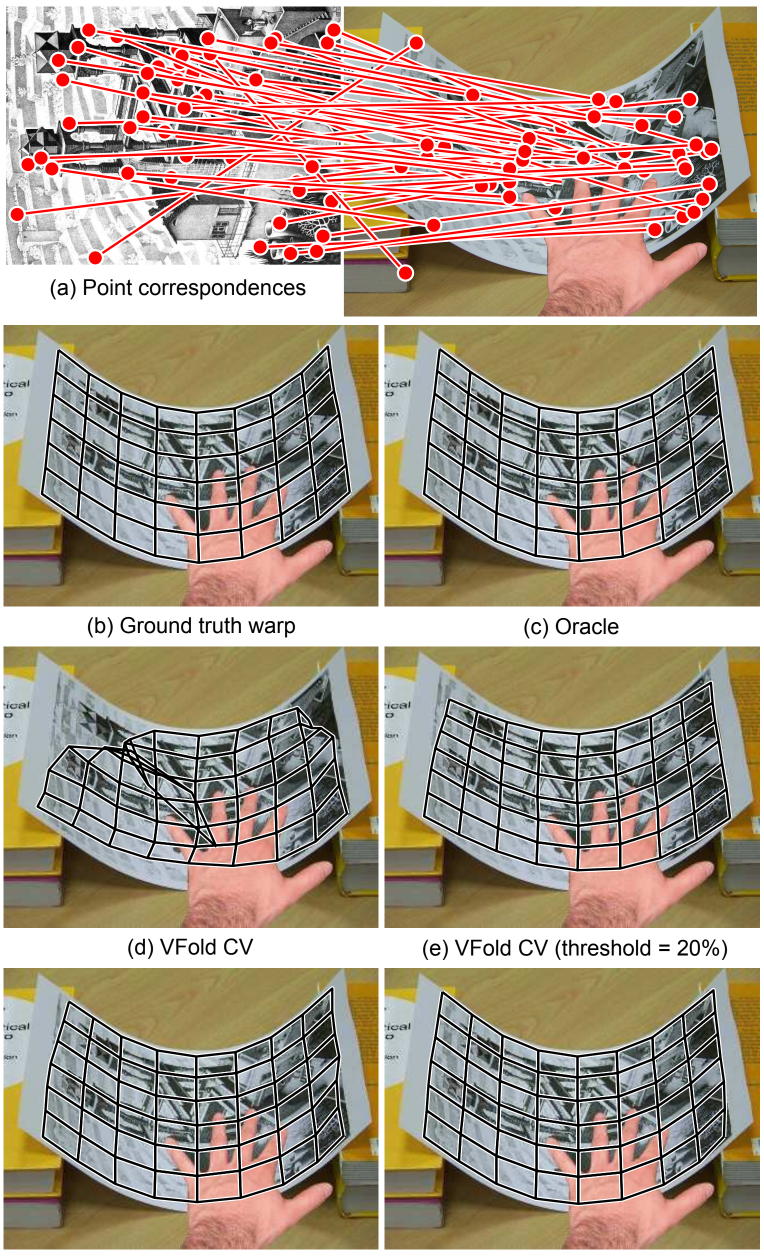
Criterion	**RGE**
V-Fold CV	1.852%
V-Fold CV (robust)	0.675%
Our criterion	0.190%
Our criterion (robust variant)	0.197%

Table 5.2: RGE for the experiment of figure 5.23.

'Waterfall' of Maurits Escher Figure 5.24 shows an experiment similar to the one conducted with the 'cubist image'. Nonetheless, there are some important differences. This time, the keypoints were extracted using the SURF detector of (Bay et al., 2008) and approximately 12% of the 621 point correspondences were erroneous. An artificial occlusion was added to the target image; we used an artificial occlusion in order to still be able to determine the ground truth warp (which is done before the insertion of the occlusion). Besides, the M-estimator scale parameter and the smoothing parameter were the only hyperparameters under study (the number of control points of the warp was set to the one of the ground truth warp). As in the 'cubist image' case, the hyperparameters chosen with our photometric criterion are better than the ones estimated with the criterion relying on the V-Fold Cross-Validation. In both cases, the robust versions of the criteria perform better than the non-robust ones. Note that the occlusion added to the target image influences the non-robust V-fold CV criterion since it introduces supplementary false point correspondences.

5.3.5 Conclusion

We proposed a new criterion to automatically tune the hyperparameters in image registration problems. We showed that our photometric criterion performs generally better than other approaches with similar goals such as the Cross-Validation criteria. This was made possible by designing a criterion specifically adapted to the image registration problem that combines the advantages of both the feature-based and the direct approaches to image registration. Our criterion was successfully experimented in a particular but challenging setup: deformable B-spline warps, selection of an M-estimator threshold, presence of outliers and occlusions, *etc*. However, the proposed criterion is not limited to this setup: it is generic enough to be applied in other image registration problems with different constraints, different warps, and, thus, different hyperparameters.



(a) Point correspondences

(b) Ground truth warp

(c) Oracle

(d) VFold CV

(e) VFold CV (threshold = 20%)

(f) Our criterion

(g) Our criterion (threshold = 25%)

Figure 5.24: Image registered with 2 hyperparameters (γ, λ) automatically determined with several criteria. The point correspondences were obtained with SURF. The thresholds indicated in (f) and (h) are the thresholds of the trimmed means (see section 5.3.2 and section 5.3.3). Globally, the robust variants of the VFold CV criterion and of our criterion lead to acceptable results. The non-robust VFold CV criterion is greatly influenced by the presence of outliers in the point correspondences. The non-robust variant of our criterion is slightly more influenced by the occlusion than the robust variant.

Chapter 6

NURBS Warps

Standard Free-Form Deformations (FFD) built upon tensor-product B-Splines have been proved useful to model the warp between two views of a deformable surface. In this chapter, we show that the standard FFD is the affine projection of a threedimensional tensor-product B-Spline surface. We construct a new tensor-product warp relying on Non-Uniform Cubic B-Spline: the NURBS-Warp. We show that this new warp is an extension of the standard FFD that describes the perspective camera model. The parameters of this new deformation model may be estimated using one of the techniques proposed in, for instance, section 5. Our NURBS-Warp is compared to the standard FFD warp for both synthetic and real images. These experiments show that our NURBS-Warp gives better results than the other warps, especially when the perspective effects are important.

The work presented in this chapter has been first published in (Brunet et al., 2009b).

6.1 Introduction

In this chapter, we bring several contributions. We first demonstrate in section 6.2 that the warps based on tensor-product B-splines (hereinafter abbreviated *BS-Warp*) corresponds to affine imaging condition, in the sense that it models the affine projection of some 3D surface. We then propose our most important contribution in section 6.3: a novel parametric warp we call *NURBS-Warp*, that extends the classical BS-Warp to perspective projection. This warp has a simple analytical form: it is obtained as the two-way tensor-product of bivalued Non-Uniform Rational B-Splines (NURBS). Finally, we give in section 6.4 algorithms for the feature-based estimation of our NURBS-Warp. More precisely, we consider that a set $\{\mathbf{q}_k \leftrightarrow \mathbf{q}'_k\}_{k=1,\dots,r}$ of point correspondences between the two images is known, and show how the parameters that minimize the classical *transfer error* can be found, by solving:

$$\min_{\mathbf{x}} \sum_{k=1}^{r} d^2\left(\mathbf{q}'_k, \mathcal{W}(\mathbf{q}_k; \mathbf{x})\right), \tag{6.1}$$

where $\mathcal{W}$ represents the warp and $d^2(\mathbf{a}, \mathbf{b})$ is the squared euclidean distance between the points $\mathbf{a}$ and $\mathbf{b}$. We finally report experimental results in section 6.5 and conclude this section.

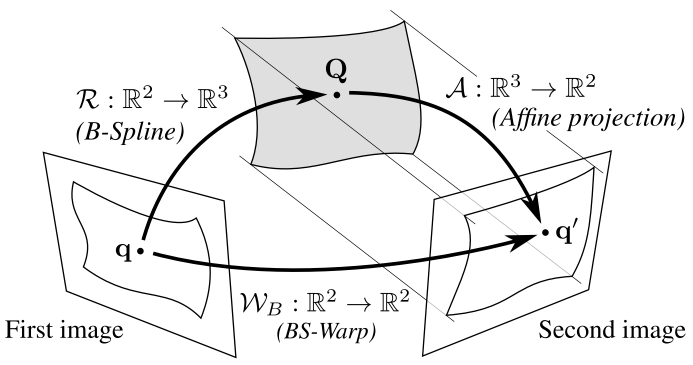
6.2 Affine Interpretation of the BS-Warps

Notation. In this chapter, we denote $\mathcal{W}_B$ the BS-Warp. As a reminder of section 2.3.2, a BS-Warp is defined as:

$$\mathcal{W}_B(\mathbf{q}; \mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{p}_{ij} N_i(x) N_j(y), \tag{6.2}$$

where the functions $N_i : \mathbb{R} \to \mathbb{R}$ are the B-Spline basis functions and the $\mathbf{p}_{ij} = (p_{ij}^x, p_{ij}^y)^\mathsf{T}$ are the control points (grouped in the parameter vector $\mathbf{x} \in \mathbb{R}^{2mn}$). The scalars m and n are the number of control points along the x and y directions respectively. Note that we consider as coincident the knot sequences used to define the B-Spline basis functions.

Affine interpretation. If we consider that the observed surface is modelled by a threedimensional tensor-product B-Spline, the BS-Warp corresponds to the transformation between the two images under affine imaging conditions (see figure 6.1 for an illustration).



$\mathbf{Q}$
$\mathcal{R} : \mathbb{R}^2 \to \mathbb{R}^3$
(B-Spline)
$\mathcal{A} : \mathbb{R}^3 \to \mathbb{R}^2$
(Affine projection)
$\mathbf{q}$
$\mathbf{q}'$
$\mathcal{W}_B : \mathbb{R}^2 \to \mathbb{R}^2$
(BS-Warp)
First image
Second image

Figure 6.1: A BS-Warp can be seen as the result of a threedimensional B-Spline surface projected under affine conditions.

Let $\mathcal{R} : \mathbb{R}^2 \to \mathbb{R}^3$ be the 2D-3D map between the first image and the threedimensional surface:

$$\mathbf{Q} = \mathcal{R}(\mathbf{q}; \mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \bar{\mathbf{p}}_{i,j} N_i(x) N_j(y), \tag{6.3}$$

with $\bar{\mathbf{p}}_{i,j} = (\bar{p}^x_{i,j}\, \bar{p}^y_{i,j}\, \bar{p}^z_{i,j})^\mathsf{T} \in \mathbb{R}^3$ the 3D control points of the surface. Let $\mathcal{A} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ be the affine projection of the surface into the second image:

$$\mathcal{A}(\mathbf{Q}) = \mathsf{A}\mathbf{Q}, \tag{6.4}$$

with A the matrix which models the affine projection, assuming that the 3D surface is expressed within the coordinate frame of the second camera:

$$\mathsf{A} = \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \end{pmatrix}. \tag{6.5}$$

Given these notations, the warped point $\mathbf{q}'$ can be written $\mathcal{A}(\mathcal{R}(\mathbf{q}))$ which, after expansion, gives:

$$\mathbf{q}' = \sum_{i=1}^{m} \sum_{j=1}^{n} \begin{pmatrix} \bar{p}^x_{i,j} \\ \bar{p}^y_{i,j} \end{pmatrix} N_i(x) N_j(y). \tag{6.6}$$

Equation (6.6) matches the definition of equation (6.2) of a BS-Warp.

BS-Warps are not suited for perspective imaging conditions. As we just demonstrated, BS-Warps are obtained under affine imaging conditions. However, this does not prove that they are not suited for perspective imaging conditions. In this paragraph, we experimentally illustrate that BS-Warps are indeed not suited for perspective imaging conditions. This comes from the fact that the division appearing in a perspective projection is not present in the BS-Warp model.

The bad behavior of the BS-Warp in the presence of perspective effects is illustrated in figure 6.2. In this experiment, we simulate a set of point correspondences by transforming a regular grid with an homography parametrized by a scalar a that controls the amount of perspective effect. The 3×3 matrix of this homography, H_a, is given by:

$$\mathsf{H}_a \propto \frac{1}{a} \begin{pmatrix} (a+1)^2/4 & 0 & -(a^2-1)/4 \\ 0 & a(a+1)/2 & 0 \\ -(a^2-1)/4 & 0 & (a+1)^2/4 \end{pmatrix}, \tag{6.7}$$

where $\propto$ indicates equality up to scale. The larger $|a-1|$, the more important the perspective effect. Figure 6.2 c clearly shows that the transformation can be correctly modelled by a BS-Warp only when $a = 1$, *i.e.* when the perspective effect is barely existant. Figure 6.2 d shows that the number of control points of a BS-Warp must be significantly large in order to correctly model a perspective effect.

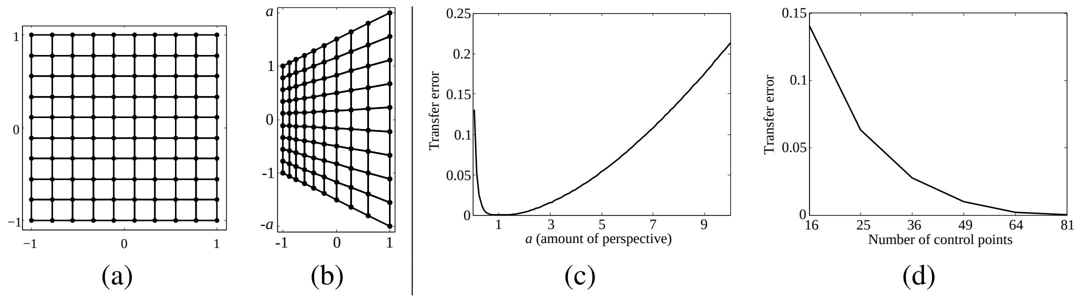


Figure 6.2: Bad behavior of the BS-Warp in the presence of perspective effects. (a) Data points on a regular grid (first image). (b) Transformed points simulating a perspective effect with an homography (second image). (c) Influence of the perspective effect on a BS-Warp with 16 control points. (d) The perspective effect ($a = \frac{5}{2}$) can be modeled with a BS-Warp but it requires a large amount of control points.

6.3 NURBS-Warps

This section introduces a new warp, the NURBS-Warp, which is built upon tensor-product Non-Uniform Cubic B-Splines. We show that the NURBS-Warp naturally appears when replacing the affine projection by a perspective one in the image formation model of the previous section. As our experimental results will show, the NURBS-Warp performs better in the presence of perspective effects. All the necessary details about the NURBS model have been presented in section 2.3.3.

Perspective interpretation. Following the same reasoning than for the BS-Warp in the previous section, we show that the NURBS-Warp corresponds to perspective imaging conditions. This is illustrated in figure 6.3.

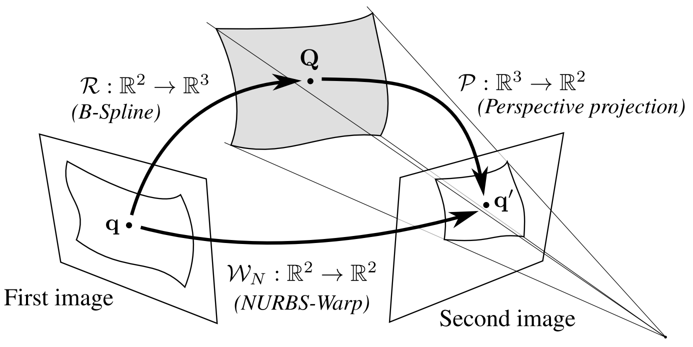

$\mathbf{Q}$
$\mathcal{R} : \mathbb{R}^2 \to \mathbb{R}^3$
(B-Spline)
$\mathcal{P} : \mathbb{R}^3 \to \mathbb{R}^2$
(Perspective projection)
$\mathbf{q}$
$\mathbf{q}'$
$\mathcal{W}_N : \mathbb{R}^2 \to \mathbb{R}^2$
(NURBS-Warp)
First image
Second image

Figure 6.3: A NURBS-Warp can be seen as the result of a threedimensional B-Spline surface projected under perspective conditions.

Let $\mathcal{P}$ be the perspective projection:

$$\mathcal{P}(\mathbf{Q}) = \psi(\mathsf{K}\mathbf{Q}), \tag{6.8}$$

with K the matrix of intrinsic parameters for the second camera. ψ is the homogeneous to affine coordinates function, *i.e.* $\psi(\check{\mathbf{q}}) = \mathbf{q}$ where $\check{\mathbf{q}}$ are the homogeneous coordinates of $\mathbf{q}$ ($\check{\mathbf{q}}^\mathsf{T} = (\mathbf{q}^\mathsf{T}\ 1)$). We assume that the image coordinates are chosen such that the origin coincides with the principal point:

$$\mathsf{K} = \begin{pmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.9}$$

Replacing $\mathbf{Q}$ by its expression of equation (6.3) in equation (6.8) leads to:

$$\mathcal{P}(\mathbf{Q}) = \mathcal{P}(\mathcal{R}(\mathbf{q})) = \frac{1}{\sum_{i=1}^m \sum_{j=1}^n \bar{p}^z_{i,j} N_i(x) N_j(y)} \begin{pmatrix} f_x \sum_{i=1}^m \sum_{j=1}^n \bar{p}^x_{i,j} N_i(x) N_j(y) \\ f_y \sum_{i=1}^m \sum_{j=1}^n \bar{p}^y_{i,j} N_i(x) N_j(y) \end{pmatrix}. \tag{6.10}$$

Defining $w_{i,j} = \bar{p}^z_{i,j}$, $p^x_{i,j} = f_x \frac{\bar{p}^x_{i,j}}{w_{i,j}}$ and $p^y_{i,j} = f_y \frac{\bar{p}^y_{i,j}}{w_{i,j}}$, equation (6.10) is the very definition of a tensor-product NURBS with control points $\mathbf{p}^\mathsf{T}_{i,j} = (p^x_{i,j}, p^y_{i,j})$ and weights $w_{i,j}$ (see section 2.3.3). We denote $\mathcal{W}_N$ this new warp and call it a *NURBS-Warp*:

$$\mathcal{W}_N(\mathbf{q}; \mathbf{x}) = \frac{\sum_{i=1}^m \sum_{j=1}^n \mathbf{p}_{i,j} w_{i,j} N_i(x) N_j(y)}{\sum_{i=1}^m \sum_{j=1}^n w_{i,j} N_i(x) N_j(y)}. \tag{6.11}$$

Here, the warp parameters, *i.e.* the control points and the weights, are grouped into a vector $\mathbf{x} \in \mathbb{R}^{3mn}$.

Using the NURBS-Warp in the setup used for the experiment of figure 6.2 leads to a transfer error consistently smaller than 10^{-5} pixels.

Homogeneous NURBS-Warp. The NURBS-Warp defined by equation (6.11) can be expressed with homogeneous coordinates. We note $\check{\mathcal{W}}_N$ the NURBS-Warp in homogeneous coordinates:

$$\check{\mathcal{W}}_N(\mathbf{q};\mathbf{x}) \propto \begin{pmatrix} \frac{\sum_{i=1}^m \sum_{j=1}^n p_{i,j}^x w_{i,j} N_i(x) N_j(y)}{\sum_{i=1}^m \sum_{j=1}^n w_{i,j} N_i(x) N_j(y)} \\ \frac{\sum_{i=1}^m \sum_{j=1}^n p_{i,j}^y w_{i,j} N_i(x) N_j(y)}{\sum_{i=1}^m \sum_{j=1}^n w_{i,j} N_i(x) N_j(y)} \\ 1 \end{pmatrix} \propto \sum_{i=1}^m \sum_{j=1}^n \begin{pmatrix} p_{i,j}^x w_{i,j} \\ p_{i,j}^y w_{i,j} \\ w_{i,j} \end{pmatrix} N_i(x) N_j(y). \tag{6.12}$$

We observe that in the homogeneous version of equation (6.12), our NURBS-Warp does a linear combination of control points in homogeneous coordinates, as opposed to the classical BS-Warp of equation (6.2) that does a linear combination of control points in affine coordinates. This is what makes our NURBS-Warp able to model perspective projection, thanks to the division 'hidden' in the homogeneous coordinates.

6.4 Parameter Estimation

In this section, we show how the BS-Warp and the NURBS-Warp parameters can be estimated. Here, we consider the feature-based approach for estimating the warp parameters. It has the advantage of being one of the most simple approach to estimate the parameters. We can thus concentrate on the main topic of this chapter, *i.e.* the representational power of the warps. As it was explained in section 5.1, this feature-based approach to image registration amounts to solve the following minimization problem:

$$\min_{\mathbf{x}} \sum_{k=1}^{r} d^2\left(\mathbf{q}'_k, \mathcal{W}(\mathbf{q}_k;\mathbf{x})\right). \tag{6.13}$$

The main difficulty here is that problem (6.13) is not linear when using the NURBS-warps.

6.4.1 The BS-Warp

The dependency of the BS-Warp to its parameters is linear. As a consequence, the optimization problem (6.13) for a BS-Warp simply reduces to an ordinary linear least-squares minimization problem. The details for solving such a problem have already been detailed in, for instance, section 5.3.

6.4.2 The NURBS-Warp

Contrarily to the BS-Warp, the dependency of the NURBS-Warp to its parameters is not linear. Problem (6.13) thus leads one to a non-linear least-squares optimization problem. Note that this remark would still be true for other estimation techniques such as the direct approach to image registration. It is a well-known fact that this kind of problem can be efficiently solved using an iterative algorithm such as Levenberg-Marquardt. Such an algorithm needs an initial solution (see section 2.2.2.7). We propose three approaches to compute an initial set of parameters (which are further detailed in the sequel of this section):

- **First approach:** Act as if the images were taken under affine imaging conditions.
- **Second approach:** Act as if the warp relating the two images was an homography.
- **Third approach:** Use an algebraic approximation to the transfer error function.

Since all of these three approaches are relatively cheap to compute, it is possible to test the three of them and choose as initial parameters those that give the smallest transfer error.

First approach (affine initialization). An initial solution for the NURBS-Warp estimation problem can be computed by setting all the weights to 1. By doing so, the equation defining a NURBS-warp reduces to the expression of a simple BS-Warp. An initial set of parameters can thus be computed using ordinary least-squares minimization. Since the BS-Warp corresponds to affine imaging conditions, this approach is expected to give good results when the effects of perspective are limited.

Second approach (homographic warp). Even if the homographic warp is not suited to deformable environments, it can be a good approximation (particularly if the surface bending is not important). We denote $\mathcal{W}_H$ the homographic warp. It is defined by:

$$\mathcal{W}_H(\mathbf{q}; \mathbf{x}) = \frac{1}{p_7 x + p_8 y + 1} \begin{pmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{pmatrix}, \tag{6.14}$$

with $\mathbf{x}^\mathsf{T} = (p_1 \ \dots \ p_8)$. Minimizing the transfer error of equation (6.13) with the homographic warp $\mathcal{W}_H$ can be achieved iteratively using, for instance, the Levenberg-Marquardt algorithm (and by taking the identity warp or the algebraic solution as an initialization). A complete review of homographic warp estimation can be found in (Hartley and Zisserman, 2003a).

Third approach (algebraic approximation). The third and last approach we propose to initialize the optimization process consists in minimizing an algebraic approximation to the transfer error:

$$\min_{\mathbf{x}} \sum_{k=1}^{r} d_a^2 \left(\mathbf{q}'_k, \check{\mathcal{W}}_N(\mathbf{q}_k; \mathbf{x})\right), \tag{6.15}$$

with $d_a^2(\mathbf{q}', \check{\mathbf{q}}) = \|\mathsf{S}(\tilde{\mathbf{q}}' \times \check{\mathbf{q}})\|^2$ an algebraic distance between the points[1] $\mathbf{q}$ and $\mathbf{q}'$. The operator S removes the last element of a 3-vector. $\tilde{\mathbf{q}}'$ are the scaled homogeneous coordinates of $\mathbf{q}'$ (*i.e.* $\tilde{\mathbf{q}}'^\mathsf{T} = (\mathbf{q}'^\mathsf{T} \ 1)$). If we make the following variable change in equation (6.12) of the homogeneous NURBS-Warp: $a_{i,j} = p^x_{i,j} w_{i,j}$ and $b_{i,j} = p^y_{i,j} w_{i,j}$ then it is straightforward to see that the algebraic distance is the squared euclidean norm of an expression linear with respect to the parameters $a_{i,j}$, $b_{i,j}$ and $w_{i,j}$. Replacing the algebraic distance by its expression in the initial optimization problem of equation (6.15) leads to homogeneous linear least-squares. This problem can be solved using the singular value decomposition presented in section 2.2.2.10.

6.5 Experiments

6.5.1 Simulated Data

Simulation setup. We generate two images by simulating two cameras looking at a surface with different deformations between the views. The surface is generated as a linear combination of two objects: a simple plane and a more deformed surface (see figure 6.4). A single parameter, α, controls the amount of surface bending. Points on the surface are projected on the images and corrupted with an additive gaussian noise. The perspective effect is controlled by varying the distance between the scene and the camera (the focal length, f, is adjusted so that the apparent size of the imaged surface remains approximately the same). The influence of three quantities are tested independently:

- the amount of noise: controlled by the standard deviation σ in pixels;

[1]This algebraic distance is valid only if the point coordinates are normalized according to (Hartley and Zisserman, 2003a). We did not introduced it in our equations for the sake of clarity.

- the amount of bending: controlled by the previously described parameter α;
- the amount of perspective: controlled by the scene to camera distance d, in pixels.

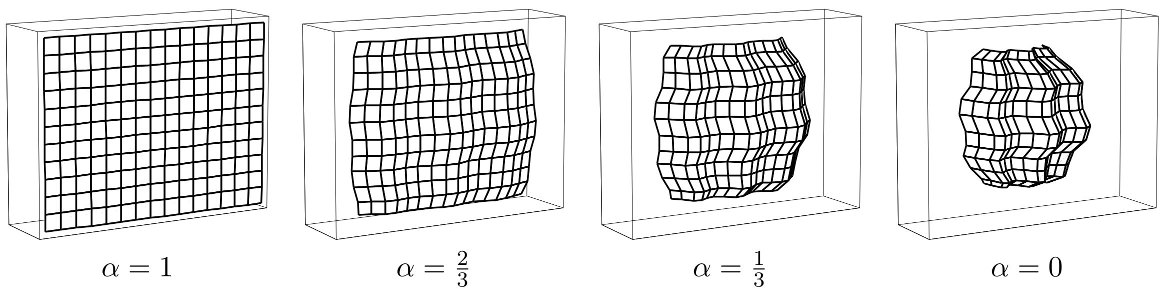


Figure 6.4: The simulated threedimensional surfaces used in our experiments are obtained as linear combinations of a plane ($\alpha = 1$) and of a more distorted surface ($\alpha = 0$).

The default simulation parameters are $r = 192$ point correspondences, $\sigma = 1$ pixel, $\alpha = \frac{1}{2}$, $d = 800$ pixels, $f = 400$ pixels and n control points. These parameters yield mild perspective effects. For each set of parameters, the generated surface is also rotated around its center. The reported results are the average of the mean transfer error over 100 surfaces which have different rotations and corruptions of the projected points. For each varied parameter, three plots report the results for different numbers of control points ($n = 16$, $n = 25$ and $n = 36$). Four curves are reported on each plot. They correspond to the homographic warp, the initial estimate of the NURBS-Warp using the algebraic distance, the optimal BS-Warp and the final NURBS-Warp.

Two general observations can be made about those results. First, the experiments show that the NURBS-Warp always outperforms the three other approaches. This comes from the fact that it is designed, by construction, to model more complex deformations between images[2]. Second, the following results show that the larger the number of control points, the lower the transfer error.

Influence of noise. Figure 6.5 shows the influence of the amount of noise on the estimated warps. We can see that the influence of this factor is relatively limited and, more importantly, that this influence is similar for all the four warps.

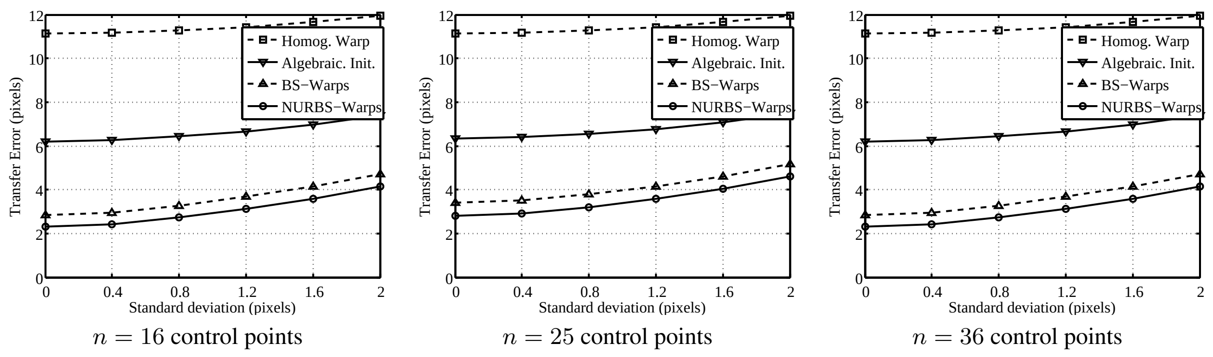
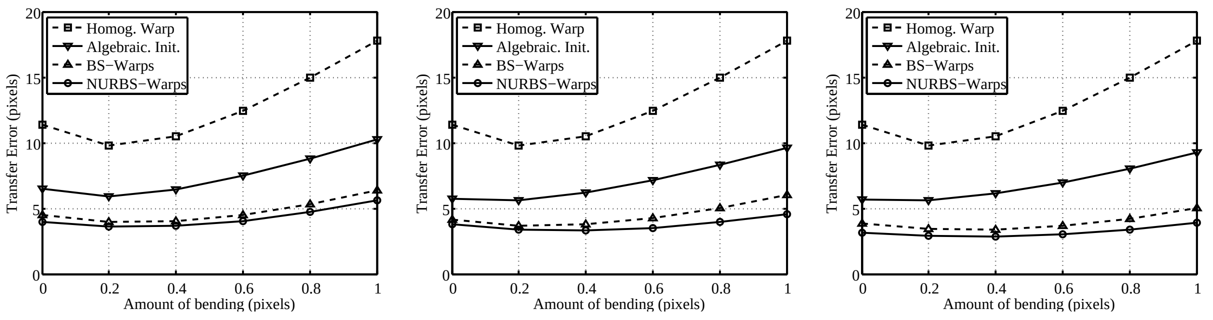


Figure 6.5: The influence of the amount of noise.

Influence of bending. The influence of the amount of bending is studied by varying the parameter α. The results are reported in figure 6.6. This experiment shows, as expected, that the homographic warp is the one which is the most influenced by the deformations of the observed surface. It comes from the fact that the homographic warp inherently does not model deformable environments.

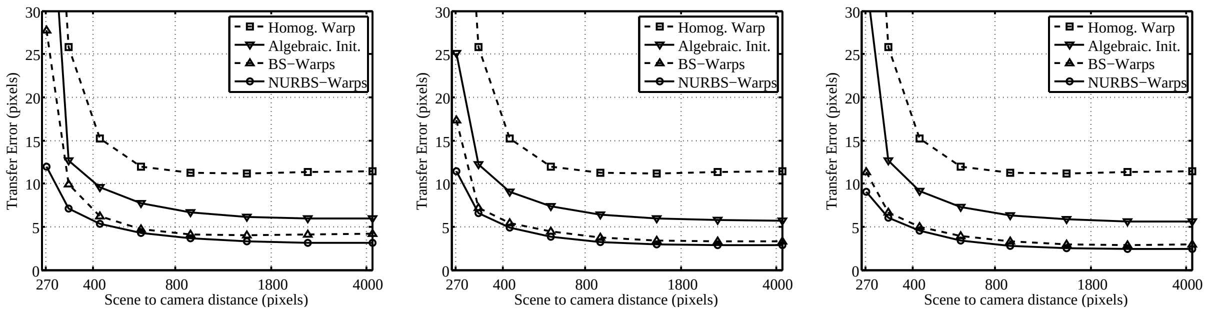
[2]Besides, it seems natural that the NURBS-Warp estimate is better than the other approaches since its parameter estimation starts from an initial solution which corresponds to the parameters of the best estimate among the three other approaches.



$n = 16$ control points $n = 25$ control points $n = 36$ control points

Figure 6.6: The influence of the amount of bending.

Influence of perspective (1). The influence of the perspective effect is studied by varying the scene to camera distance: the larger the distance, the more affine the imaging conditions. The results are reported in figure 6.7. This experiment truly reveals the full power of the proposed NURBS-Warp. Indeed, for the lowest number of control points ($n = 16$), we see that with an important perspective effect ($d = 270$), the NURBS-Warp is more than twice as good as the BS-Warp. Two main reasons explain this result. First, the NURBS-Warp has been designed to model perspective effects. Second, the behavior of the BS-Warp is better for affine imaging conditions. Figure 6.7 also shows that the BS-Warp can correctly model the deformations but it requires a large amount of control points.



$n = 16$ control points $n = 25$ control points $n = 36$ control points

Figure 6.7: The influence of the amount of perspective.

Influence of perspective (2). In the previous experiment the warps had to cope simultaneously with perspective effects and large surface deformations. As a consequence, it is difficult to see the benefit brought by our NURBS-Warp. We thus present the influence of perspective with less complex deformations. We use the same experimental setup except for the generated surface which is now obtained as the linear combination of a plane and a half-cylinder (see figure 6.8 a). Figure 6.8 c shows that the NURBS-Warp brings a real benefit compared to the BS-Warp when the perspective effect are important: for example, we see that compared to the BS-Warp, the NURBS-Warp transfer error is over 3 times lower for a camera to scene distance of 270 pixels.

From the last two experiments, we can say that the proposed NURBS-Warp is well suited for large perspective effects. However, when the surface deformations are significant, both the NURBS-Warp and the BS-Warp require a lot of control points. Since it increases the number of degree of freedom, the BS-Warp can also cope with perspective effects.

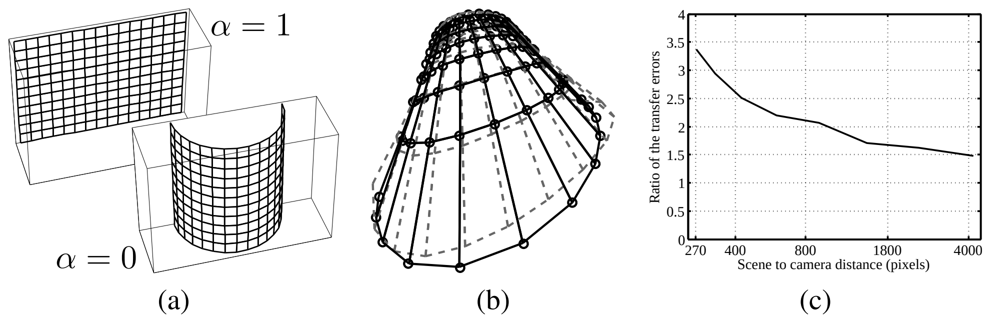


Figure 6.8: (a) Simple deformations are obtained using a linear combination of a plane and a half-cylinder. (b) The NURBS-Warp (black visualization grid) models better the perpective effect than the BS-Warp (dashed grid). The black circles represent the ground truth location of the vertices. (c) The BS-Warp transfer error divided by the one of the NURBS-Warp.

6.5.2 Real Images

Figure 6.9 and figure 6.10 show examples of warps estimated for a rigid and a deformable scene respectively. The warps are estimated from 130 point correspondences that we entered manually (represented by the small circles). We set the number of control points to 16 for both the BS-Warp and the NURBS-Warp. The estimated warps are represented by a visualization grid. This grid is shown in its rest position in the first image. It is then transferred into the second image using the estimated warps. The conclusion of this experiment is that the NURBS-Warp is the one that gives the best results. This is especially true when perspective effects and surface deformations are combined. In particular, we can see in figure 6.10 that the NURBS-Warp is the only warp able to retrieve realistic surface deformations. Note that the transfer errors reported in figure 6.9 and figure 6.10 may seem important. These high values come from the fact that the images have a quite high resolution (approximately 3072×2304 pixels).

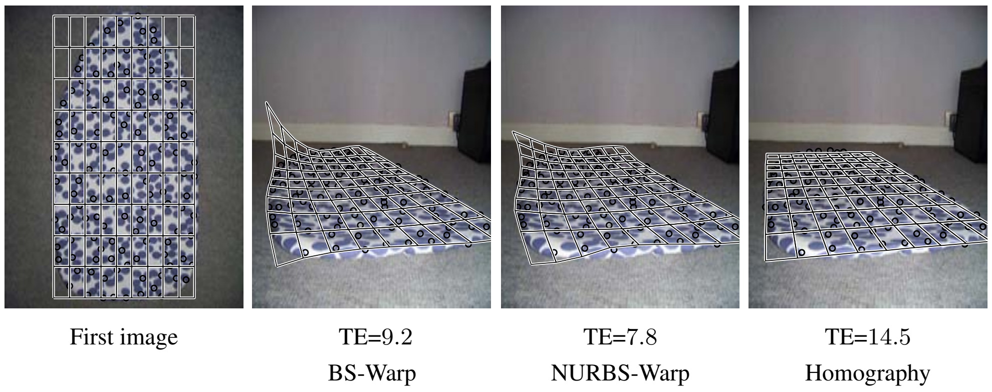


Figure 6.9: Warps estimated for a rigid surface (TE: Transfer Error in pixels).

6.6 Conclusion

We introduced a new parametric warp we called NURBS-Warp. It was derived by analyzing the classical BS-Warp, that we showed is based on the two-way tensor product of bivariate B-Splines. As a first contribution, we showed that the BS-Warp is intrinsically affine: it does not model the effect of perspective projection. Our NURBS-Warp, based on the tensor product of NURBS, is an extension of the BS-Warp. It models perspective projection, and thus copes with more complex deformations with less control points, as our experimental results show. An estimation procedure from point correspondences was given. It allowed us to demonstrate the representational power of our warp compared to the BS-Warp, and against various factors such as noise, surface

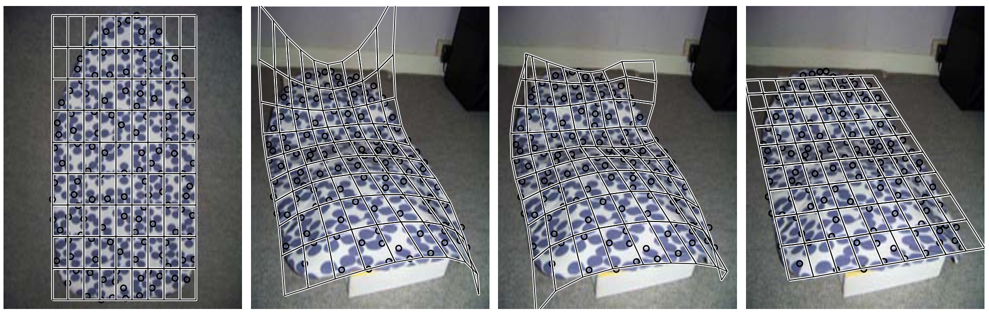


Figure 6.10: Warps estimated for a deformable scene (TE: Transfer Error in pixels).

shape and number of control points. However, it might be interesting for future work to study how classical robust and pixel-based methods can be applied to our NURBS-Warp.

Chapter 7

Monocular Template-based Reconstruction of Smooth and Inextensible Surfaces

In this chapter, we deal with another problem of importance in computer vision: the reconstruction of a deforming surface from a monocular sequence of images. Our contribution is a new algorithm that works under the assumption that the deformable surface is inextensible. This problem may be seen as a natural continuation of the aspects previously treated in this document. Indeed, as we will show, the approach we use for reconstructing deformable surfaces is again formulated as a parameter estimation problem. Besides, given the considered approach, the reconstruction of deformable surfaces requires one to first register the current image with some template image. Our contributions related to this topic from section 5 and section 6 may thus be used here. The work presented in this chapter has been published in (Brunet et al., 2010e).

7.1 Introduction

Monocular surface reconstruction of deformable objects is a challenging problem which has known renewed interest during the past few years. This problem is fundamentally ill-posed because of the depth ambiguities; there are virtually an infinite number of 3D surfaces that have exactly the same projection. It is thus necessary to use additional constraints ensuring the consistency of the reconstructed surface.

In this chapter, we present two algorithms for monocular reconstruction of deformable and inextensible surfaces under some general assumptions. First, we consider the *template-based* case. Reconstruction is achieved from point correspondences between an input image and a template image showing a flat reference shape from a fronto-parallel point of view. Second, we suppose the intrinsic parameters of the camera to be known. Third, we assume that the camera is a perspective camera. These are common assumptions (Perriollat et al., 2010; Salzmann et al., 2009; Shen et al., 2010).

Over the years, different types of constraints have been proposed to disambiguate the problem of monocular reconstruction of deformable surfaces. They can be divided into two main categories: the *statistical* and the *physical* constraints. For instance, the methods relying on the low-rank factorization paradigm (Bartoli et al., 2008; Brand, 2005; Bregler et al., 2000; Del Bue, 2008; Olsen and Bartoli, 2008; Torresani et al., 2008; Xiao et al., 2006) can be classified as statistical approaches. Learning approaches such as (Gay-Bellile et al., 2006; Salzmann et al., 2007, 2008b, 2009) also belong to the statistical approaches. Work such as (Salzmann et al., 2009), where the reconstructed surface is represented as a linear combination of inextensible deformation modes, is also a statistical approach. Physical constraints include spatial and temporal priors on the surface to reconstruct (Gumerov et al., 2004; Prasad et al., 2006). Statistical and physical priors can be combined (Bartoli et al., 2008; Del Bue, 2008). A physical prior of particular interest is the hypothesis of having an inextensible surface (Perriollat et al., 2010; Salzmann et al., 2008a, 2009; Shen et al., 2010). In this chapter, we consider this type of surface. This hypothesis means that the geodesics on the surface may not change their length across time. However, computing geodesics is generally hard to achieve and it is even more difficult to incorporate such constraints in a reconstruction algorithm. There exist several approaches to approximate this type of constraint. For instance, if the points are sufficiently close together, the geodesic between two 3D points on the surface can be approximated by the Euclidean distance (Shen et al., 2009). An efficient approximation consists in saying that the geodesic distance between two points is an upper bound to the Euclidean distance (Perriollat et al., 2010; Salzmann et al., 2008a).

Algorithms for monocular reconstruction of deformable surfaces can also be categorized according to the type of surface model (or representation) they use. The *point-wise* methods utilize a sparse representation of the 3D surface, *i.e.* they only retrieve the 3D positions of the data points (Perriollat et al., 2010). Other methods use more complex surface models such as triangular meshes (Salzmann et al., 2008a, 2009) or smooth surfaces such as Thin-Plate Splines (Bartoli, 2008a; Perriollat et al., 2010). In this latter case, the 3D surface is represented as a parametric 2D-3D map between the template image space and the 3D space. Smooth surfaces are generally obtained by fitting a parametric model to a sparse set of reconstructed 3D points: the smooth surface is not actually used in the 3D reconstruction process. In this chapter, we propose an algorithm that directly estimate a smooth 3D surface based on Free-Form Deformations (Rueckert et al., 1999). Having an inextensible surface means that the surface must be everywhere a local isometry. This induces conditions on the Jacobian matrix of the 2D-3D map. We show that these conditions can be integrated in a non-linear least-squares minimization problem along with some other constraints that force the consistency between the reconstructed surface and the point correspondences. Such a problem can be solved using an iterative optimization procedure such as Levenberg-Marquardt that we initialize using a point-wise reconstruction algorithm. Our approach is highly effective in the sense that it outperforms previous approaches in term of accuracy of the reconstructed surface and in terms of inextensibility.

Another important aspect in monocular reconstruction of deformable surfaces is the way noise is handled. It can be accounted for in the template image (Perriollat et al., 2010) or in the input image (Salzmann et al., 2009). There exist different approaches for handling the noise. For instance, one can minimize a reprojection error, *i.e.* the distance between the data points of the input image and the projection of the reconstructed 3D points. It is also possible to hypothesize maximal inaccuracies in the data points. We propose a point-wise approach that accounts for noise in both the template and the input images. This approach is formulated as a second-order cone program (SOCP) (Boyd and Vandenberghe, 2004).

Notation	**Description**
P	Matrix of the intrinsic parameters of the camera ($\mathsf{P} \in \mathbb{R}^{3\times 3}$)
	(The camera is assumed to be at the coordinate origin, so the matrix P
	may be assumed to be square and invertible.)
$\mathbf{p}_k^\mathsf{T}$	kth row of the matrix P
n_c	Number of point correspondences
$\mathbf{q}_i$	ith point in the template image
$\mathbf{q}'_i$	ith point in the input image; $i \in \{1, \dots, n_c\}$
$\bar{\mathbf{q}}_i$	Point $\mathbf{q}_i$ in homogeneous coordinates
$\mathbf{u}_i$	Sightline corresponding to the point $\mathbf{q}'_i$ ($\mathbf{u}_i = (\mathsf{P}^{-1}\bar{\mathbf{q}}'_i)/\|\mathsf{P}^{-1}\bar{\mathbf{q}}'_i\|$)
μ_i	Depth of the point $\mathbf{Q}_i$
$\mathbf{Q}_i$	Reconstructed 3D point i
d_{ij}	Euclidean distance between points i and j ($d_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|$)
$\hat{x}$	True value of x (for $x = \mathbf{q}'_i, \mathbf{q}_i, \mathbf{Q}_i, \mathbf{u}_i, \mu_i, d_{ij}$)

Table 7.1: Summary of the notation used in this chapter.

7.2 Related Work on Inextensible Surface Reconstruction

A popular assumption made in deformable surface reconstruction is to consider that the surface to reconstruct is inextensible (Perriollat et al., 2010; Salzmann et al., 2008a, 2009; Shen et al., 2010). This assumption is reasonable for many types of material such as paper and some types of fabrics. Having an inextensible surface means that the surface is an isometric deformation of the reference shape. Another way of putting it is to say that the length of the geodesics between pairs of points remains unchanged when the surface deforms. An exact transcription of this principle is difficult to integrate in a reconstruction algorithm. Indeed, while it is trivial to compute the geodesic in a flat reference shape, it is quite difficult to do it for a bent surface (especially when the surface is represented as a sparse set of points or a triangular mesh). Many approximations have thus been proposed.

The first type of approximation consists in saying that if the surface does not deform too much then the Euclidean distance is a good approximation to the geodesic distance. Such an approach has been used for instance in (Salzmann et al., 2007, 2008a; Shen et al., 2010; Zhu et al., 2009). Note that these types of constraints are usually set in a soft way. For a given set of point pairs on the surface, the Euclidean distance should not diverge too much from the geodesic distances. This approximation is better when there are a large number of points. Depending on the surface model it is not always possible to vary the number of points.

Although the Euclidean approximation can work well in some cases, this approximation gives poor results when creases appear in the 3D surface. In this case, the Euclidean distance between two points on the surface can shrink, as illustrated in figure 7.1. The 'upper bound approach' is a now classical approach (Perriollat

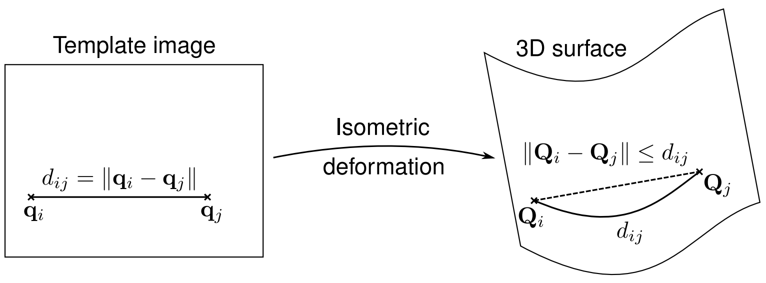


Figure 7.1: Inextensible object deformation. The Euclidean distance between two points is necessarily less than or equal to the length of the geodesic that links those two points (this length is easily computable if we have a template image representing the flat reference surface from a fronto-parallel point of view).

et al., 2010; Salzmann et al., 2009) which consists in noticing that even if the Euclidean distance between two points can shrink it can never be greater than the length of the corresponding geodesic. In other words, the *inextensibility constraint* $\|\mathbf{Q}_i - \mathbf{Q}_j\| \leq d_{ij}$ must be satisfied for any pair of points $(\mathbf{Q}_i, \mathbf{Q}_j)$ lying on the surface. The second principle of such algorithms is to say that a 3D point $\mathbf{Q}_i$ must lie on the sightline $\mathbf{u}_i$, *i.e.* $\mathbf{Q}_i = \mu_i \mathbf{u}_i$. These two constraints are not sufficient to reconstruct the surface. Indeed, nothing prevents the reconstructed surface from shrinking towards the optical centre of the camera. This problem is 'solved' using a heuristic that has been proven to be very effective in practice. It consists in considering a perspective camera and in maximizing the depth of the reconstructed 3D points.

These ideas have been implemented in different manners. For instance, (Perriollat et al., 2010) proposes a dedicated algorithm that enforces the inextensibility constraints. This algorithm accounts for noise only in the template image (by simply increasing a little bit the geodesic distances in the template, *i.e.* by replacing d_{ij} with $d_{ij} + \varepsilon_{\mathcal{T}}$ where $\varepsilon_{\mathcal{T}}$ is the maximal inaccuracy of the points in the template image). Another sort of implementation is given by (Salzmann et al., 2008a, 2009). In these papers, a convex cost function combining the depth of the reconstructed points and the negative of the reprojection error is maximized while enforcing the inequality constraints arising from the surface inextensibility. The resulting formulation can be easily turned into an SOCP problem. A similar approach is explored in (Shen et al., 2010). These last two methods account for noise in the input image. The approach of (Perriollat et al., 2010) is a point-wise method. The approaches of (Salzmann et al., 2008a, 2009; Shen et al., 2010) use a triangular mesh as surface model, and the inextensibility constraints are applied to the vertices of the mesh.

7.3 Convex Formulation of the Upper Bound Approach with Noise in all Images

In this section, we propose a convex formulation of the principles sketched in section 7.2 that, compared to (Perriollat et al., 2010), accounts for noise in both the template and the input images. We can express this in terms of image-plane measurements. As in (Salzmann et al., 2008a, 2009), our approach is formulated as an SOCP problem. However, contrary to (Salzmann et al., 2008a, 2009), our approach is a point-wise method that does not require us to tune the relative influence of minimizing the reprojection error and maximizing the depths.

7.3.1 Noise in the Template Only

Let us first remark that the basic principles explained in section 7.2 can be formulated as SOCP problems. In this first formulation, the noise is only accounted for in the template image. The inextensibility constraint

$\|\mathbf{Q}_i - \mathbf{Q}_j\| \le d_{ij} + \varepsilon_{\mathcal{T}}$ can be written:

$$\|\mu_i \mathbf{u}_i - \mu_j \mathbf{u}_j\| \le d_{ij} + \varepsilon_{\mathcal{T}}. \tag{7.1}$$

Including the maximization of the depths, we obtain this SOCP problem:

$$\begin{array}{lll} \max\limits_{\boldsymbol{\mu}} & \sum\limits_{i=1}^{n_c} \mu_i & \\ \text{subject to} & \|\mu_i \mathbf{u}_i - \mu_j \mathbf{u}_j\| \le d_{ij} + \varepsilon_{\mathcal{T}} & \forall (i,j) \in \mathcal{E} \\ & \mu_i \ge 0 & i \in \{1, \ldots, n_c\} \end{array} \tag{7.2}$$

where $\boldsymbol{\mu}^{\mathsf{T}} = \begin{pmatrix} \mu_1 & \ldots & \mu_{n_c} \end{pmatrix}$, and $\mathcal{E}$ is a set of pairs of points to which the inextensibility constraints are applied.

7.3.2 Noise in Both the Template and the Input Images

Let us now suppose that the inaccuracies are expressed in terms of image-plane measurements. Suppose that points are measured in the image with a maximum error of $\varepsilon_{\mathcal{I}}$, *i.e.*

$$\|\hat{\mathbf{q}}'_i - \mathbf{q}'_i\| \le \varepsilon_{\mathcal{I}}, \qquad \forall i \in \{1, \ldots, n_c\}. \tag{7.3}$$

Since we are searching for the true 3D position of the point $\mathbf{Q}_i$, we say that:

$$\hat{\mathbf{q}}'_i = \frac{1}{\mathbf{p}_3^{\mathsf{T}} \mathbf{Q}_i} \begin{pmatrix} \mathbf{p}_1^{\mathsf{T}} \mathbf{Q}_i \\ \mathbf{p}_2^{\mathsf{T}} \mathbf{Q}_i \end{pmatrix}. \tag{7.4}$$

Equation (7.3) can thus be rewritten:

$$\left\| \frac{1}{\mathbf{p}_3^{\mathsf{T}} \mathbf{Q}_i} \begin{pmatrix} \mathbf{p}_1^{\mathsf{T}} \mathbf{Q}_i \\ \mathbf{p}_2^{\mathsf{T}} \mathbf{Q}_i \end{pmatrix} - \mathbf{q}'_i \right\| \le \varepsilon_{\mathcal{I}}. \tag{7.5}$$

We finally add the inextensibility constraints and the maximization of the depths (which are given by $\mathbf{p}_3^{\mathsf{T}} \mathbf{Q}_i$) and we obtain the following SOCP problem:

$$\begin{array}{lll} \max\limits_{\mathbf{Q}} & \mathbf{p}_3^{\mathsf{T}} \sum\limits_{i=1}^{n} \mathbf{Q}_i & \\ \text{subject to} & \left\| \begin{bmatrix} \mathbf{p}_1^{\mathsf{T}} \\ \mathbf{p}_2^{\mathsf{T}} \end{bmatrix} \mathbf{Q}_i - \mathbf{q}'_i \mathbf{p}_3^{\mathsf{T}} \mathbf{Q}_i \right\| \le \varepsilon_{\mathcal{I}} \, \mathbf{p}_3^{\mathsf{T}} \mathbf{Q}_i & \forall i \in \{1, \ldots, n_c\} \\ & \|\mathbf{Q}_i - \mathbf{Q}_j\| \le d_{ij} & \forall (i,j) \in \mathcal{E} \\ & \mathbf{p}_3^{\mathsf{T}} \mathbf{Q}_i \ge 0 & \forall i \in \{1, \ldots, n_c\} \end{array} \tag{7.6}$$

where $\mathbf{Q}$ is the concatenation of the 3D points $\mathbf{Q}_i$, for $i \in \{1, \ldots, n_c\}$.

7.4 Smooth and Inextensible Surface Reconstruction

Although the strategem of maximizing the sum of depths $\sum_{i=1}^{n_c} \mu_i$ described in the previous section gives reasonable results, it is merely a heuristic, not based on any valid principle related to surface properties. We therefore consider next a new formulation based on the principle of surface inextensibility.

Let the surface be modelled as a function $\mathcal{W} : \mathbb{R}^2 \to \mathbb{R}^3$, mapping the planar template to 3-dimensional space. The inextensibility constraint is equivalent to saying that the map $\mathcal{W}$ must be everywhere a local isometry. This condition may be expressed in terms of its Jacobian. Let $\mathsf{J}(\mathbf{q}) \in \mathbb{R}^{3\times 2}$ be the Jacobian matrix $\partial\mathcal{W}/\partial\mathbf{q}$ evaluated at the point $\mathbf{q}$. The map $\mathcal{W}$ is an isometry at $\mathbf{q}$ if the columns of $\mathsf{J}(\mathbf{q})$ are orthonormal. This local isometry can be enforced for the whole surface with the following least-squares constraint:

$$\iint \left\| \mathsf{J}(\mathbf{q})^\mathsf{T}\mathsf{J}(\mathbf{q}) - \mathsf{I}_2 \right\|^2 \mathrm{d}\mathbf{q} = 0. \tag{7.7}$$

In practice, we consider a discretization of the quantity in equation (7.7), namely

$$\mathcal{E}_i(\mathcal{W}) = \sum_{j=1}^{n_j} \left\| \mathsf{J}(\mathbf{g}_j)^\mathsf{T}\mathsf{J}(\mathbf{g}_j) - \mathsf{I}_2 \right\|^2, \tag{7.8}$$

where $\{\mathbf{g}_j\}_{j=1}^{n_j}$ is a set of 2D points in the template image space taken on a fine and regular grid (for instance, a grid of size 30×30). This term $\mathcal{E}_i(\mathcal{W})$ measures the departure from inextensibility of the surface $\mathcal{W}$.

Our minimization problem is then to minimize this quantity, over all possible surfaces, subject to the projection constraints, namely that point $\mathcal{W}(\mathbf{q}_i)$ projects to (or near to) the image point $\mathbf{q}_i'$, for all i.

7.4.1 Parametric Surface Model

The problem just described involves a minimization over all possible surfaces. Instead of considering this as a variational problem over all possible surfaces, we consider a parametrized family of surfaces. For this purpose, we chose Free-Form Deformations (FFD) (Rueckert et al., 1999) based on uniform cubic B-splines (Dierckx, 1993). All the details on this parametric model have been given in section 2.3.2. Here, we just precise the notation used in this chapter. Let $\mathcal{W}_{\boldsymbol{\ell}} : \mathbb{R}^2 \to \mathbb{R}^3$ be the parametric FFD, parametrized by a family of 3D points $\boldsymbol{\ell}_{jk}$; $j \in \{1, \ldots, n_u\}$, $k \in \{1, \ldots, n_v\}$, which act as 'attractors' for the surface.

For a point $\mathbf{q} = (u, v)$ in the template, the surface point is explicitly given as

$$\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{q}) = \sum_{j=1}^{n_u}\sum_{k=1}^{n_v} \boldsymbol{\ell}_{jk} N_j(u) N_k(v). \tag{7.9}$$

The functions N_j are the B-spline basis functions (Dierckx, 1993) which are polynomials of degree 3. If point $\mathbf{q}_i = (u_i, v_i)$ is fixed and known then the surface point $\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{q}_i)$ is expressed as a linear combination of the points $\boldsymbol{\ell}_{jk}$, and hence can be written in the form $\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{q}_i) = \mathsf{W}_i\boldsymbol{\ell}$, where W_i is a $3 \times n_u n_v$ matrix depending only on the point $\mathbf{q}_i$, and $\boldsymbol{\ell}$ is the vector obtained by concatenating all the points $\boldsymbol{\ell}_{jk}$. Thus, the 3D point is a linear expression in terms of the parameter vector $\boldsymbol{\ell}$. Since the polynomials N_j and N_k depend only on a local set of the attractor points $\boldsymbol{\ell}_{jk}$, the matrix W_i is sparse, which is important for computational efficiency.

7.4.2 Surface Reconstruction as a Least-Squares Problem

By replacing $\mathbf{Q}_i$ by $\mathsf{W}_i\boldsymbol{\ell}$ in equation (7.6) we may arrive at a constraint:

$$\left\| \left(\begin{bmatrix} \mathbf{p}_1^\mathsf{T} \\ \mathbf{p}_2^\mathsf{T} \end{bmatrix} - \mathbf{q}_i'\mathbf{p}_3^\mathsf{T} \right) \mathsf{W}_i\boldsymbol{\ell} \right\| \leq \varepsilon_{\mathcal{I}} \mathbf{p}_3^\mathsf{T}\mathsf{W}_i\boldsymbol{\ell}. \tag{7.10}$$

We may then formulate the optimization problem as minimizing the inextensibility cost $\mathcal{E}_i(\mathcal{W}_{\boldsymbol{\ell}})$ given in equation (7.8) over all choices of parameters $\boldsymbol{\ell}$, subject to constraints equation (7.10). The constraints are SOCP constraints, but the cost function equation (7.8) is of higher degree in the parameters. To avoid the difficulties of constrained non-linear optimization, we choose a different course, by including the reprojection error into the cost function, leading to an unconstrained problem.

To simplify the formulation of the reprojection error, we introduce the depths μ_i as subsidiary variables, for reasons that become evident below. This is not strictly necessary, but reduces the degree of the reprojection-error term. The minimization problem now takes the form:

$$\min_{\boldsymbol{\mu},\boldsymbol{\ell}} \mathcal{E}_d(\boldsymbol{\mu},\boldsymbol{\ell}) + \alpha\mathcal{E}_i(\boldsymbol{\ell}) + \beta\mathcal{E}_s(\boldsymbol{\ell}), \tag{7.11}$$

where $\mathcal{E}_d$, $\mathcal{E}_i$, $\mathcal{E}_s$ are the *data* (reprojection error), *inextensibilty*, and *smoothing* terms respectively. The data term ensures the consistency of the point correspondences with the reconstructed surface. $\mathcal{E}_i$ forces the inextensibility of the surface. $\mathcal{E}_b$ promotes smooth surface in order to cope with, for instance, lack of data. The relative influence of these three terms are controlled with the weights $\alpha \in \mathbb{R}_+$ and $\beta \in \mathbb{R}_+$. Note that the choice of α and β is generally not critical.

The inextensibility term has been described previously. We now describe the two other terms in equation (7.11).

Data term. Replacing $\mathbf{Q}_i$ by $\mathsf{W}_i\boldsymbol{\ell}$ in equation (7.5) gives an expression for the reprojection error associated with some point. However, the resulting expression is non-linear with respect to the parameters $\boldsymbol{\ell}$. We thus prefer a linear data term expressed in terms of '3D errors', which is the reason why we introduced the depths $\boldsymbol{\mu}$ of the data points in the optimization problem. The data term is then defined by:

$$\mathcal{E}_d(\boldsymbol{\mu},\boldsymbol{\ell}) = \sum_{i=1}^{n_c} \left\|\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{q}_i) - \mu_i \mathsf{P}^{-1}\bar{\mathbf{q}}'_i\right\|^2, \tag{7.12}$$

which measures the distance between the point $\mathcal{W}_{\boldsymbol{\ell}}$ on the surface and the point at depth μ_i along the ray defined by $\mathbf{q}'_i$.

Smoothing term. In some cases, the point correspondences and the hypothesis of an inextensible surface are not sufficient. For instance, imagine that there is no point correspondence in a corner of the surface. In this case, there is nothing that indicates how the surface should behave. The corners of the surface can bend freely as long as they do not extend or shrink (like the corners of a piece of paper). To overcome this difficulty, we can add a third term (the smoothing term) in our cost function that favours non-bending surfaces. Note that usually, such terms are used to compensate for the undesirable effects of under-fitting and over-fitting. Doing so is usually a problem because it requires one to determine a correct value for the weight associated to the smoothing term (value β in equation (7.11)). This is a sensible and critical way of balancing the effective complexity of the surface against the complexity of the data. Here, we do not have to care too much. Indeed, the complexity of the surface is limited by the fact that it is inextensible. Any small value (but big enough to be not negligible, for instance $\beta = 10^{-4}$) is thus suitable for the weight of the smoothing term. We define our smoothing term using the bending energy:

$$\mathcal{E}_s(\boldsymbol{\mu},\boldsymbol{\ell}) = \sum_{i=1}^{3} \iint \left\|\frac{\partial^2 \mathcal{W}^i_{\boldsymbol{\ell}}(\mathbf{q})}{\partial \mathbf{q}^2}\right\|^2_{\mathcal{F}} \mathrm{d}\mathbf{q}. \tag{7.13}$$

where $\mathcal{W}^i_{\boldsymbol{\ell}}(\mathbf{q})$ is the i-th coordinate of the point, and $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm of the Hessian matrix. With FFD, there exists a simple and linear closed-form expression for the bending energy:

$$\mathcal{E}_s(\boldsymbol{\ell}) = \|\mathsf{B}^{1/2}\boldsymbol{\ell}\|^2 = \boldsymbol{\ell}^\mathsf{T}\mathsf{B}\boldsymbol{\ell} \tag{7.14}$$

where $\mathsf{B} \in \mathbb{R}^{3p\times 3p}$ is a symmetric, positive, and semi-definite matrix which can be easily computed from the second derivatives of the B-spline basis functions.

7.4.2.1 Initial solution.

The problem of equation (7.11) is a non-linear least-squares minimization problem typically solved using an iterative scheme such as Levenberg-Marquardt. Such an algorithm requires a correct initial solution. We used an FFD surface fitted to the 3D points reconstructed with one of the point-wise methods presented in section 7.3. Subsequently, since we use a surface model which is linear with respect to its parameters, the initial parameters $\boldsymbol{\ell}$ can be found by solving the least-squares problem:

$$\min_{\boldsymbol{\ell}} \sum_{i=1}^{n_c} \left\|\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{q}_i) - \mathbf{Q}_i\right\|^2 \Leftrightarrow \min_{\boldsymbol{\ell}} \sum_{i=1}^{n_c} \|\mathsf{W}_i\boldsymbol{\ell} - \mathbf{Q}_i\|^2 \ . \tag{7.15}$$

An alternative is to modify the problem (7.6), expressing $\mathbf{Q}_i$ in terms of the required parameters $\boldsymbol{\ell}$, according to $\mathbf{Q}_i = \mathsf{W}_i\boldsymbol{\ell}$. Then one may solve for $\boldsymbol{\ell}$ directly using SOCP. If necessary, the linear smoothing term of equation (7.13) can be included in equation (7.15).

7.5 Experimental Results

7.5.1 Experiments on Synthetic Data

In this section, we experiment several aspects of different reconstruction algorithms. We first use synthetic piece of papers, such as those of figure 7.2, randomly generated using the code provided by (Perriollat and Bartoli, 2007). The piece of papers are square and 200mm wide. The input images are simulated by projecting the deformed piece of paper with a virtual camera placed at approximately 1 meter of the paper sheet and with a focal length of 36mm. A set of n_c point correspondences are generated by taking random locations on the 3D surface. A zero mean Gaussian noise with standard deviation of 1 pixel is added to the point correspondences. There are no self-occlusion in the data.

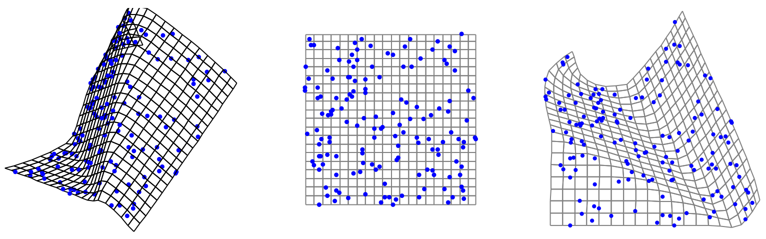


Figure 7.2: Example of randomly generated piece of paper. Left: 3D surface. Middle: template image. Right: input image. The blue dots are examples of point correspondences.

Several algorithms are compared in our experiments:

- SOCPimg: our point-wise method described in section 7.3.2 ;
- FFDref: our smooth reconstruction algorithm described in section 7.4.2 ;

- FFDinit: the initial solution of our smooth reconstruction algorithm, as described in section 7.4.2 ;

- Salz: the convex formulation proposed in (Salzmann et al., 2009). This method is similar to SOCPimg except for the noise that is not handled the same way. In (Salzmann et al., 2009), the author minimizes a cost function that includes a 'reprojection error' in order to cope with the noise. In SOCPimg, the noise is handled with hard constraints.

- PerrioInit: the 'upper depth bound' approach of (Perriollat et al., 2008, 2010) which is a point-wise algorithm that iteratively enforces the inextensibility constraints ;

- PerrioRef: the 'refined approach' of (Perriollat et al., 2008, 2010) which minimizes a cost function resulting in a refined estimation of the 3D points obtained with PerrioInit.

7.5.1.1 Reconstruction Errors

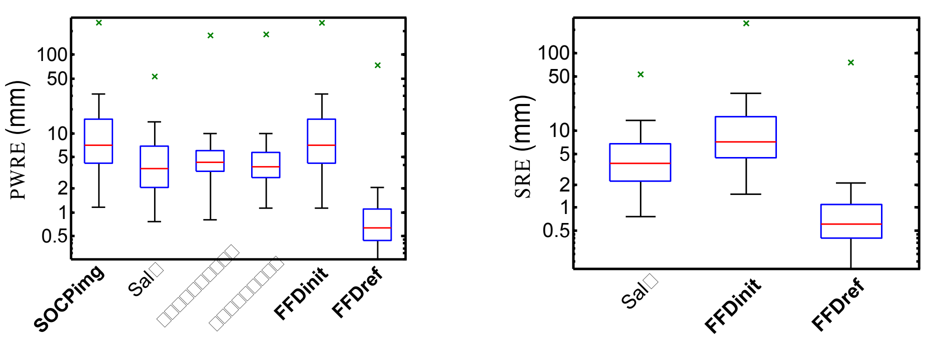
The discrepancy between the reconstructed and the ground truth surfaces are quantified with two measures, depending on the surface model used by the algorithms. The *point-wise reconstruction error* (PWRE), denoted e_p, can be used for all the algorithms. It is defined by:

$$e_p = \frac{1}{n_c} \sum_{i=1}^{n_c} \|\mathbf{Q}_i - \hat{\mathbf{Q}}_i\|. \tag{7.16}$$

For algorithms that uses more complex surface models, such as triangular meshes or FFD, we measures the *surface reconstruction error* (SRE), denoted e_s. It is the difference between the reconstructed surface $\mathcal{W}_{\boldsymbol{\ell}}$ and the ground truth surface $\hat{\mathcal{W}}$:

$$e_s = \iint \left\|\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{q}) - \hat{\mathcal{W}}(\mathbf{q})\right\| \mathrm{d}\mathbf{q}. \tag{7.17}$$

In this experiment, we use 1,000 randomly generated paper sheets with 150 points correspondences. Figure 7.3 (a) shows the PWRE for all the algorithms and figure 7.3 (b) shows the SRE for the algorithms that use a complex surface model. The main result of this experiment is that our approach FFDref gives the smallest reconstruction errors (PWRE and SRE). Globally, the methods that use complex surface models get better results than the point-wise approaches.



(a) Point-wise reconstruction error (b) Surface reconstruction error

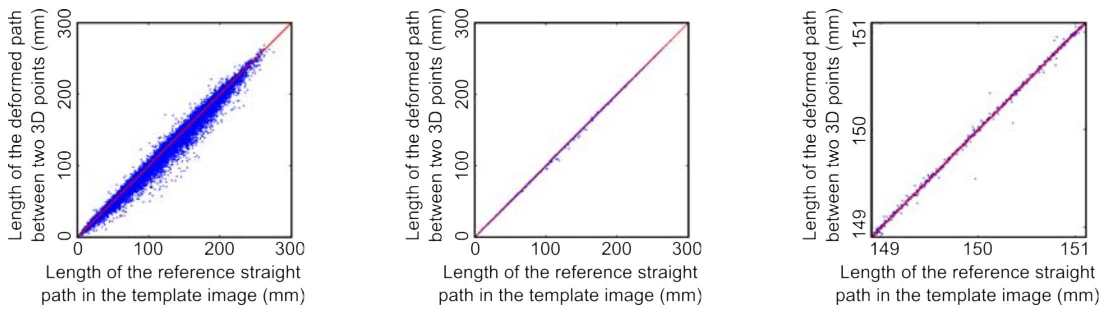
Figure 7.3: Comparison of the reconstruction errors for different algorithms. The central red line is the median. The limits of the blue box are the 25th and the 75th percentiles. The black 'whiskers' cover approximately 99.3% of the experiment outcomes. The green crosses are the maximal errors over the 1000 trials.

7.5.1.2 Length of Geodesics

When a reconstructed 3D surface is reconstructed in a truly inextensible way, the transformation of the straight line linking two points in the template image must be the geodesic linking the corresponding two 3D points on the surface. In particular, the length of these two paths must be identical. Testing this hypothesis for our algorithms FFDinit and FFDref is the goal of this experiment. To do so, we use the same data than in the previous experiment. For each surface, we choose randomly 10,000 pairs of points in the template image. For each pair of points $(\mathbf{g}_i, \mathbf{g}_j)$, the length l_{ij}^{3D} of the deformed path linking the 3D points $\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{g}_i)$ and $\mathcal{W}_{\boldsymbol{\ell}}(\mathbf{g}_j)$ on the surface is approximated with the following formula:

$$l_{ij}^{3D} = \sum_{k=1}^{n_g} \left\| \mathcal{W}_{\boldsymbol{\ell}}(\mathbf{g}_i + \tfrac{k}{n_g}\|\mathbf{g}_j - \mathbf{g}_i\|) - \mathcal{W}_{\boldsymbol{\ell}}(\mathbf{g}_i + \tfrac{k-1}{n_g}\|\mathbf{g}_j - \mathbf{g}_i\|) \right\|, \qquad (7.18)$$

where n_g is the number of intermediate points used for the approximation (we use $n_g = 200$ since we experimentally observed that the approximation stabilizes for values of n_g greater than 180). The lengths of the deformed paths are plotted against their reference length in the template image in figure 7.4 (a) for FFDinit and in figure 7.4 (b,c) for FFDref. Figure 7.4 (b) and figure 7.4 (c) show that, with the surfaces reconstructed with FFDref, the length of the deformed paths are almost equal to the length they should have if they were actual geodesics. In other words, our approach FFDref reconstructs 3D surfaces which are truly inextensible. On the other hand, figure 7.4 (a) shows that the initial solution FFDinit (which is just an FFD fitted to a sparse set of reconstructed 3D points) seems to be much less inextensible.



(a) FFDinit (b) FFDref (c) Magnification of (b)

Figure 7.4: Plot of the length of deformed paths against the length they should have if the reconstructed surface was truly inextensible. The red diagonal line is the place where all the blue points should be for inextensible surfaces.

Let l_{ij}^{2D} be the Euclidean distance between the points $\mathbf{g}_i$ and $\mathbf{g}_j$. Table 7.2 gives some statistics on the relative error between the computed length l_{ij}^{3D} and the reference length l_{ij}^{2D}, *i.e.* the quantity $(l_{ij}^{2D} - l_{ij}^{2D})/l_{ij}^{3D}$. These numbers confirm the results seen in figure 7.4.

	Mean	**Std deviation**	**Median**	**Minimun**	**Maximum**
FFDinit	0.0119	0.0417	0.0036	-1.9689	0.8931
FFDref	2.0084×10^{-5}	7.1965×10^{-4}	5.8083×10^{-6}	-0.0505	0.3396

Table 7.2: Statistics on the relative errors between the length of transformed paths and the length they should have.

7.5.1.3 Gaussian curvature

The Gaussian curvature is the product of the two principal curvature (which are the reciprocal of the radius of the osculating circle). For an inextensible surface, the Gaussian curvature is null. In this experiment, we check if this property is satisfied by the smooth surfaces reconstructed with FFDinit and FFDref. We used the same

	Mean	Std deviation	Median	Minimun	Maximum
FFDinit	4.9458×10^{-4}	0.0875	9.7302×10^{-5}	7.5122×10^{-14}	258.2379
FFDref	5.0046×10^{-6}	7.1320×10^{-4}	1.7333×10^{-6}	2.2325×10^{-14}	1.5199

Table 7.3: Statistics on the (absolute value of the) Gaussian curvatures for $1{,}000$ reconstructed surfaces and $10{,}000$ points per surface.

$1{,}000$ reconstructed surfaces than in the previous experiment. The Gaussian curvature, denoted κ, is computed for $10{,}000$ randomly chosen points on the surface with the formula $\kappa = \frac{\det(\mathbf{II})}{\det(\mathbf{I})}$, where $\mathbf{I}$ and $\mathbf{II}$ are the first and the second fundamental forms of the parametric surface (Gray, 1997). The results of this experiment are reported in table 7.3. It shows that, in average, the Gaussian curvature of the surfaces reconstructed using FFDref are consistently close to 0. It also shows that FFDref gives Gaussian curvatures which are 100 times smaller than the ones obtained with FFDinit. These results demonstrate that the surfaces reconstructed with our approach FFDref are indeed inextensible. Note that this kind of experiment cannot be achieved if a smooth surface is not available.

7.5.2 Experiments on Real Data

The algorithms used in the synthetic experiments of section 7.5.1 are applied to real data in figure 7.5 and figure 7.6. These figures shows that our approaches gives good results on real data. In particular, figure 7.5 shows that our method FFDref outperforms the other approaches in the presence of a self-occlusion. This comes from the fact that FFDref requires the surface to be inextensible everywhere, even if there are no point correspondences (which is the case on the self-occluded part of the paper sheet). An accurate stereo reconstruction of the surface in figure 7.6 were available. We compare in table 7.4 the average 3D errors between the surfaces reconstructed with a monocular approach to the stereo reconstruction. Again, our method FFDref is the one giving the best results.

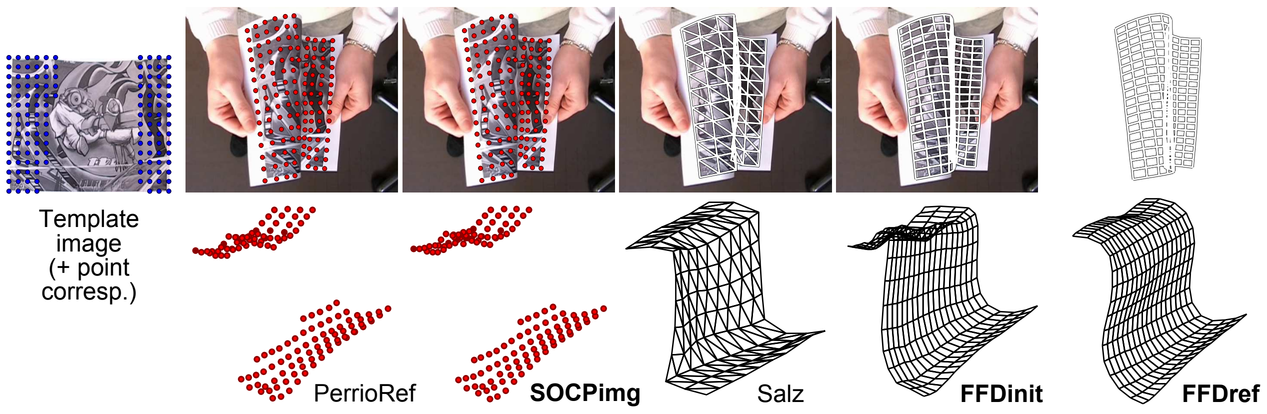


Figure 7.5: Illustration of monocular reconstruction algorithms in the presence of a self-occlusion (the point correspondences were automatically extracted using (Gay-Bellile, 2008)). Note how our algorithm FFDref is able to recover a reasonable shape for the occluded part.

PerrioRef	SOCPimg	Salz	FFDinit	FFDref
2.388	2.261	4.743	2.259	**1.991**

Table 7.4: Average 3D error (in millimeters) with respect to the stereo reconstruction of the surface for the surfaces of figure 7.6.

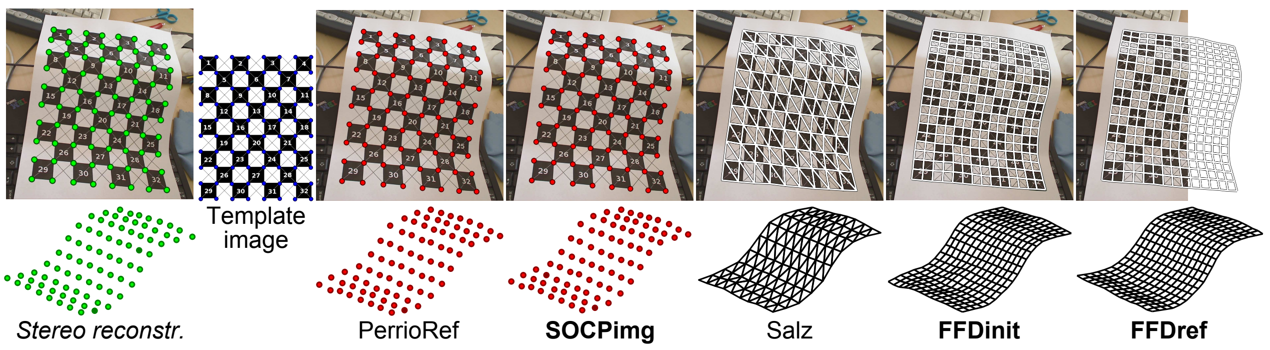


Figure 7.6: Illustration of the results obtained with several monocular reconstruction algorithms. First row: input image along with a reprojection of the reconstructed 3D surface. Second row: reconstructed surface from a different point of view. Note that the stereo reconstruction (first column) is not a monocular algorithm: it is just used to assert the quality of the other reconstructed surfaces (see table 7.4).

7.6 Conclusion

In this chapter, we presented new approaches for monocular reconstruction of inextensible surfaces imaged by a perspective camera. In particular, we proposed a SOCP formulation of the problem that accounts for noise in both the template and the input images. We also designed an algorithm that directly reconstruct a smooth surface based on free-form deformations. This algorithm outperforms previous approaches in terms of precision of the reconstructed surface. Besides, we experimentally showed that the surfaces reconstructed with this algorithm are truly inextensible. The only drawback of this approach is that it is formulated as a non-linear least-squares minimization problem with a non-convex cost function. However, we proposed a method to build an initial solution which is close to the optimum. It allows us to get rid of the difficulties linked to the non-convexity of the cost function.

Chapter 8

Conclusion

Achievements. The work presented in this thesis deals with the modelling and the estimation of parametric models in surface fitting, image registration, and 3D reconstruction. We have presented various contributions related to different topics in Computer Vision: range surface fitting, feature-based and direct image registration, 3D reconstruction of a deforming surface. Several aspects have been considered from the design of parametric models to the estimation of the parameters and the hyperparameters themselves. Although these contributions may appear quite different, they are linked together in different ways. First, all of our contributions consider a deformable environment. This is interesting since over the past few decades Computer Vision has produced, for the topics we have considered, useful and effective results but mainly in rigid environments. Second, it is clear that our contributions all point at a single objective: the reconstruction of arbitrary deformable surfaces. Indeed, surface fitting is a general problem which is useful to convert a sparse set of 3D points into a smooth and analytical representation of the same surface. Image registration (whatever approach is used) is generally one of the early steps in 3D surface reconstruction algorithms. In this thesis, we surely not have solved the ultimate goal of Computer Vision that would be the 3D reconstruction of arbitrary surfaces in deformable environments but our contributions constitute a step towards this goal.

Synthesis of our contributions. As said previously, we have made contributions linked to several topics of Computer Vision. These contributions include the following items.

Surface fitting. We have proposed several methods to fit a smooth parametric surface to range data. These algorithms are efficient in the sense that they provide accurate results in reasonable computational times. This has been made possible by a correct modelling of the considered data. For instance, we have taken into account the fact that the typical noise of a depth map acquired with a ToF camera is heteroskedastic. We have also exploited the particular properties of the tensor-product surface model to achieve good computational performance when the data are arranged on a regular grid.

Image registration. We have made several contributions related to image registration ranging to a general model of warp able to model perspective effects and deformable environment to specific technique to parameter and hyperparameter estimations. More precisely, we have exploited the properties of the NURBS to efficiently model deformations appearing under perspective image conditions. Concerning the estimation of the parameters, we have proposed contributions related with the two main approaches to image registration: the direct approach and the feature-based approach. In the direct approach, we have presented a new algorithm that allows one to discard the delicate problem related to the region of interest. This has been made possible by an adequate modelling of the so-called off-target pixels which consisted in saying that such pixels may be seen as classical outliers. Therefore, a standard robust estimation framework based on M-estimators allowed us to have a unified processing of all the pixels. In feature-based image registration, we have proposed a new approach to tune any hyperparameters that may intervene in such problems.

3D Surface reconstruction. The purpose of our last contribution has been to reconstruct a 3D surface in a deformable environment from a monocular video. It is well known that such a problem is intrinsically ill-posed since potentially there exists an infinite number of surfaces having the same projection in an image. We thus considered common assumptions which consisted in saying that the surface to reconstruct was inextensible and that a reference shape was known. Given these assumptions, we proposed two algorithms to solve the reconstruction problem. First, we proposed a point-wise algorithm, *i.e.* an algorithm that reconstructed only a sparse set of 3D points. This first algorithm is formulated as an SOCP problem. Second, we proposed a method that directly reconstruct a smooth parametric surface. It is formulated as a standard least-squares minimization problem. Our main contribution was to propose a new term enforcing the inextensibility of the reconstructed surface.

Hyperparameter selection. The automatic selection of hyperparameters has been an important transversal topic in the work presented in this thesis. We have emphasized the fact that choosing proper hyperparameters is important to get correct results. We have also shown that this choice may not be always trivial. In this thesis, hyperparameters have been handled in several contexts. We have proposed two different ways of automatically tuning the hyperparameters in range surface fitting: the L-Tangent Norm criterion and an adaptation of Morozov's discrepancy principle. In feature-based image registration, we have proposed a new framework to automatically select any hyperparameter. It relies on the fact that in feature-based image registration, the point correspondences are not the only available data: there is also the photometric information contained in the images. Therefore, we have proposed to use the point correspondences as a training set for the natural parameters and the photometric information as a test set for the hyperparameters.

What's next? The ultimate goal of automatically reconstructing an arbitrary deforming surface has not been reached yet. There is still a long way to go before any surface in any context could be reconstructed from images. Even though the design of such a ultimate algorithm would be a nice proceeding to this thesis, we also focus on shorter term goals.

Range surface fitting. Even though we have proposed efficient algorithms for fitting surfaces on range data, there are some problems which still await to be solved. In particular, handling discontinuities is a major issue in range surface fitting. Several approaches may be considered, such as using a different parametric surface model that would be able to properly model discontinuities. Such a model would require one to design specific parameter estimation techniques.

Hyperparameters in feature-based image registration. We have proposed a generic framework to automatically determine proper hyperparameters in feature-based image registration. However, the criteria resulting of our idea are generally difficult to optimize. In particular, they may have several local minima and they may not be continuous. An automatic procedure to minimize the proposed criterion would be a natural conclusion to this work. Given the nature of the cost function to optimize, combinatorial optimization based on metaheuristics (such as simulated annealing, Kangaroo's algorithm, or genetic algorithms (Fleury and Gourgand, 1998)) would be a good starting point to solve this problem.

Monocular reconstruction of smooth surface. We have proposed an effective way to reconstruct smooth and inextensible surfaces from a monocular sequence of images. This has been made possible by introducing a term in the cost function that penalizes departure from pure inextensibility. Although efficient, this approach has an important computational cost. Other ways of enforcing inextensibility should be explored. In particular, noticing that an inextensible surface has a null Gaussian curvature may be a good starting point. Besides, not all surfaces are inextensible. Other types of constraints should also be envisaged.

Regularization terms. As we have showed in this thesis, it is often required to use a regularization term in order to have well-posed problems. We often used regularization terms based on the bending energy, regardless of the problem's nature. Other regularization terms could be useful. In particular, it would be interesting to study regularization terms of order higher than two. It would also be interesting to use multiple regularization terms and select the most appropriate one based on the data.

Appendix A

Feature-Driven Direct Non-Rigid Image Registration

In this appendix, we report an article which has been submitted to the International Journal of Computer Vision. This article has been recently accepted with minor revisions. The work presented in this article has been done in collaboration with Vincent Gay-Bellile[1] and Adrien Bartoli. Note that the article is reproduced without any modification compared to the journal version (except for the numbering of the sections). In particular, this implies that there might be some slight differences in the notation compared to the rest of this document.

[1]CEA LIST, Embedded Vision Systems Laboratory, Point Courrier 94, Gif-sur-Yvette, F-91191 France

Abstract. The direct registration problem for images of a deforming surface has been well studied. Parametric flexible warps based, for instance, on the Free-Form Deformation or a Radial Basis Function such as the Thin-Plate Spline, are often estimated using additive Gauss-Newton-like algorithms. The recently proposed compositional framework has been shown to be more efficient, but cannot be directly applied to such non-groupwise warps.

Our main contribution in this paper is the Feature-Driven framework. It makes possible the use of compositional algorithms for most parametric warps such as those above mentioned. Two algorithms are proposed to demonstrate the relevance of our Feature-Driven framework: the Feature-Driven Inverse Compositional and the Feature-Driven Learning-based algorithms. As another contribution, a detailed derivation of the Feature-Driven warp parameterization is given for the Thin-Plate Spline and the Free-Form Deformation. We experimentally show that these two types of warps have a similar representational power. Experimental results show that our Feature-Driven registration algorithms are more efficient in terms of computational cost, without loss of accuracy, compared to existing methods.

A.1 Introduction

Registering images of a deforming surface is important for tasks such as video augmentation by texture editing, deformation capture and non-rigid Structure-from-Motion. This is a difficult problem since the appearance of imaged surfaces varies due to several phenomena such as camera pose, surface deformation, lighting and motion blur. Recovering a 3D surface, its deformations and the camera pose from a monocular video sequence is intrinsically ill-posed. While prior information can be used to disambiguate the problem, see *e.g.* (Bregler et al., 2000; Gay-Bellile et al., 2006; Pilet et al., 2005), it is common to avoid a full 3D model by using image-based deformation models, *e.g.* (Bartoli and Zisserman, 2004; Bookstein, 1989; Cootes et al., 1998; Lim and Yang, 2005). The Thin-Plate Spline warps (TPS) is one possible deformation model, proposed in a seminal paper by (Bookstein, 1989), that has been shown to effectively model a wide variety of image deformations in different contexts. Recent work shows that the TPS warp can be estimated not only with the traditional landmark based method, but also with direct methods, *i.e.* by minimizing the intensity discrepancy between registered images (Bartoli and Zisserman, 2004; Lim and Yang, 2005). Other non-rigid warps include Radial Basis Functions (with *e.g.* multiquadrics (Little et al., 1997) or Wendland's (Fornefett et al., 1999) as kernel function) and Free-Form Deformations (Rueckert et al., 1999).

The Gauss-Newton algorithm with additive update of the parameters is usually used for conducting the minimization. Its main drawback is that the Hessian matrix must be recomputed and inverted at each iteration. More efficient solutions have been proposed by (Baker et al., 2004) based on compositional updating of the parameters. They might lead to a constant Hessian matrix. Most non-rigid warps do not form groups, preventing the use of compositional algorithms which require one to compose and possibly invert the warps. Despite several attempts to relax the groupwise assumption by various approximations (Gay-Bellile et al., 2006; Matthews and Baker, 2004; Romdhani and Vetter, 2003), there is no simple solution in the literature.

This paper is an extended version of an earlier conference version (Gay-Bellile et al., 2007). With respect to the literature, it brings several contributions:

- The main contribution of this paper is the *Feature-Driven registration concept*. It allows one to devise compositional algorithms while relaxing the strict groupwise warp requirement, and is thus applicable to most non-rigid parametric warps. The main idea is to parametrize the warp by a set of *driving features* instead of the usual control points or coefficients, and to act on these features directly.

- Two operations, *reversion* and *threading*, are defined using the Feature-Driven parametrization. They respectively approximate inversion and composition when those are not guaranteed to exist or cannot be easily computed.
- Using our Feature-Driven framework, we extend the Inverse Compositional algorithm to non-rigid warps. Our framework also allows us to propose a forward Learning-based algorithm for non-rigid warps. Besides, we improve the classical linear learning algorithm using a new piecewise linear relationship.
- We give a detailed derivation of the Feature-Driven parametrization for the TPS and FFD warps. In particular, we present an extrapolation method for the FFD warp, required for the Feature-Driven framework. We also show that the TPS and FFD warps have a similar representational power.

We experimentally show that Feature-Driven algorithms are clearly more efficient without loss of accuracy compared to previous state-of-the-art methods. The combination of the Feature-Driven framework with Learning-based local registration outperforms other algorithms for most experimental setups.

Roadmap. Previous work is reviewed in section A.2. In particular, previous attempts to extending compositional algorithms to non-groupwise warps are presented in section A.2.2. The Feature-Driven framework and the associated operations are explained in section A.3. Registration with the Feature-Driven Inverse Compositional and the Feature-Driven Learning-based algorithms are described in section A.4. The Feature-Driven parametrization of the TPS and of the FFD warps are detailed in section A.5. Experimental results on simulated and real data are reported in section A.6. Conclusions and further work are discussed in section A.7. Details on the piecewise linear relationship we used for the Learning-based local registration step are given in appendix A.7.

Notation. Scalars are in italics (x), vectors in bold ($\mathbf{v}$), matrices in sans-serif (M) and sets (or collections) in fraktur ($\mathfrak{C}$). Vectors are always considered as column vectors. The inverse of a matrix M is written M^{-1}, the pseudo-inverse $\mathsf{M}^{\dagger}$ and the transpose M^{T}. The symbol $\mathbb{R}$ denotes the set of the real numbers. The identity matrix of size n is denoted I_n. The notation $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ corresponds to the matrices of size $m \times n$ filled with zeros and ones respectively. The operator that vectorizes a matrix is denoted $\boldsymbol{\nu}$, *i.e.* $\boldsymbol{\nu}(\mathsf{M}) = \left(\mathbf{m}_1^{\mathsf{T}} \ \dots \ \mathbf{m}_l^{\mathsf{T}}\right)^{\mathsf{T}}$ where the vectors $\{\mathbf{m}_i\}_{i=1}^{l}$ are the columns of M. Conversely, the operator ζ_p builds a matrix of size $\mathbb{R}^{q\times p}$ from a vector of size $\mathbb{R}^{pq}$, *i.e.* $\zeta_p(\mathbf{v}) = (\mathbf{v}_1 \ \dots \ \mathbf{v}_p) \in \mathbb{R}^{q\times p}$ where $\mathbf{v} = \left(\mathbf{v}_1^{\mathsf{T}} \ \dots \ \mathbf{v}_l^{\mathsf{T}}\right)^{\mathsf{T}} \in \mathbb{R}^{pq}$. The notation ζ is used to abbreviate ζ_2. We denote rms$(\mathbf{v})$ the Root Mean of Squares (RMS) of the m-vector $\mathbf{v}$, *i.e.* rms$(\mathbf{v}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \mathbf{v}_i^2} \propto \|\mathbf{v}\|$, with $\| \bullet \|$ the two-norm.

Images are considered as $\mathbb{R}^2 \to \mathbb{R}$ functions[2] and are denoted using calligraphic fonts ($\mathcal{A}$). If $\mathfrak{C}$ is a collection of pixels then $\boldsymbol{\xi}_{\mathfrak{C}}(\mathcal{I})$ is the vector in which are stacked the values of $\mathcal{I}$ for all the pixels indicated in $\mathfrak{C}$. More precisely, if $\mathfrak{C} = \{\mathbf{q}_i\}_{i=1}^{|\mathfrak{C}|}$ then $\boldsymbol{\xi}_{\mathfrak{C}}(\mathcal{I}) = \left(\mathcal{I}(\mathbf{q}_1) \ \dots \ \mathcal{I}(\mathbf{q}_{|\mathfrak{C}|})\right)^{\mathsf{T}} \in \mathbb{R}^{|\mathfrak{C}|}$ where $|\mathfrak{C}|$ is the cardinal of the set $\mathfrak{C}$.

The images to be registered are written $\mathcal{I}_i$ with $i = 1, \dots, n$. The texture image, *e.g.* the region of interest in the first image, is denoted $\mathcal{I}_0$. The set of pixels of interest, *i.e.* the subset of pixels of the image $\mathcal{I}_0$ actually used to estimate a warp, is denoted $\mathfrak{R}$. A generic parametric warp is written $\mathcal{W}$. It depends on a parameter vector $\mathbf{u}_i$ for image $\mathcal{I}_i$ and maps a point $\mathbf{q}_0$ from the texture image to the corresponding point $\mathbf{q}_i$ in the i-th image: $\mathbf{q}_i = \mathcal{W}(\mathbf{q}_0; \mathbf{u}_i)$. The notation $\mathcal{W}(\mathbf{q}; \bullet)$ designates the warp as a function of its parameters, *i.e.* an $\mathbb{R}^l \times \mathbb{R}^2$ function where l is the size of the parameter vector, instead of as a function of the pixels.

[2]In practice, images are $\mathbb{R}^2 \to \mathbb{R}^c$ functions where c is the number of channels. For the sake of simplicity and without loss of generality, we consider that $c = 1$ in all the derivations of this article.

A.2 Problem Statement and Previous Work

The registration of images of deformable surfaces has received a growing attention over the past decade. Usually, for purely twodimensional registration, as is the case in this paper, smoothness constraints are used to filter out the noise and 'fill-in' the optical flow in untextured image areas. These soft constraints are either implicitly incorporated in a parameterized warp or enforced through regularization. In this paper, we focus on direct as opposed to feature-based methods, *e.g.* (Pilet et al., 2005; Torr and Zisserman, 1999). In the feature-based methods, the warp parameters are estimated from features, such as points, which have first been extracted and matched in the images to register (Szeliski, 2006). Note that there exist methods that mix both the feature-based and the direct approaches to image registration. See for instance, (Georgel et al., 2008; Johnson and Christensen, 2002).

Direct registration consists in minimizing the pixel value discrepancy. Registration of an image sequence is posed as a set of nonlinear optimization problems, each of which estimating $\mathbf{u}_i$ using the registration $\mathbf{u}_{i-1}$ of the previous frame as an initial solution. The discrepancy function $\mathcal{C}$ is usually chosen as the two-norm of the difference $\mathcal{D}$ between the texture image and the current one, warped towards the texture image, *i.e.* $\mathcal{D}(\mathbf{q};\mathbf{u}_i) = \mathcal{I}_0(\mathbf{q}) - \mathcal{I}_i(\mathcal{W}(\mathbf{q};\mathbf{u}_i))$, giving:

$$\mathcal{C}(\mathbf{u}_i) = \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{D}(\mathbf{q};\mathbf{u}_i)\|^2 . \tag{A.1}$$

Other choices are possible for the cost function, such as the Mutual Information, see *e.g.* (Meyer et al., 1997; Pluim et al., 2003), or a criterion based on the gradient of images, see (Haber and Modersitzki, 2006).

Several algorithms have been proposed to minimize $\mathcal{C}$. We classify them in two groups: the Forward Additive algorithms and the Inverse Compositional ones.

A.2.1 Forward Additive Algorithms

Forward Additive Gauss-Newton (FA-GN). Using an additive update of the parameter vector, *i.e.* $\mathbf{u}_i \leftarrow \mathbf{u}_i + \boldsymbol{\delta}$, Gauss-Newton can be used in a straightforward manner for minimizing equation (A.1) or in conjunction with complexity tuning schemes as in (Bartoli and Zisserman, 2004; Lim and Yang, 2005) for the TPS warp. The local Gauss-Newton approximation to $\mathcal{C}$ is given by the first order Taylor expansion in $\boldsymbol{\delta}$ of each squared term in equation (A.1):

$$\mathcal{C}(\mathbf{u}_i + \boldsymbol{\delta}) \approx \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{D}(\mathbf{q};\mathbf{u}_i) + \mathbf{g}(\mathbf{q};\mathbf{u}_i)^\mathsf{T}\boldsymbol{\delta}\|^2. \tag{A.2}$$

The gradient vector $\mathbf{g}$ is the product of the image gradient vector and of the Jacobian matrix K of the warp, *i.e.* $\mathbf{g} = \nabla\mathcal{I}_i^\mathsf{T}\mathsf{K}$. The Gauss-Newton approximation induces a Linear Least Squares minimization problem in $\boldsymbol{\delta}$. Defining J as the Jacobian matrix of the error, obtained by stacking the gradient vectors $\mathbf{g}(\mathbf{q};\mathbf{u}_i)^\mathsf{T}$ for all the pixels $\mathbf{q}$ in $\mathfrak{R}$, and $\mathbf{d} = \xi_{\mathfrak{R}}(\mathcal{D}(\,\bullet\,;\mathbf{u}_i))$ the residual error vector, the solution is obtained through the normal equations:

$$\mathsf{H}\boldsymbol{\delta} = -\mathbf{b} \quad \text{with} \quad \mathsf{H} = \mathsf{J}^\mathsf{T}\mathsf{J} \quad \text{and} \quad \mathbf{b} = \mathsf{J}^\mathsf{T}\mathbf{d}. \tag{A.3}$$

The matrix H is the Gauss-Newton approximation to the Hessian matrix. Note that J, H and $\mathbf{d}$ depend on $\mathbf{u}_i$. The Jacobian matrix J must be recomputed at each iteration, implying that H must be recomputed and inverted as well.

Forward Additive ESM (FA-ESM). A second order approximation of $\mathcal{C}$ called ESM (Efficient Second-order Minimization), theoretically better than the Gauss-Newton one, is proposed in (Benhimane and Malis, 2004). Combined with an additive update of the parameters, it gives:

$$\mathcal{C}(\mathbf{u}_i + \boldsymbol{\delta}) \approx \sum_{\mathbf{q}\in\mathfrak{R}} \left\| \mathcal{D}(\mathbf{q};\mathbf{u}_i) + \tfrac{1}{2}\left(\mathbf{g}(\mathbf{q};\mathbf{u}_i) + \mathbf{g}(\mathbf{q};\mathbf{u}_0)\right)^{\mathsf{T}} \boldsymbol{\delta} \right\|^2 . \tag{A.4}$$

The ESM approximation has been shown to improve the convergence rate, compared to Gauss-Newton, without increasing the computation time per iteration since the gradient vectors $\mathbf{g}(\mathbf{q};\mathbf{u}_0)$ are constant.

A.2.2 Inverse Compositional Algorithms

The major drawback of the two above-presented methods is that the image gradient vector for each pixel in $\mathfrak{R}$ must be recomputed at each iteration. This is the most expensive step of the process. A major improvement was proposed by (Baker et al., 2004) with the Inverse Compositional algorithm. The first key idea consists in switching the roles of the texture and of the current images:

$$\min_{\tilde{\mathbf{u}}} \sum_{\mathbf{q}\in\mathfrak{R}} \left\| \mathcal{I}_0\left(\mathcal{W}(\mathbf{q};\tilde{\mathbf{u}})\right) - \mathcal{I}_i\left(\mathcal{W}(\mathbf{q};\mathbf{u}_i)\right) \right\| . \tag{A.5}$$

The second idea is to update the current warp by composition:

$$\mathcal{W}(\,\bullet\,;\mathbf{u}_i) \leftarrow \mathcal{W}(\,\bullet\,;\mathbf{u}_i) \circ \mathcal{W}^{-1}(\,\bullet\,;\tilde{\mathbf{u}}). \tag{A.6}$$

Using Gauss-Newton for local registration leads to a constant Jacobian matrix and a constant Hessian matrix whose inverse is thus pre-computed. Of course, the inverted warp $\mathcal{W}^{-1}$ exists only if the considered set of warps is a group. For instance, homographic warps are used in the original Inverse Compositional algorithm (Baker et al., 2004). In this case, inversion is obtained by inverting the associated 3×3 matrix and composition by multiplying those matrices. Several attempts have been made to relax the groupwise requirement for flexible models.

As proposed in (Gay-Bellile et al., 2006; Matthews and Baker, 2004; Romdhani and Vetter, 2003), these attempts usually consist in finding the best approximating warp for the pixels of interest in $\mathfrak{R}$:

$$\mathbf{u}_i \leftarrow \arg\min_{\mathbf{u}'_i} \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{W}(\mathcal{W}(\mathbf{q};\mathbf{u});\mathbf{u}_i) - \mathcal{W}(\mathbf{q};\mathbf{u}'_i)\|^2. \tag{A.7}$$

In (Matthews and Baker, 2004; Romdhani and Vetter, 2003), the warp is induced by a triangular mesh whose deformations are guided by a parameter vector. This minimization problem is usually solved in two steps. First the vertices in the current image are computed using the assumption of local rigidity. They usually are not in accordance with a model instance in *e.g.* the case of 3D Morphable Model (Gay-Bellile et al., 2006; Romdhani and Vetter, 2003). Second, the parameter update is recovered by minimizing a prediction error, *i.e.* the distance between the updated vertices and those induced by the parameters. This last step may be time consuming since nonlinear optimization is required. Warp inversion is approximated with first order Taylor expansion in (Matthews and Baker, 2004), while (Romdhani and Vetter, 2003) draws on triangular meshes to avoid linearization. By comparison, our method reverts and threads warps in closed-form: it does not require optimization.

Other methods have been proposed to obviate the shortcomings induced by non-groupwise warps. For instance, one may force the solution space to contain diffeomorphic warps (Charpiat et al., 2005; Johnson and Christensen, 2002; Joshi and Miller, 2000). The solution space thus constitutes a group. Requiring the warps to be diffeomorphisms may be an overly strong requirement. This is especially true when the deformations are not too important. Indeed, as noted in (Johnson and Christensen, 2002), with such deformations the estimated warps may be diffeomorphisms even though the solution space contains non-diffeomorphic warps. Besides, such approaches make the use of standard deformation models such as the TPS and the FFD generally impossible. This is one interesting point of the proposed approach in this paper: it proposes an efficient method which is built on top of the most common deformation models for image registration. Finally, enforcing the estimated warps to be diffeomorphisms is often achieved by adding supplementary constraints to an initial forward estimation algorithm. These constraints are generally impractical to design a fast inverse-compositional estimation algorithm.

A.3 Feature-Driven Registration

In this section, we present our principal contribution: the Feature-Driven framework. This framework, in which one directly acts on warp driving features, has two main advantages. First, it often is better balanced to tune feature positions, expressed in pixels, than coefficient vectors that may be difficult to interpret, as for the TPS or the FFD warps. Second, it allows one to use the efficient compositional framework in a straightforward manner. Indeed, warp composition and inversion cannot be directly done for non-groupwise warps. We propose empirical means for approximating warp composition and inversion through their driving features, called threading and reversion respectively. Our Feature-Driven framework is generic in the sense that it can be applied to almost any parametric warps such as the TPS or the FFD warps, as shown in section A.5.

A.3.1 Feature-Driven Warp Parameterization

Ignoring the set of parameters, a warp is an $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ function usually parameterized by a set of n control points $\mathbf{p}_i = (p_i^x \; p_i^y)^\mathsf{T} \in \mathbb{R}^2$ for $i = 1, \ldots, n$. These control points are grouped in a vector $\mathbf{p} = \boldsymbol{\nu}(\mathsf{P}) \in \mathbb{R}^{2n}$ where $\mathsf{P} \in \mathbb{R}^{n \times 2}$ is the matrix defined by $\mathsf{P}^\mathsf{T} = (\mathbf{p}_1 \; \ldots \; \mathbf{p}_n)$. We write ω the warp in its natural parameterization. The warp ω is said to be linear when it can be defined as a linear combination of its control points:

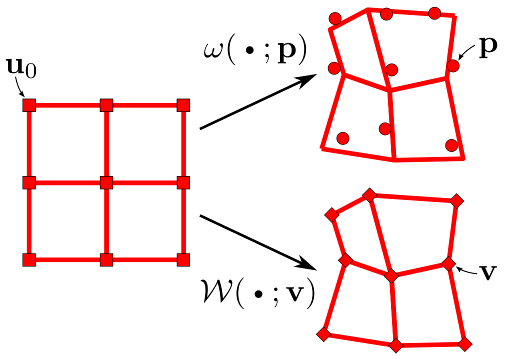
$$\omega(\mathbf{q}; \mathbf{p}) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{P}, \tag{A.8}$$

where $\boldsymbol{\ell}_\mathbf{q} \in \mathbb{R}^n$ is a vector that depends on the point $\mathbf{q}$ and on the type of warp being considered. Note that the dependency on $\mathbf{q}$ of $\boldsymbol{\ell}_\mathbf{q}$ is usually non linear even if the warp is linear. The control points are usually not interpolated. They just act as 'attractors' to the warp. It thus makes their interpretation difficult. The Feature-Driven concept is in fact a change of parameterization. The control points are replaced by a set of features that are interpolated by the warp. We call them the *driving features* and denote them $\mathbf{u}_0$ in the texture image and $\mathbf{v}$ in the current one (see figure A.1). We denote $\mathcal{W}$ the Feature-Driven parameterization of the warp ω. Loosely speaking, matching the driving features between two images is equivalent to defining a warp since the warp can be used to transfer the driving features from one image to the other, while conversely, the warp can be computed from the driving features. Indeed, if the warp is linear with respect to its control points then it is always possible to find a matrix E such that:

$$\mathcal{W}(\mathbf{q}; \mathbf{v}) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E} \mathsf{V}, \tag{A.9}$$

with $\mathsf{V} = \zeta(\mathbf{v})$, *i.e.* V equals to $\mathbf{v}$ reshaped on two columns. Matrix E can be pre-computed. Details on how the matrix E is obtained for the TPS and the FFD warps are given in section A.5.1 and section A.5.2 respectively. If we write $\mu_{\mathbf{q}} = \mathsf{E}^\mathsf{T}\ell_{\mathbf{q}}$ then the Feature-Driven parameterization of the warp $\mathcal{W}$ is given by:

$$\mathcal{W}(\mathbf{q}; \mathbf{v}) = \mu_{\mathbf{q}}^\mathsf{T}\mathsf{V}. \tag{A.10}$$



$\mathbf{u}_0$
$\omega(\bullet\,; \mathbf{p})$
$\mathbf{p}$
$\mathcal{W}(\bullet\,; \mathbf{v})$
$\mathbf{v}$

Figure A.1: Illustration of the Feature-Driven parameterization. The vector $\mathbf{u}_0$ contains the features in the texture image (the centers). $\omega(\bullet\,; \mathbf{p})$ and $\mathcal{W}(\bullet\,; \mathbf{v})$ are two different representations of the same warp. The first one is parameterized with the 'natural' control points $\mathbf{p}$ while the second one is parameterized with the driving features $\mathbf{v}$.

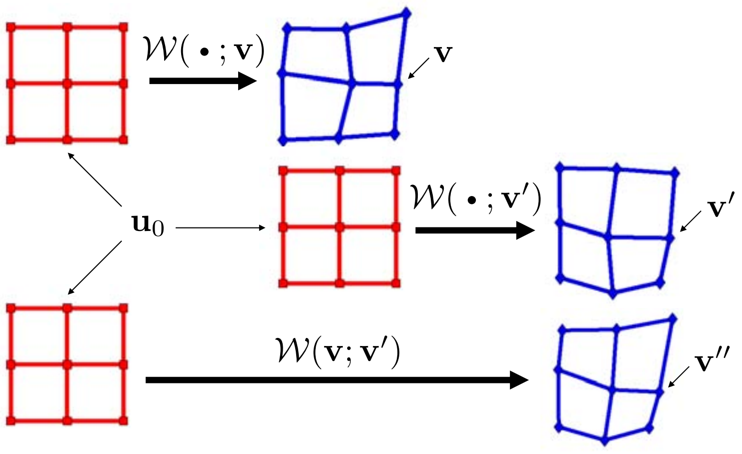
Identity Warp. If we denote $\mathbf{u}_0$ the features in the texture image then $\mathcal{W}(\bullet\,; \mathbf{u}_0)$ is the identity warp[3], *i.e.* the warp that leaves the location of the features $\mathbf{u}_0$ unchanged.

A.3.2 Threading Warps

Given two sets of driving features, $\mathbf{v} = \boldsymbol{\nu}\left((\mathbf{v}_1 \ \dots \ \mathbf{v}_l)^\mathsf{T}\right)$ and $\mathbf{v}' = \boldsymbol{\nu}\left((\mathbf{v}'_1 \ \dots \ \mathbf{v}'_l)^\mathsf{T}\right)$, we want to find a third set $\mathbf{v}'' = \boldsymbol{\nu}\left((\mathbf{v}''_1 \ \dots \ \mathbf{v}''_l)^\mathsf{T}\right)$ defined such that threading the warps induced by $\mathbf{v}$ and $\mathbf{v}'$ results in the warp induced by $\mathbf{v}''$, as shown in figure A.2. We propose a simple and computationally cheap way to do it, as opposed to previous work. This is possible thanks to the Feature-Driven parameterization. Our idea for threading warps is very simple: we apply the $\mathbf{v}'$ induced warp to the features $\mathbf{v}$; the resulting set of features is $\mathbf{v}''$. We thus define the warp threading operator, denoted $\square$, as:

$$\mathcal{W}(\bullet\,; \mathbf{v}) \,\square\, \mathcal{W}(\bullet\,; \mathbf{v}') \overset{\text{def}}{=} \mathcal{W}(\bullet\,; \mathbf{v}'') \qquad \text{with } \mathbf{v}'' = \mathcal{W}(\mathbf{v}; \mathbf{v}'), \tag{A.11}$$

where $\mathcal{W}(\mathbf{v}; \mathbf{v}')$ is meant to be applied to each feature in $\mathbf{v}$.



$\mathcal{W}(\bullet\,; \mathbf{v})$
$\mathbf{v}$
$\mathbf{u}_0$
$\mathcal{W}(\bullet\,; \mathbf{v}')$
$\mathbf{v}'$
$\mathcal{W}(\mathbf{v}; \mathbf{v}')$
$\mathbf{v}''$

Figure A.2: The Feature-Driven warp threading process: $\mathbf{v}''$ is defined by $\mathbf{v}'' = \mathcal{W}(\mathbf{v}; \mathbf{v}')$.

[3]Actually, it can also be a very close approximation, depending on how the matrix E is defined.

Examples of our warp threading process are shown in figure A.3. We synthesized two sets of driving features $\mathbf{v}$ and $\mathbf{v}'$ by randomly disturbing a 3×3 regular grid from its rest position $\mathbf{u}_0$. As expected, threading a warp $\mathcal{W}(\bullet\,;\mathbf{v})$ with the identity $\mathcal{W}(\bullet\,;\mathbf{u}_0)$ returns the original warp *i.e.* $\mathcal{W}(\bullet\,;\mathbf{v}) = \mathcal{W}(\bullet\,;\mathbf{v}) \,\square\, \mathcal{W}(\bullet\,;\mathbf{u}_0)$ and $\mathcal{W}(\bullet\,;\mathbf{v}) = \mathcal{W}(\bullet\,;\mathbf{u}_0) \,\square\, \mathcal{W}(\bullet\,;\mathbf{v})$.

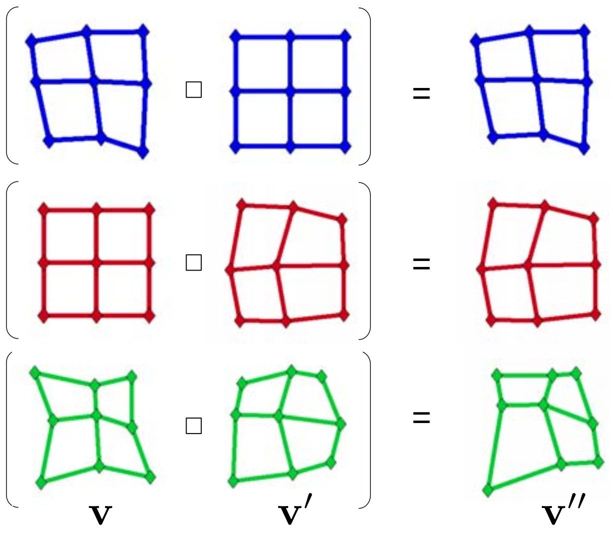


Figure A.3: Examples for the warp threading process.

A.3.3 Reverting Warps

Given a set $\mathbf{v}$ of driving features, we want to determine the features $\mathbf{v}'$ such that the warp they induce is the reversion of the one induced by $\mathbf{v}$. This is illustrated in figure A.4. As for the threading, our Feature-Driven framework yields a very simple solution. The idea is that applying the $\mathbf{v}'$ induced warp to $\mathbf{v}$ should give $\mathbf{u}_0$ *i.e.* , the fixed driving features in the texture image. We thus introduce the reversion operator $\diamond$ as:

$$\mathcal{W}(\bullet\,;\mathbf{v})^{\diamond} \overset{\text{def}}{=} \mathcal{W}(\bullet\,;\mathbf{v}') \qquad \text{with } \mathcal{W}(\mathbf{v};\mathbf{v}') = \mathbf{u}_0, \tag{A.12}$$

This amounts to solving an exactly determined linear system, the size of which is the number of driving features. Using equation (A.10), we obtain:

$$\mathsf{M}\mathsf{V}' = \mathsf{U}_0, \tag{A.13}$$

where $\mathsf{M} \in \mathbb{R}^{l\times l}$ is the matrix defined by $\mathsf{M}^{\mathsf{T}} = \begin{pmatrix}\boldsymbol{\mu}_{\mathbf{v}_1} & \cdots & \boldsymbol{\mu}_{\mathbf{v}_l}\end{pmatrix}$, $\mathsf{V}' = \zeta(\mathbf{v}')$ and $\mathsf{U}_0 = \zeta(\mathbf{u}_0)$. The driving features $\mathbf{v}'$ of the reverted warp are thus given by:

$$\mathbf{v}' = \boldsymbol{\nu}\left(\mathsf{M}^{-1}\mathsf{U}_0\right). \tag{A.14}$$

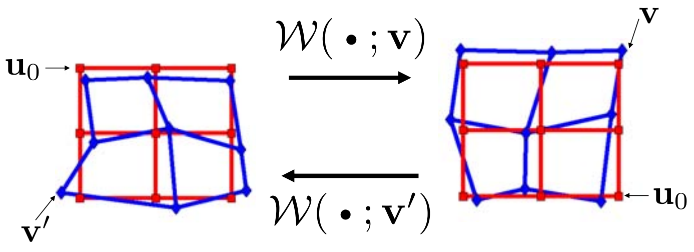

$\mathbf{u}_0$
$\mathcal{W}(\bullet\,;\mathbf{v})$
$\mathbf{v}$
$\mathcal{W}(\bullet\,;\mathbf{v}')$
$\mathbf{u}_0$
$\mathbf{v}'$

Figure A.4: The Feature-Driven warp reversion process: $\mathbf{v}'$ is defined such that $\mathcal{W}(\mathbf{v};\mathbf{v}') = \mathbf{u}_0$.

Examples of the reverting process are shown in figure A.5. We synthesized driving features $\mathbf{v}$ by randomly disturbing a 3×3 regular grid from its rest position $\mathbf{u}_0$. The driving features $\mathbf{v}'$ result from reverting the warp $\mathcal{W}(\bullet\,;\mathbf{v})$. Threading warps $\mathcal{W}(\bullet\,;\mathbf{v})$ and $\mathcal{W}(\bullet\,;\mathbf{v}')$ introduce a new set of driving features $\mathbf{v}''$. As expected, the features $\mathbf{v}''$ are similar to the original grid $\mathbf{u}_0$ with an average residual error of 10^{-13} pixels.

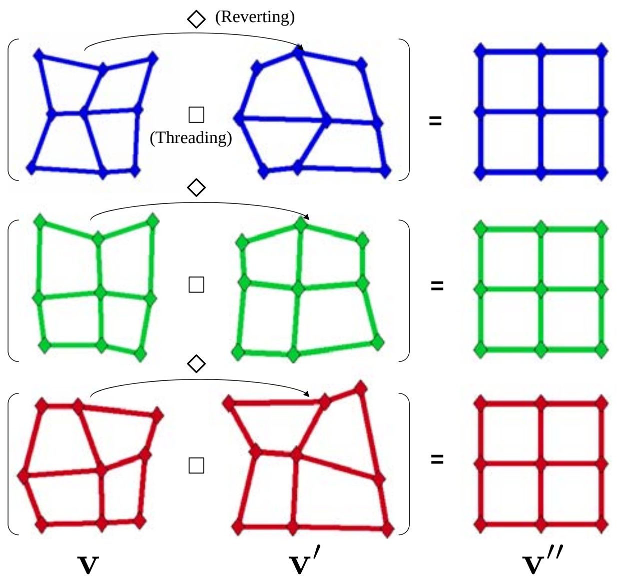


Figure A.5: Illustration of the warp reversion process on three examples.

A.3.4 Compositional Feature-Driven Registration

Relying on the Feature-Driven parameterization properties, we extend compositional algorithms to non-groupwise warps. The following three steps are repeated until convergence, as shown in figure A.6:

- **Step 1: Warping.** The current driving features $\mathbf{u}_i$ are used to warp the input image $\mathcal{I}_i$, thereby globally registering it to the texture image by creating the warped image $\mathcal{I}_W$:

$$\mathcal{I}_W(\mathbf{q}) \overset{\text{def}}{=} \mathcal{I}_i(\mathcal{W}(\mathbf{q};\mathbf{u}_i)). \tag{A.15}$$

- **Step 2: Local registration.** The driving features $\mathbf{u}$ are estimated in the warped image $\mathcal{I}_W$. Several algorithms can be used. They are described in section A.4.1 and section A.4.2. Note that for the Inverse Compositional algorithm, warp reversion is done at this step, based on equation (A.12).

- **Step 3: Updating.** The current driving features $\mathbf{u}_i$ and those in the warped image $\mathbf{u}$ are combined by threading the warps using equation (A.11), to update the driving features $\mathbf{u}_i$ in the current image.

Note that in previous work (Gay-Bellile et al., 2006; Matthews and Baker, 2004; Romdhani and Vetter, 2003) a preliminary step is required before applying the update rule, as reviewed in section A.2.2. In comparison, our Feature-Driven framework makes it naturally included into the third step.

Illumination changes are handled by globally normalizing the pixel values in the texture and the warped images at each iteration. Another approach could be used such as the light-invariant approach of (Pizarro and Bartoli, 2007).

A.4 Local Registration Algorithms

In the literature, there are two ways for estimating local registration: the Forward and the Inverse approaches (Baker et al., 2004). The former evaluates directly the warp which aligns the texture image $\mathcal{I}_0$ with the warped

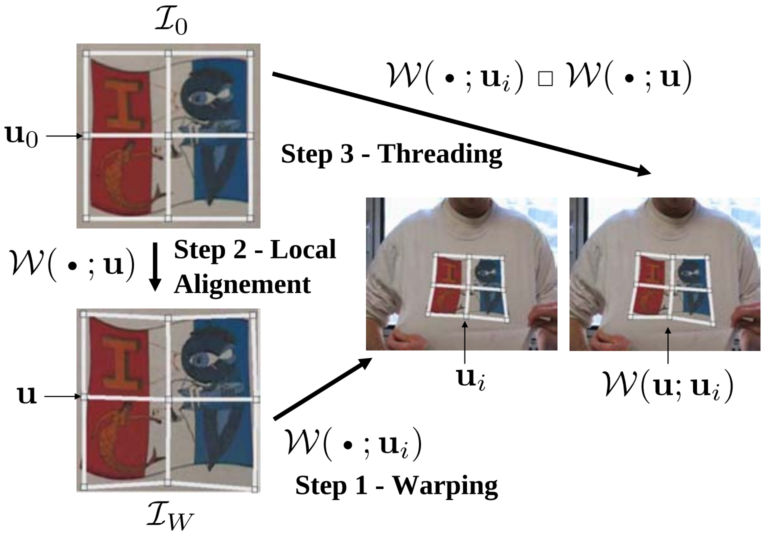

$\mathcal{I}_0$
$\mathcal{W}(\bullet\,;\mathbf{u}_i) \,\square\, \mathcal{W}(\bullet\,;\mathbf{u})$
$\mathbf{u}_0$
Step 3 - Threading
$\mathcal{W}(\bullet\,;\mathbf{u})$
Step 2 - Local Alignement
$\mathbf{u}$
$\mathbf{u}_i$
$\mathcal{W}(\mathbf{u};\mathbf{u}_i)$
$\mathcal{W}(\bullet\,;\mathbf{u}_i)$
Step 1 - Warping
$\mathcal{I}_W$

Figure A.6: The three steps of the Compositional Feature-Driven registration.

image $\mathcal{I}_W$. The latter computes the warp which aligns the warped with the texture image and then inverts the warp. They are both compatible with approximations of the cost function such as Gauss-Newton, ESM, learning-based, *etc*We describe in details the Inverse Gauss-Newton and the Forward Learning-based local registration steps.

A.4.1 Local Registration with Gauss-Newton

Combining an Inverse local registration with a Gauss-Newton approximation of the cost function is efficient since this combination makes invariant the approximated Hessian matrix used in the normal equations to be solved at each iteration. We cast this approach in the Feature-Driven framework, making it possible to extend Inverse Compositional registration to the TPS and the FFD warps.

In the Inverse Compositional framework, local registration is achieved by minimizing the local discrepancy error:

$$\mathcal{C}_l(\tilde{\mathbf{u}}) = \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{I}_0(\mathcal{W}(\mathbf{q};\tilde{\mathbf{u}})) - \mathcal{I}_W(\mathbf{q})\|^2. \tag{A.16}$$

Using Gauss-Newton as local registration engine, the gradient vector is the product of the texture image gradient vector and of the constant Jacobian matrix K of the warp: $\mathbf{g} = \nabla\mathcal{I}_0^{\mathsf{T}}\mathsf{K}$. Matrix K is given in section A.5. The Jacobian matrix of this least squares cost is thus constant. The Hessian matrix H and its inverse are computed off-line. However, the driving features $\tilde{\mathbf{u}}$ are located on the reference image $\mathcal{I}_0$. They must be located on the warped image $\mathcal{I}_W$ for being used in the update. We use our warp reversion process for finding the driving features $\mathbf{u}$ on the warped image *i.e.* , $\mathbf{u}$ such that $\mathcal{W}(\tilde{\mathbf{u}};\mathbf{u}) = \mathbf{u}_0$. An overview of Feature-Driven Inverse Gauss-Newton registration is shown in table A.1.

A.4.2 Learning-Based Local Registration

Learning-based methods model the relationship between the local increment δ and the intensity discrepancy $\mathbf{d}$ with an interaction function f:

$$\delta = f(\mathbf{d}). \tag{A.17}$$

The interaction function is often approximated using a linear model, *i.e.* $f(\mathbf{d}) = \mathsf{F}\mathbf{d}$ where F is called the interaction matrix. This relationship is valid locally around the texture image parameters $\mathbf{u}_0$. Compositional

Off-line

- Evaluate the gradient $\nabla\mathcal{I}_0$ of the reference image
- Evaluate the constant Jacobian K of the warp
- Compute the Jacobian matrix J of the cost function
- Compute the pseudo Hessian $\mathsf{H} = \mathsf{J}^\mathsf{T}\mathsf{J}$ and its inverse H^{-1}

On-line

- Compute the error vector $\mathbf{d} = \boldsymbol{\xi}_{\mathfrak{R}}(\mathcal{I}_0 - \mathcal{I}_W)$
- Compute $\mathbf{b} = \mathsf{J}^\mathsf{T}\mathbf{d}$
- Estimate $\tilde{\mathbf{u}} = \mathbf{u}_0 - \mathsf{H}^{-1}\mathbf{b}$
- Find the driving features $\mathbf{u}$ such that $\mathcal{W}(\tilde{\mathbf{u}};\mathbf{u}) = \mathbf{u}_0$
- Update by threading: $\mathcal{W}(\,\bullet\,;\mathbf{u}_i) \leftarrow \mathcal{W}(\,\bullet\,;\mathbf{u}_i) \circ \mathcal{W}(\,\bullet\,;\mathbf{u})$

Table A.1: Overview of our Feature-Driven Inverse Compositional Gauss-Newton registration.

algorithms are thus required, as in (Jurie and Dhome, 2002) for homographic warps. The Feature-Driven framework naturally extends this approach to non-groupwise warps. However in (Cootes et al., 1998) the assumption is made that the domain where the linear relationship is valid covers the whole set of registrations. They thus apply their interaction function around the current parameters, avoiding the warping and the composition steps. This does not appear to be a valid choice in practice.

The interaction function is learned from artificially perturbed texture images $\mathcal{A}_j$. They are obtained through random perturbations of the reference parameter $\mathbf{u}_0$. In the literature, linear and non linear interaction functions are used. They are learned with different regression algorithms such as Least Squares (LS) (Cootes et al., 1998; Jurie and Dhome, 2002), Support Vector Machines (SVM) or Relevance Vector Machines (RVM) (Agarwal and Triggs, 2006). Details are given below for a linear interaction function, *i.e.* an interaction matrix, learned through Least Squares regression. Table A.2 summarizes the steps of learning-based local registration.

Off-line

- Learn the interaction function f

On-line

- Compute the error vector $\mathbf{d} = \boldsymbol{\xi}_{\mathfrak{R}}(\mathcal{I}_0 - \mathcal{I}_W)$
- Compute $\mathbf{u} = \mathbf{u}_0 + f(\mathbf{d})$
- Update by threading: $\mathcal{W}(\,\bullet\,;\mathbf{u}_i) \leftarrow \mathcal{W}(\,\bullet\,;\mathbf{u}_i) \circ \mathcal{W}(\,\bullet\,;\mathbf{u})$

Table A.2: Overview of our Learning-based registration.

Generating training data with a Feature-Driven Warp. The driving features in the texture image are disturbed from their rest position $\mathbf{u}_0$ with randomly chosen directions $\boldsymbol{\theta}_j$ and magnitudes $\mathbf{r}_j$:

$$\mathbf{u}_j = \mathbf{u}_0 + \boldsymbol{\delta}_j \qquad \text{with } \boldsymbol{\delta}_j = \begin{pmatrix} \mathbf{r}_j \odot \cos(\boldsymbol{\theta}_j) \\ \mathbf{r}_j \odot \sin(\boldsymbol{\theta}_j) \end{pmatrix}, \tag{A.18}$$

where $\cos(\boldsymbol{\theta}_j)$ and $\sin(\boldsymbol{\theta}_j)$ are meant to be applied to all the elements of $\boldsymbol{\theta}_j$ and $\odot$ denotes the element-wise product. The magnitude is clamped between a lower and an upper bound, determining the area of validity of the interaction matrix to be learned. For a Feature-Driven warp, fixing this magnitude is straightforward since the driving features are expressed in pixels. It can be much more complex when the parameters are difficult to interpret such as the usual coefficients of the TPS and the FFD warps. There are two ways to synthesize images:

$$\mathcal{A}_j(\mathbf{q}) \leftarrow \mathcal{I}_0(\mathcal{W}(\mathbf{q}; \mathbf{u}_j)^{\diamond}) \tag{A.19}$$

or

$$\mathcal{A}_j(\mathbf{q}) \leftarrow \mathcal{I}_0(\arg\min_{\mathbf{q}} \|\mathcal{W}(\mathbf{q}; \mathbf{u}_j) - \mathbf{q}\|). \tag{A.20}$$

The former requires warp inversion whereas the latter requires a cost optimization, per-pixel. In our experiments, we use equation (A.19). Our Feature-Driven warp reversion process is thus used to warp the texture image. Training data generation with a Feature-Driven warp is illustrated in figure A.7.

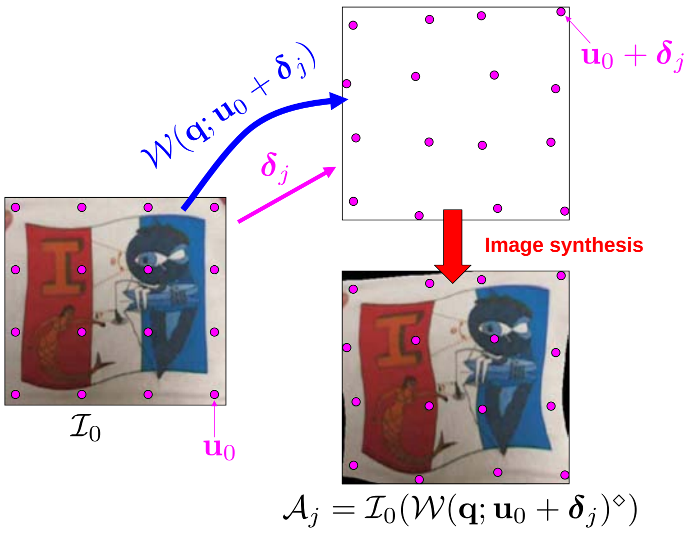

$\mathcal{W}(\mathbf{q}; \mathbf{u}_0 + \boldsymbol{\delta}_j)$
$\boldsymbol{\delta}_j$
$\mathbf{u}_0 + \boldsymbol{\delta}_j$
Image synthesis
$\mathcal{I}_0$
$\mathbf{u}_0$
$\mathcal{A}_j = \mathcal{I}_0(\mathcal{W}(\mathbf{q}; \mathbf{u}_0 + \boldsymbol{\delta}_j)^{\diamond})$

Figure A.7: Generating training data with a Feature-Driven warp.

Learning. The residual vector is computed for the pixels of interest in $\mathfrak{R}$:

$$\mathbf{d}_j = \boldsymbol{\xi}_{\mathfrak{R}} (\mathcal{I}_0 - \mathcal{A}_j). \tag{A.21}$$

The training data are gathered in matrices $\mathsf{D} = (\boldsymbol{\delta}_1 \ \dots \ \boldsymbol{\delta}_m) \in \mathbb{R}^{|\mathfrak{R}| \times m}$ and $\mathsf{L} = (\mathbf{d}_1 \ \dots \ \mathbf{d}_m) \in \mathbb{R}^{|\mathfrak{R}| \times m}$. The interaction matrix $\mathsf{F} \in \mathbb{R}^{|\mathfrak{R}| \times |\mathfrak{R}|}$ is computed by minimizing a Linear Least Squares error in the image space, expressed in pixel value unit, giving:

$$\mathsf{F} = \left(\mathsf{L}\mathsf{D}^{\mathsf{T}} (\mathsf{D}\mathsf{D}^{\mathsf{T}})^{-1} \right)^{\dagger}. \tag{A.22}$$

This is one of the two possibilities for learning the interaction matrix. The other possibility is dual. It minimizes an error in the parameter space, *i.e.* expressed in pixels. The two approaches have been experimentally compared. Learning the interaction matrix in the image space give the best results. Thereafter, we use this option.

A piecewise linear interaction function. Experiments show that a linear approximation of the relationship between the local increment $\boldsymbol{\delta}$ and the intensity discrepancy $\mathbf{d}$, though computationally efficient, does not always give satisfying results. The drawback is that if the interaction matrix covers a large domain of deformation magnitudes, the registration accuracy is spoiled. On the other hand, if the matrix is learned for small deformations only, the convergence basin is dramatically reduced. Using a nonlinear interaction function learned through RVM or SVM partially solves this issue. We use a simple piecewise linear relationship as interaction function. It means that we learn not only one but a series $\mathsf{F}_1, \ldots, \mathsf{F}_\kappa$ of interaction matrices, each of them covering a different range of displacement magnitudes. The interaction function is thus of the form $f(\mathbf{d}) = \sum_{i=1}^{\kappa} a_i \mathsf{F}_i$. More details are given in appendix A.7.

A.5 Feature-Driven Warps

In this section, we specialize the generic Feature-Driven parameterization presented in section A.3.1 for two types of warps: the TPS and the FFD warps. Since the representational power of the TPS warp and of the FFD warp are equivalent (see experiments in section A.6.1), we focused our experiments on the TPS warp. However, it is important to show how the FFD warp can actually be used in the Feature-Driven framework. In particular, we show how the standard FFD model can be extended in order to be compatible with the warp reversion operation.

A.5.1 The Feature-Driven Thin-Plate Spline Warp

A.5.1.1 Definition

Ignoring the parameters, a TPS $\bar{\omega}_{\mathsf{T}}$ is an $\mathbb{R}^2 \to \mathbb{R}$ function. It is the Radial Basis Function that minimizes the integral bending energy. In its natural parameterization, a TPS is driven by a set of $l+3$ weights $\bar{p}_k \in \mathbb{R}$. These weights are grouped in a vector of parameters $\bar{\mathbf{p}} \in \mathbb{R}^{l+3}$. The evaluation of a TPS at the point $\mathbf{q}^\mathsf{T} = (x \ y)$ is given by:

$$\bar{\omega}_{\mathsf{T}}(\mathbf{q}; \bar{\mathbf{p}}) = \sum_{i=1}^{l} \bar{p}_i \rho \left(d^2(\mathbf{q}, \mathbf{c}_i) \right) + \bar{p}_{l+1} x + \bar{p}_{l+2} y + \bar{p}_{l+3}. \tag{A.23}$$

The l 2D points $\mathbf{c}_k$ are called the centers. They are also the driving features in the texture image. They can be located at any place but, in practice, we place them on a regular grid. The function d^2 gives the squared euclidean distance between its two arguments. The function ρ is the TPS basis function and is defined by $\rho(r) = r^2 \log(r)$ for $r > 0$ and $\rho(0) = 0$. In matrix form, equation (A.23) is equivalent to:

$$\bar{\omega}_{\mathsf{T}}(\mathbf{q}; \bar{\mathbf{p}}) = \boldsymbol{\ell}_{\mathbf{q}}^\mathsf{T} \bar{\mathbf{p}}, \tag{A.24}$$

with $\boldsymbol{\ell}_{\mathbf{q}}^\mathsf{T} = \begin{pmatrix} \rho(d^2(\mathbf{q}, \mathbf{c}_1)) & \cdots & \rho(d^2(\mathbf{q}, \mathbf{c}_l)) & \mathbf{q}^\mathsf{T} & 1 \end{pmatrix} \in \mathbb{R}^{l+3}$.

Standard $\mathbb{R}^2 \to \mathbb{R}^2$ TPS warps are obtained by replacing the scalar weights $\bar{p}_i$ by the control points $\mathbf{p}_k = (p_k^x \ p_k^y) \in \mathbb{R}^2$. The control points are grouped in a single matrix of parameters $\mathsf{P} \in \mathbb{R}^{(l+3) \times 2}$ defined by $\mathsf{P}^\mathsf{T} = (\mathbf{p}_1 \ \ldots \ \mathbf{p}_{l+3})$. The TPS warp is thus defined by:

$$\omega_{\mathsf{T}}(\mathbf{q}; \mathbf{p}) = \boldsymbol{\ell}_{\mathbf{q}}^\mathsf{T} \mathsf{P}, \qquad \text{with } \mathbf{p} = \boldsymbol{\nu}(\mathsf{P}). \tag{A.25}$$

A.5.1.2 Feature-Driven Parameterization

The Feature-Driven parameterization of the TPS warp consists in replacing the control points by some features (*i.e.* points) in the current image. A point $\mathbf{a}_k^\mathsf{T} = (a_k^x \; a_k^y) \in \mathbb{R}^2$ is assigned to each center $\mathbf{c}_k$ defined in the texture image. The features $\mathbf{a}_k$ are grouped in a single matrix $\mathsf{A} \in \mathbb{R}^{l\times 2}$. Similarly, the centers $\mathbf{c}_k$ are grouped in a matrix $\mathsf{C} \in \mathbb{R}^{l\times 2}$. Following (Bookstein, 1989), the control points of a TPS can be determined from the correspondences $\mathbf{c}_k \leftrightarrow \mathbf{a}_k$:

$$\omega_\mathsf{T}(\mathbf{c}_k; \mathbf{p}) = \mathbf{a}_k \qquad \forall k \in \{1, \dots, l\}, \tag{A.26}$$

while enforcing the 3 ‘side-conditions’ ensuring that the TPS has square integrable second derivatives (more details can be found in (Wahba, 1990)):

$$\sum_{i=1}^{l} x_i \mathbf{p}_i = \mathbf{0}_{2\times 1}, \quad \sum_{i=1}^{l} y_i \mathbf{p}_i = \mathbf{0}_{2\times 1}, \quad \sum_{i=1}^{l} \mathbf{p}_i = \mathbf{1}_{2\times 1}. \tag{A.27}$$

Combining these $l + 3$ conditions in a single matrix gives the following exactly determined linear system:

$$\mathsf{M}_\lambda \mathsf{P} = \begin{pmatrix} \mathsf{A} \\ \mathbf{0}_{3\times 1} \end{pmatrix}, \tag{A.28}$$

with $\mathsf{M}_\lambda \in \mathbb{R}^{(l+3)\times(l+3)}$ the matrix defined by:

$$\mathsf{M}_\lambda = \begin{pmatrix} \mathsf{N}_\lambda & \mathsf{Q} \\ \mathsf{Q}^\mathsf{T} & \mathbf{0}_{3\times 3} \end{pmatrix}, \tag{A.29}$$

with $\mathsf{N}_\lambda = \mathsf{N} + \lambda \mathsf{I}_l$, $\mathsf{N}^\mathsf{T} = \begin{pmatrix} \boldsymbol{\ell}_{\mathbf{c}_1}^\mathsf{T} & \cdots & \boldsymbol{\ell}_{\mathbf{c}_l}^\mathsf{T} \end{pmatrix}$ and $\mathsf{Q} = \begin{pmatrix} \mathsf{C} & \mathbf{1}_{l\times 1} \end{pmatrix} \in \mathbb{R}^{l\times 3}$. Adding $\lambda \mathsf{I}_l$ to N acts as a regularizer. Determining the control points P from the equation (A.28) can be done in a straightforward manner as the solution of an exactly determined linear system. The resulting matrix of control points, denoted P_λ, is a nonlinear function of the regularization parameter λ and a linear function of the features A:

$$\mathsf{P}_\lambda = \mathsf{M}_\lambda^{-1} \begin{pmatrix} \mathsf{A} \\ \mathbf{0}_{3\times 1} \end{pmatrix}. \tag{A.30}$$

P_λ is a linear ‘back-projection’ of the feature matrix A. It can be computed efficiently using the blockwise matrix inversion formulas:

$$\mathsf{P}_\lambda = \mathsf{E}_\lambda \mathsf{A} \tag{A.31}$$

with:

$$\mathsf{E}_\lambda = \begin{pmatrix} \mathsf{N}_\lambda^{-1} \left(\mathsf{I}_l - \mathsf{Q} \left(\mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \mathsf{Q} \right) \mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \right) \\ \left(\mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \mathsf{Q} \right)^{-1} \mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \end{pmatrix}. \tag{A.32}$$

This expression has the advantages of separating λ and A and introduces units: while P_λ has no obvious unit, A in general has (*e.g.* pixels, meters). Finally, if we replace the natural parameters $\mathbf{p}$ in the definition of the TPS warp ω_T (equation (A.25)) by their expression given in the equation (A.32), we get the Feature-Driven parameterization of the TPS warp, denoted $\mathcal{W}_\mathsf{T}$:

$$\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{a}, \lambda) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E}_\lambda \mathsf{A}, \qquad \text{with } \mathbf{a} = \boldsymbol{\nu}(\mathsf{A}). \tag{A.33}$$

We use the notation $\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{a})$ for $\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{a}, 10^{-4})$. We choose $\lambda = 10^{-4}$ to ensure good numerical conditioning of the matrix N_λ.

Jacobian matrix of the warp. The Jacobian matrix of the warp is needed by the Gauss-Newton based algorithms for local registration (see *e.g.* , section A.2.1 or section A.4.1). We denote K_T the Jacobian matrix of the TPS warp evaluated at the point $\mathbf{q}$. It is defined by $\mathsf{K}_\mathsf{T} = \frac{\partial \mathcal{W}_\mathsf{T}}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \in \mathbb{R}^{2\times 2(l+3)}$ and is given by:

$$\begin{aligned}\mathsf{K}_\mathsf{T} &= \begin{pmatrix} \frac{\partial \mathcal{W}_\mathsf{T}^x}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \\ \frac{\partial \mathcal{W}_\mathsf{T}^y}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathcal{W}_\mathsf{T}^x}{\partial \mathbf{a}^x}(\mathbf{q}; \mathbf{a}) & \mathbf{0}_{1\times(l+3)} \\ \mathbf{0}_{1\times(l+3)} & \frac{\partial \mathcal{W}_\mathsf{T}^y}{\partial \mathbf{a}^y}(\mathbf{q}; \mathbf{a}) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E}_\lambda & \mathbf{0}_{1\times(l+3)} \\ \mathbf{0}_{1\times(l+3)} & \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E}_\lambda \end{pmatrix}, \end{aligned} \tag{A.34}$$

where $\mathcal{W}_\mathsf{T}^x$ and $\mathcal{W}_\mathsf{T}^y$ are the first and the second coordinates of the warp $\mathcal{W}_\mathsf{T}$ and $\mathsf{A} = (\mathbf{a}^x \ \mathbf{a}^y)$.

A.5.2 The Feature-Driven Free-Form Deformation

Tensor-product B-Splines are a particular model of Free-Form Deformations. They are a general model of polynomial functions which have been proved to be useful for image registration (Rueckert et al., 1999). Even if there is a wide variety of B-Splines (with various degrees for the polynomial basis or by choosing exotic knot sequences), we limit our study to the case of the Uniform Cubic B-Splines since it best matches the needs of image registration. For the sake of simplicity, we will abbreviate it FFD.

A.5.2.1 Definition

Monodimensional case. Ignoring the parameters, a monodimensional FFD $\bar{\omega}_\mathsf{F}$ is an $\mathbb{R} \to \mathbb{R}$ function defined as a linear combination of the basis functions N_i weighted by the scalars $\bar{p}_k$ called the weights:

$$\bar{\omega}_\mathsf{F}(x; \bar{\mathbf{p}}) = \sum_{i=1}^{m} \bar{p}_i N_i(x), \tag{A.35}$$

where $\bar{\mathbf{p}} \in \mathbb{R}^m$ is the vector that contains all the weights $\bar{p}_k$. The basis functions are defined using a knot sequence, *i.e.* a non-decreasing sequence $k_1 < \ldots < k_{m+4}$. The FFD is said to be uniform when the knot sequence is uniform, *i.e.* all the knot intervals $[k_i, k_{i+1}]$ have the same length s. In this case, the basis functions N_i are defined by using four polynomials of degree three, the blending functions (see figure A.8(a) for an illustration):

$$N_i(x) = \begin{cases} b_1(x) = \frac{1}{6}\hat{x}^3 \\ \qquad\qquad \text{if } x \in [k_i, k_{i+1}] \\ b_2(x) = \frac{1}{6}\left(-3\hat{x}^3 + 3\hat{x}^2 + 3\hat{x} + 1\right) \\ \qquad\qquad \text{if } x \in [k_{i+1}, k_{i+2}] \\ b_3(x) = \frac{1}{6}\left(3\hat{x}^3 - 6\hat{x}^2 + 4\right) \\ \qquad\qquad \text{if } x \in [k_{i+2}, k_{i+3}] \\ b_4(x) = \frac{1}{6}\left(-\hat{x}^3 + 3\hat{x}^2 - 3\hat{x} + 1\right) \\ \qquad\qquad \text{if } x \in [k_{i+3}, k_{i+4}] \\ 0 \qquad\qquad \text{otherwise} \end{cases} \tag{A.36}$$

where $\hat{x}$ is the normalized abscissa of x defined as $\hat{x} = \frac{x - k_I}{s}$ for $x \in [k_I, k_{I+1}]$.

Domain. We can see from equation (A.35) that an FFD is non-zero only over the interval $[k_1, k_{m+4}]$. However, it is common practice to reduce the domain to $[k_4, k_{m+1}]$. By doing so, there are always exactly 4 non-zero basis functions on each knot interval, as figure A.8(b) illustrates.

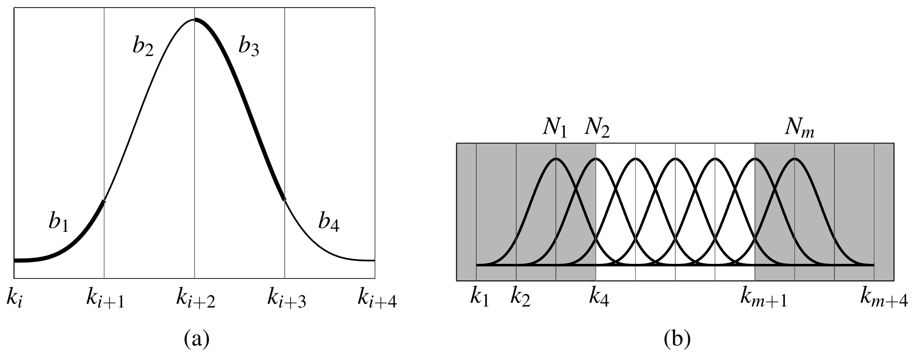


Figure A.8: (a) The basis functions of an FFD are bell-shaped curves with bounded support. They are defined using 4 polynomial pieces of degree three: b_1, b_2, b_3 and b_4. (b) The usual (or natural) definition domain of an FFD is represented by the non-grayed part.

FFD warp. The standard $\mathbb{R}^2 \to \mathbb{R}^2$ FFD warp is obtained as the two-way tensor-product of monodimensional FFDs. Using its natural parameterization, the evaluation of the FFD warp ω_F at the point $\mathbf{q} = (x\ y)^\mathsf{T}$ is given by:

$$\omega_\mathsf{F}(\mathbf{q}; \mathbf{p}) = \sum_{j=1}^{n} \sum_{i=1}^{m} \mathbf{p}_k N_i(x) N_j(y), \quad \text{with } k = (j-1)m + i. \tag{A.37}$$

The mn control points $\mathbf{p}_k$ are grouped in the vector $\mathbf{p} \in \mathbb{R}^{2mn}$ that is defined as $\mathbf{p} = \boldsymbol{\nu}(\mathsf{P})$ where $\mathsf{P} \in \mathbb{R}^{mn \times 2}$ is the matrix given by $\mathsf{P}^\mathsf{T} = (\mathbf{p}_1\ \dots\ \mathbf{p}_{mn})$. The control points of an FFD warp are not more meaningful than the ones of a TPS warp. They are not interpolated: they just act as 'attractors' to the warp. Equation (A.37) can be rewritten in matrix form:

$$\omega_\mathsf{F}(\mathbf{q}; \mathbf{p}) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{P}, \tag{A.38}$$

where $\boldsymbol{\ell}_\mathbf{q} \in \mathbb{R}^{mn}$ is the vector defined by:

$$\boldsymbol{\ell}_\mathbf{q}^\mathsf{T} = (N_1(x)N_1(y)\ \dots\ N_m(x)N_1(y)\ \dots\ N_m(x)N_n(y)). \tag{A.39}$$

A.5.2.2 Feature-Driven Parameterization

The Feature-Driven parameterization of the FFD warp is similar to the one of the TPS warp in the sense that it makes the warp driven by features expressed in pixels in both the texture and the current images. The centers of the TPS warp were used as features in the texture image. Such centers do not exist for FFD warps. We thus introduce a set of points $\mathbf{c}_k$ that will be used as features in the texture image. We call these points *centers* for consistency with the TPS warps. We use $l = mn$ centers located on a regular grid. A feature $\mathbf{a}_k$ in the current image is associated to every center $\mathbf{c}_k$. The control points $\mathbf{p}$ of the FFD warp can be determined from the correspondences $\mathbf{c}_k \leftrightarrow \mathbf{a}_k$ by enforcing the following constraints:

$$\omega_\mathsf{F}(\mathbf{c}_k; \mathbf{p}) = \mathbf{a}_k, \qquad \forall k \in \{1, \dots, l\}. \tag{A.40}$$

Since the number of features is equal to the number of degrees of freedom of the FFD warp, the determination of the parameters from the features can be carried out with an exactly determined linear system:

$$\mathsf{MP} = \mathsf{A}, \tag{A.41}$$

with $\mathsf{M}^\mathsf{T} = \begin{pmatrix} \boldsymbol{\ell}_{\mathbf{c}_1} & \dots & \boldsymbol{\ell}_{\mathbf{c}_l} \end{pmatrix} \in \mathbb{R}^{l \times l}$, $\mathsf{P} = \zeta(\mathbf{p}) \in \mathbb{R}^{l \times 2}$ and $\mathsf{A}^\mathsf{T} = (\mathbf{a}_1 \dots \mathbf{a}_{mn}) \in \mathbb{R}^{2 \times l}$. The solution of the linear system of equation (A.41) can be written $\mathsf{P} = \mathsf{EA}$ where $\mathsf{E} = \mathsf{M}^{-1}$. The existence of the matrix E is guaranteed if the Schoenberg-Whitney conditions are satisfied (see (de Boor, 2001)) as it is the case when the centers are located on a regular grid. Note that the matrix E can be pre-computed. Finally, the Feature-Driven parameterization of the FFD warp, denoted $\mathcal{W}_\mathsf{F}$, is given by replacing the natural parameters $\mathbf{p}$ in equation (A.38) with their expression in function of the features $\mathbf{a} = \boldsymbol{\nu}(\mathsf{A}) \in \mathbb{R}^{2l}$:

$$\mathcal{W}_\mathsf{F}(\mathbf{q}; \mathbf{a}) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{EA}. \tag{A.42}$$

Jacobian matrix of the warp. The Jacobian matrix $\mathsf{K}_\mathsf{F} \in \mathbb{R}^{2 \times 2l}$ for FFD warps can be computed following exactly the same reasoning as for the TPS warp:

$$\begin{aligned} \mathsf{K}_\mathsf{F} = \begin{pmatrix} \frac{\partial \mathcal{W}_\mathsf{F}^x}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \\ \frac{\partial \mathcal{W}_\mathsf{F}^y}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \end{pmatrix} &= \begin{pmatrix} \frac{\partial \mathcal{W}_\mathsf{F}^x}{\partial \mathbf{a}^x}(\mathbf{q}; \mathbf{a}) & \mathbf{0}_{1 \times l} \\ \mathbf{0}_{1 \times l} & \frac{\partial \mathcal{W}_\mathsf{F}^y}{\partial \mathbf{a}^y}(\mathbf{q}; \mathbf{a}) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E} & \mathbf{0}_{1 \times l} \\ \mathbf{0}_{1 \times l} & \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E} \end{pmatrix}. \end{aligned} \tag{A.43}$$

A.5.2.3 Extrapolation

The computations involved in the warp reversion operation (see section A.3.3) can lead to evaluate a warp outside of its natural definition domain. More precisely, in equation (A.11), nothing ensures that the features of the vector $\mathbf{v}$ lies in the domain of the warp. While this is not a problem with the TPS warp whose domain is infinite, extra work need to be done with the FFD warp. Indeed, with the previous definition, it is possible to evaluate an FFD warp outside of its natural domain but it is meaningless since it collapses to 0. In this section, we propose a new method to extrapolate an FFD warp outside of its domain making it virtually infinite.

The principle of the method is simple: a linear extension is added to the basis that crosses the boundaries of the domain (with some extra conditions of continuity and differentiability). While this seems almost trivial in the monodimensional case, it is less simple in two dimensions, *i.e.* for warps. Our strategy consists in defining the extension in 1D and, then, propagate it to the 2D case using the usual tensor-product.

We present the extrapolation approach in the monodimensional case and for the leftmost boundary of the domain (*i.e.* , the knot k_4). The four non-zero bases that cross this boundary are N_1, N_2, N_3 and N_4. Our idea is to drop the part of these bases that are outside the domain and to replace them with a linear extension. We call N_1^e, N_2^e, N_3^e and N_4^e the bases resulting from this process. In addition to be linear, we enforce the following constraints in order to preserve continuity and differentiability:

$$\left. \begin{aligned} N_i^e(k_4) &= N_i(k_4) \\ \frac{\partial N_i^e}{\partial x}(k_4) &= \frac{\partial N_i}{\partial x}(k_4) \end{aligned} \right\} \forall i \in 1, \dots, 4. \tag{A.44}$$

For the sake of simplicity and without loss of generality, we consider that the leftmost boundary coincides with zero ($k_4 = 0$) and that the length of the knot intervals is consistently one ($s = 1$). Under all these constraints, it follows that:

$$N_1^e(x) = \begin{cases} -\dfrac{x}{2} + \dfrac{1}{6} & \text{if } x \in (-\infty, 0] \\ N_1(x) & \text{otherwise} \end{cases}$$

$$N_2^e(x) = \begin{cases} \dfrac{2}{3} & \text{if } x \in (-\infty, 0] \\ N_2(x) & \text{otherwise} \end{cases}$$

$$N_3^e(x) = \begin{cases} \dfrac{x}{2} + \dfrac{1}{6} & \text{if } x \in (-\infty, 0] \\ N_3(x) & \text{otherwise} \end{cases}$$

$$N_4^e(x) = \begin{cases} 0 & \text{if } x \in (-\infty, 0] \\ N_4(x) & \text{otherwise} \end{cases}$$

The extended basis for the rightmost boundary are obtained by symmetry. Figure A.9 illustrates the resulting extended basis functions. The twodimensional counterparts of these newly defined extended basis functions are obtained using the tensor-product.

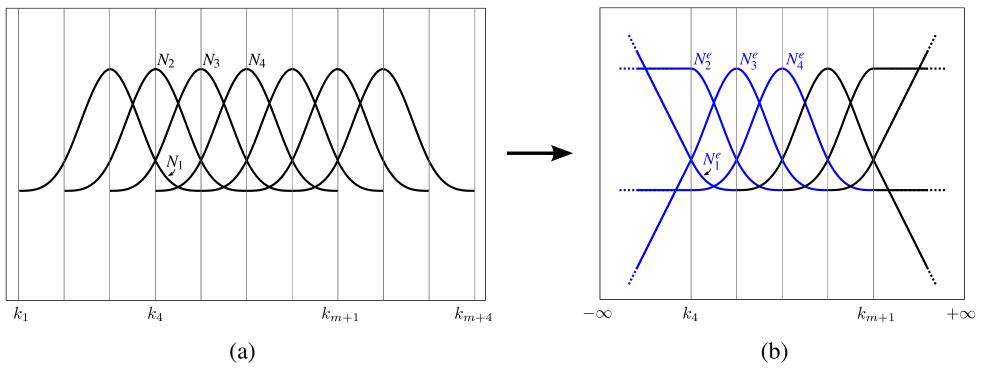


Figure A.9: (a) Standard basis functions. (b) Extended basis functions that allow one to extrapolate outside the natural domain $[k_4, k_{m+1}]$.

The proposed extension gives a remarkably good behavior to the extrapolating functions. See figure A.10 and figure A.11 for an illustration in 1D and 2D respectively. Besides, the fact that the basis functions form a partition of unity remains true ($\sum_{i=1}^4 N_i^e(x) = 1, \forall x \in (-\infty, 0]$).

A.6 Experimental Results

A.6.1 Representational Similarity of the TPS and FFD Warps

This first set of experiments is designed to compare the TPS and the FFD warp. In the light of these experiments, we believe that a fair conclusion is that the TPS and the FFD warp have the same order of representational power. This motivates our choice to only consider the TPS warp in the other experiments of this article.

Fitting error from point correspondences. The experimental setup is as follows. We synthesize a set of point correspondences $\mathbf{q}_k \leftrightarrow \mathbf{a}_k$. The points $\mathbf{q}_k$ in the first image are taken as the nodes of a regular grid of size 11×11 over the domain $[-1, 1] \times [-1, 1]$. These points are randomly and independently moved (with a given average magnitude γ) in order to build the corresponding points $\mathbf{a}_k$ in the second image. A TPS and an FFD warp are then estimated from the point correspondences. The initial data points $\mathbf{a}_k$ and the warped ones

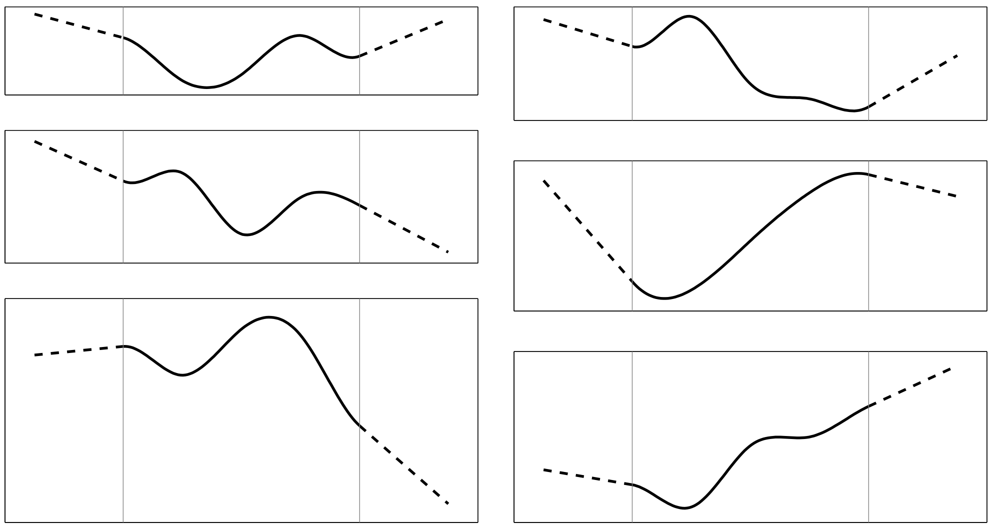


Figure A.10: Examples of our extrapolating FFD in the monodimensional case. The extrapolating parts are represented with dashed lines.

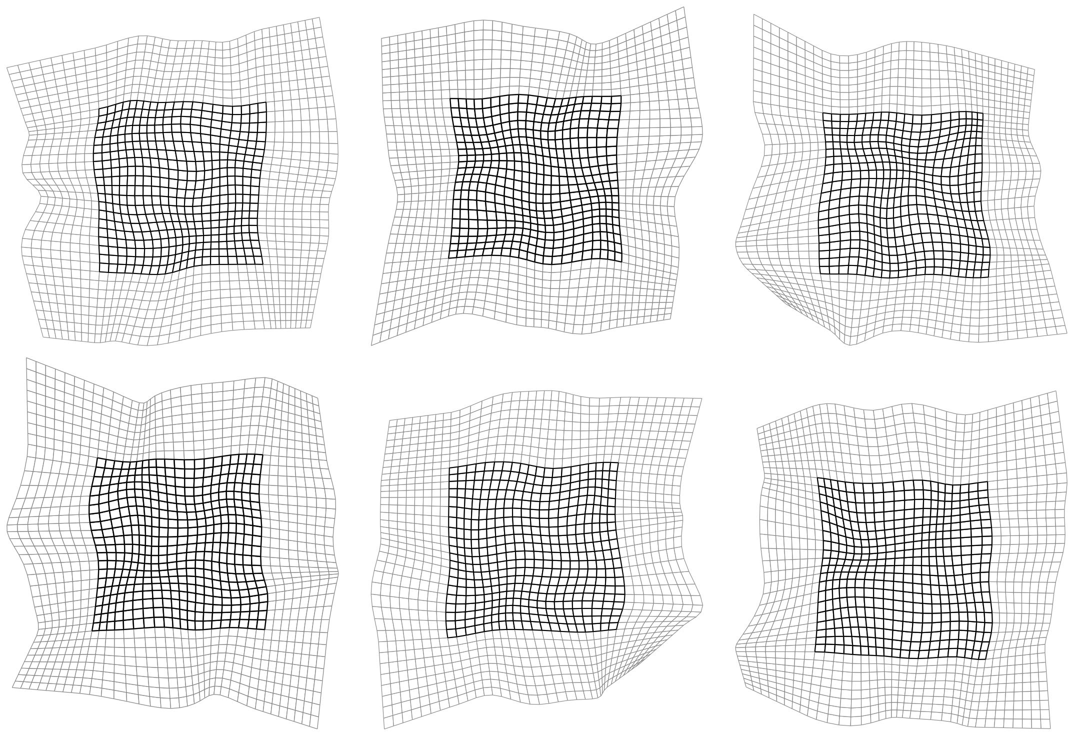
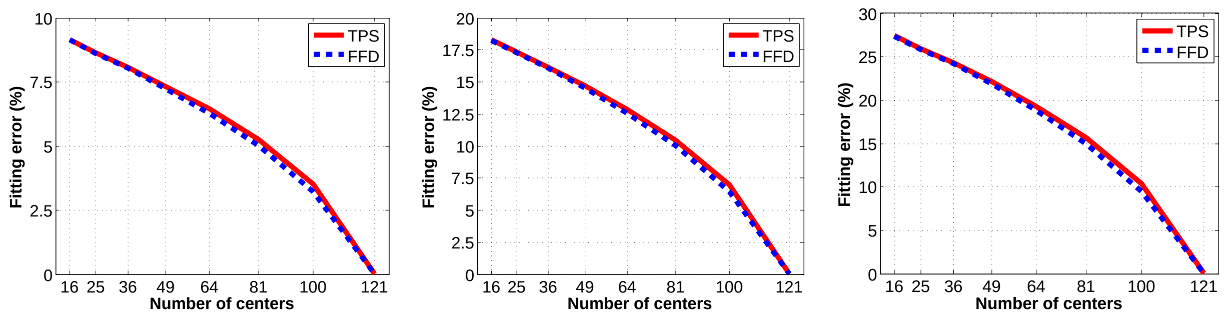


Figure A.11: Examples of our extrapolating FFD warp. The dark part of the meshes represents the warp over its initial domain while the light part is extrapolated.

are compared using the *fitting error* defined by:

$$\frac{1}{n}\sum_{i=1}^{n} d\left(\mathbf{a}_i, \mathcal{W}(\mathbf{q}_i)\right), \tag{A.45}$$

where $\mathcal{W}$ represents the estimated warp (either TPS or FFD) and d the euclidean distance. Note that the fitting error is expressed in the same unit as the points of the second image. The results are shown in figure A.12 for different displacement magnitudes. The reported values are obtained as the average over 500 trials. The fitting errors and the displacement magnitudes are expressed in percentage of the domain size. We can see that the curves in figure A.12 are almost identical. It means that none of the two considered warps prevails the other one ; they can model equally the point correspondences. Note also that the fitting error collapses to zero when the number of centers is the same as the number of point correspondences.



(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure A.12: Fitting error between the synthesized data points and the points warped with the estimated TPS and FFD warps (γ is the average displacement magnitude.)

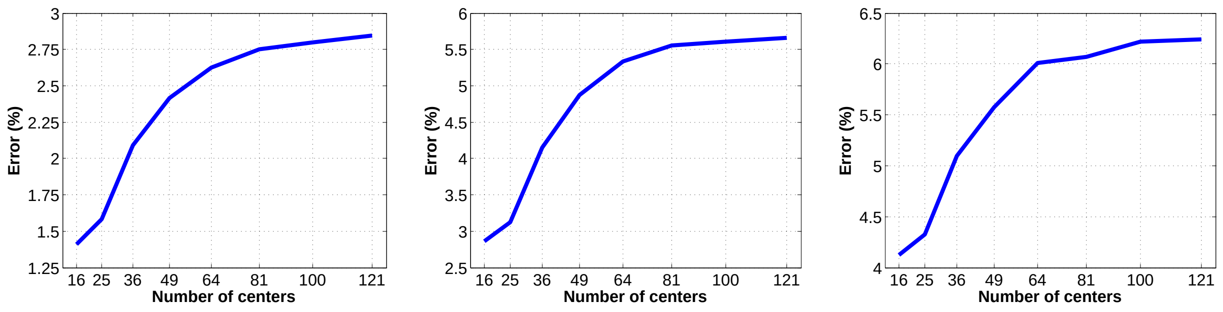
Direct comparison of the warps. This second experiment differs from the first one in the sense that the point correspondences are not generated randomly. First, a TPS warp is generated with randomly determined driving features $\mathbf{v}_{\text{T}}$. The centers of this TPS warp are taken on an $n \times n$ regular grid over the domain $\Omega = [-1, 1] \times [-1, 1]$. The generated driving features $\mathbf{v}_{\text{T}}$ are produced by moving the centers around their initial location with an average magnitude γ. Second, a set of point correspondences is synthesized by sampling the TPS warp: $\mathbf{q}_k \leftrightarrow \mathbf{a}_k = \mathcal{W}_{\text{T}}(\mathbf{q}_k; \mathbf{v}_{\text{T}})$. Third, the features $\mathbf{v}_{\text{F}}$ of an FFD warp are determined from the previously generated point correspondences. Finally, the TPS and the FFD warps are compared using the following measure:

$$\iint_{\Omega} \left\|\mathcal{W}_{\text{T}}(\mathbf{q}; \mathbf{v}_{\text{T}}) - \mathcal{W}_{\text{F}}(\mathbf{q}; \mathbf{v}_{\text{F}})\right\| \mathrm{d}\mathbf{q}. \tag{A.46}$$

The unit of this error measure is the same as the one of the points in the second image. The results are presented in figure A.13. Figure A.14 are the results obtained with the same experimental setup except that the roles of the TPS and the FFD warps have been switched. The reported errors are computed by averaging over 200 trials. These errors are expressed in percentage of the domain size.

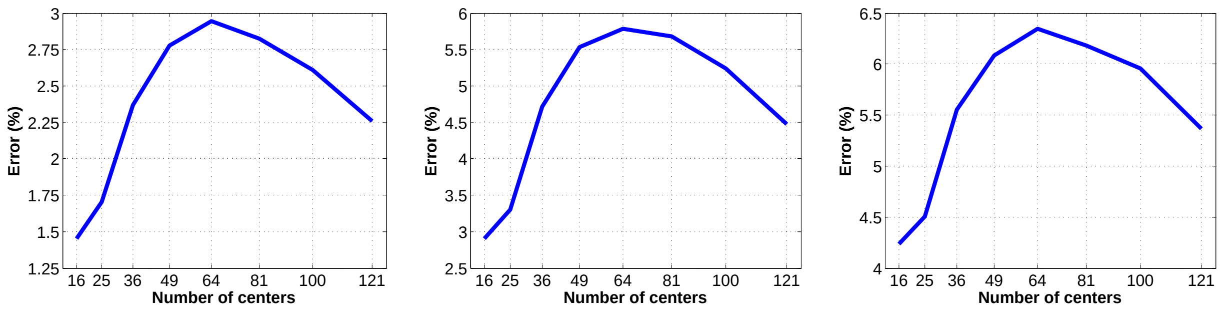
Figure A.13 tells us that given a TPS warp, it is possible to closely approximate the same deformations with an FFD warp. Conversely, the deformations induced by an FFD warp can also be closely approximated by a TPS warp (figure A.14).

Comparison of the TPS and FFD warps induced by the same driving features. Since the parameters in the Feature-Driven framework are meaningful to the warp (*i.e.* they are interpolated by the warps), we can compare



(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

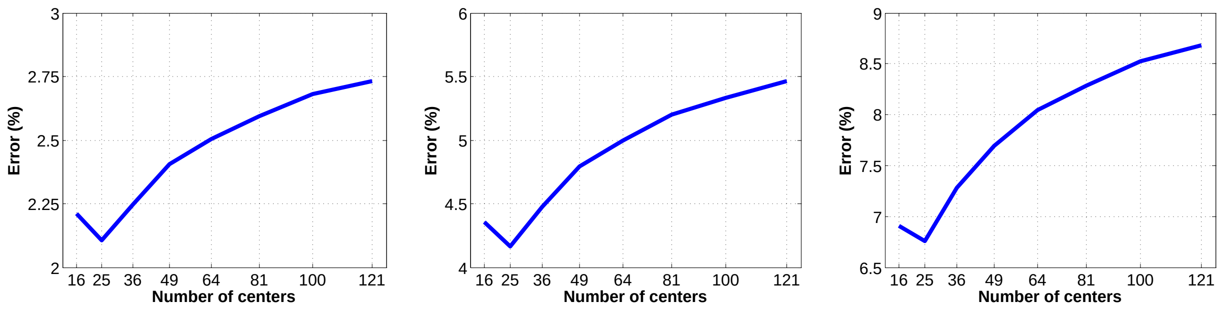
Figure A.13: Error between the FFD and the TPS warps. The FFD warp is estimated from point correspondences that comes from the TPS warp (γ is the displacement magnitude).



(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure A.14: Error between the TPS and the FFD warps. The TPS warp is estimated from point correspondences that come from the FFD warp (γ is the displacement magnitude).

the TPS and the FFD warps with the same set of driving features. In this experiment, we randomly generate some driving features with their associated centers. We then compare the TPS and the FFD warps resulting from these features using the same measure as in the previous experiment. Figure A.15 shows the results for different magnitude of transformation. The reported numbers are obtained as the average over 200 trials.



(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure A.15: Comparison of the TPS and of the FFD warps resulting from a common set of driving features (γ is the displacement magnitude).

The values of the errors and the displacement magnitudes in figure A.15 are expressed in percentage of the domain size. We can see from figure A.15 that TPS and FFD warps induced from the same set of features are close to each other.

A.6.2 Comparison of Registration Algorithms

In this second set of experiments, we compare four algorithms in terms of convergence frequency, accuracy and convergence rate:

- Two classical algorithms:
  - **FA-GN** : the Forward Additive Gauss-Newton approach used by (Bartoli and Zisserman, 2004; Lim and Yang, 2005) and described in section A.2.1.
  - **FA-ESM** : the Efficient Second-order approximation of the cost function proposed by (Benhimane and Malis, 2004) and reviewed in section A.2.1 with an additive update of the parameters[4].
- Two algorithms we propose:
  - **IC-GN** : the Feature-Driven Inverse Compositional registration of section A.3 with Gauss-Newton as local registration engine as described in section A.4.1.
  - **FC-LE** : the Feature-Driven Forward Compositional registration of section A.3, with local registration achieved through learning as described in section A.4.2.

A.6.2.1 Simulated Data

In order to assess algorithms in different controlled conditions, we synthesized images from a texture image. The driving features are placed on a 3×3 grid, randomly perturbed with magnitude r. We add Gaussian noise, with variance $\sigma\%$ of the maximum greylevel value, to the warped image. An example of such generated data is shown in figure A.16. We vary each of these parameters independently, using the following default values: $r = 2$ pixels and $\sigma = 1\%$. The estimated warps are scored by the mean Euclidean distance between the driving features which generated the warped image, and the estimated ones. Convergence to the right solution is declared if this score is lower than one pixel. The characteristics we measured are:

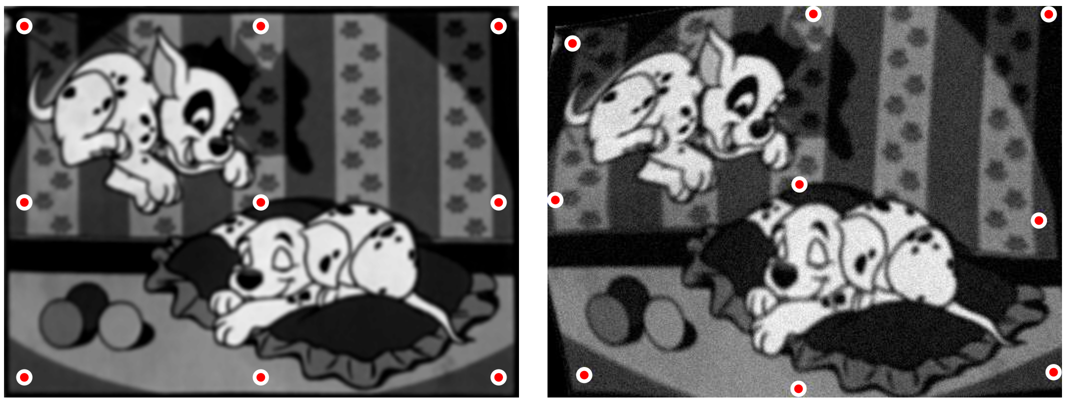
- **Convergence frequency.** This is the percentage of convergence to the right solution.
- **Accuracy.** This is the mean residual error over the trials for which an algorithm converged.
- **Convergence rate.** This is defined by the number of iterations required to converge.

The results are averages over 500 trials. Note that, in this section, the elements in the legend of each figure are ordered according to their performance (from best to worst).

Convergence frequency. The results are shown in figure A.17. FC-LE has the largest convergence basin closely followed by FA-ESM. IC-GN has the smallest convergence basin. At a displacement magnitude of 8 pixels, the convergence frequency of FC-LE is around 75% whereas it is near only 40% for FA-GN and IC-GN. FA-GN has the worst performances against noise. The other algorithms are almost unaffected by noise.

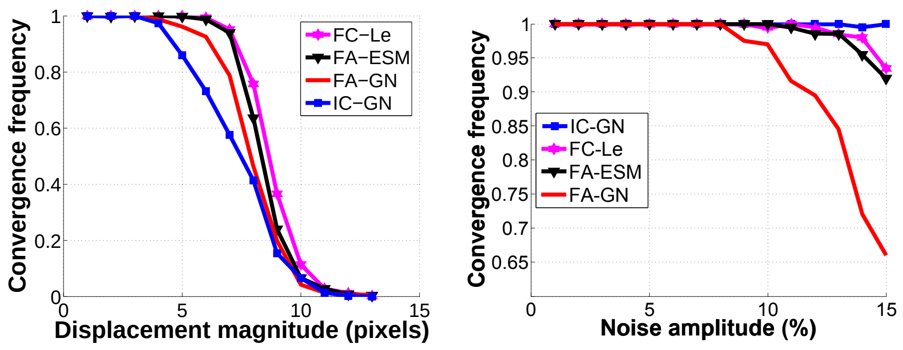
Accuracy. The results are shown in figure A.18. The four algorithms are equivalent against the displacement magnitude. Concerning the amplitude of the noise, IC-GN and FC-LE are equivalent while FA-ESM is slightly worse and FA-GN clearly worse. For example, at 6% noise, the registration errors of IC-GN and FC-LE are around 0.2 pixels, FA-ESM is at about 0.25 pixels and FA-GN at 0.35 pixels.

[4]The original ESM algorithm uses a compositional update. The Feature-Driven framework naturally extends it to non-rigid warps.



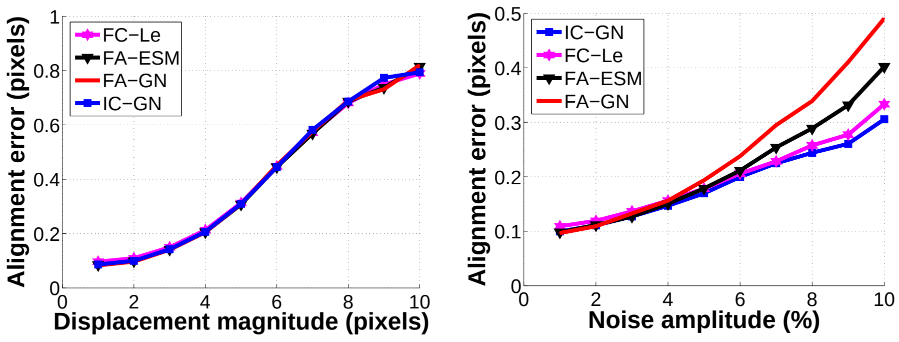
(a) (b)

Figure A.16: An example of simulated data. (a) The texture image. (b) The warped image with gaussian noise added (using $\sigma = 5\%$ and $r = 8$ pixels).



(a) (b)

Figure A.17: Comparison of the four algorithms in terms of convergence frequency against displacement magnitude (a) and noise amplitude (b).



(a) (b)

Figure A.18: Comparison of the four algorithms in terms of accuracy against displacement magnitude (a) and noise amplitude (b).

Convergence rate. The results are shown in figure A.19. The convergence rate of FC-LE and FA-ESM are almost constant against both displacement and noise amplitude. However FC-LE does better, with a convergence rate kept below 10 iterations. FA-GN and IC-GN are efficient for small displacements, *i.e.* below 5 pixels. The convergence rate increases dramatically beyond this value for both of them. FA-GN is also inefficient for noise amplitude over 4%. This is explained by the fact that the FA-GN Jacobian matrix depends mainly on the current image gradient, onto which the noise is added.

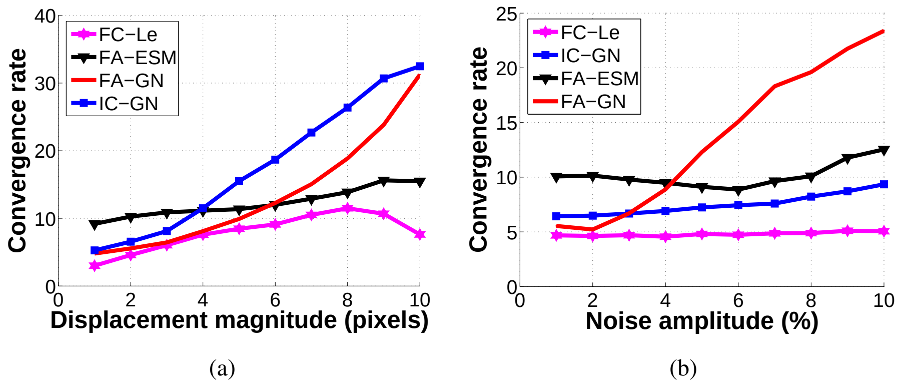


Figure A.19: Comparison of the four algorithms in terms of convergence rate against displacement magnitude (a) and noise amplitude (b).

Discussion. This set of experiments on synthetic data shows that the proposed algorithms, *i.e.* FC-LE and IC-GN, are always the best ones in the presence of noise. FC-LE also obtains the best performances against the displacement magnitude. However, the standard algorithms, *i.e.* FA-ESM and FA-GN, performs better than IC-GN for important displacement magnitudes.

A.6.2.2 Real Data

The four above described algorithms have been compared on several image sequences. We show results for four such videos[5]. We measured the average and maximum intensity RMS along the sequence, the total number of iterations and the computational time (expressed in seconds). The RMS is expressed in pixel value unit. Note that the RMS is computed on the pixels of interest, *i.e.* the pixels actually used in the registration algorithms themselves. All algorithms have been implemented in Matlab. In order to illustrate the registration, we defined a mesh on the texture image and transferred it to all the other frames. Note that these meshes are different from the estimated driving features. The registration differences between the four algorithms are generally visually indistinguishable when they converge.

The first T-shirt sequence. This sequence has 400 frames. The displacement magnitude between the frames may be important. The driving features of the warp are placed on a 3×3 grid. Results are given in table A.3 and registration for the FC-LE algorithm shown in figure A.20. FC-LE performs well on this sequence. It is the fastest and the most accurate. FA-GN, FA-ESM and IC-GN are quite equivalent in terms of alignment accuracy. FA-GN needs a lot of iterations, making it 5 times slower than FC-LE.

The paper sequence. This sequence has 350 frames. The driving features of the warp are placed on a 4×4 grid. The results are given in table A.4 and registration for the FC-LE algorithm shown in figure A.21. IC-GN

[5]These videos can be downloaded at `http://www.florentbrunet.com/ijcv2010`.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	8.70/13.67	9057	2083/5.2
FA-ESM	9.23/14.77	3658	877/2.2
IC-GN	9.69/15.82	6231	436/1.1
FC-LE	**6.66/12.87**	**3309**	**380/0.95**

Table A.3: Results for the first T-shirt sequence. Bold indicates best performances.

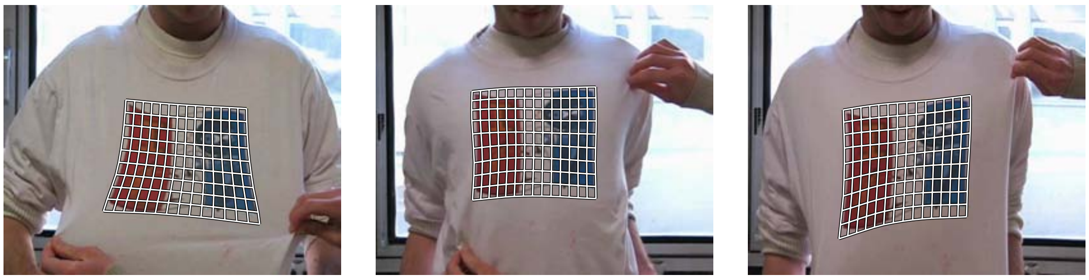


Figure A.20: Registration results for FC-LE on the first T-shirt sequence.

diverges when the deformation seems to be the most important. The other algorithms have similar alignment performances, FA-GN being slightly better. FC-LE is however 3 times faster than the other algorithms.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	**8.98/17.57**	2422	532/1.5
FA-ESM	10.22/20.49	2473	560/1.6
FC-LE	9.44/19.4	**1330**	**176/0.5**

Table A.4: Results for the paper sequence. The IC-GN algorithm diverges on this sequence. Bold indicates best performances.

The rug sequence. This short sequence has 42 frames. The displacement magnitude is high. The driving features of the warp are placed on a 5×5 grid. The results are given in table A.5 and registration for the FC-LE algorithm shown in figure A.22. As for the paper sequence, IC-GN diverges. FA-GN and FA-ESM give the most accurate alignment, FC-LE being slightly worse. On the other hand it is 7 times faster.

The second T-shirt sequence. This sequence has 623 frames. Deformations are moderate, but there are strong global illumination variations along the sequence. The driving features of the warp are placed on a 3×3 grid. The results are given in table A.6 and registration results for the FC-LE algorithm shown in figure A.23. The registration is well achieved by the four algorithms. The small residual error shows that the global illumination changes are correctly compensated. FC-LE and IC-GN are respectively 4 and 2 times faster than the classical algorithms.

Discussion. FA-GN is the most accurate algorithm. It is however inefficient, especially for important displacements. FA-ESM has almost similar performances compared to the FA-GN while being slightly more efficient. IC-GN is efficient, but looses effectiveness for high displacements. As for the experiments with synthetic data, FC-LE has the best behavior: it is similar to FA-GN for accuracy while being 5 times faster on average and is equivalent or better than IC-GN and FA-ESM in terms of alignment accuracy, computational cost and has a larger convergence basin.

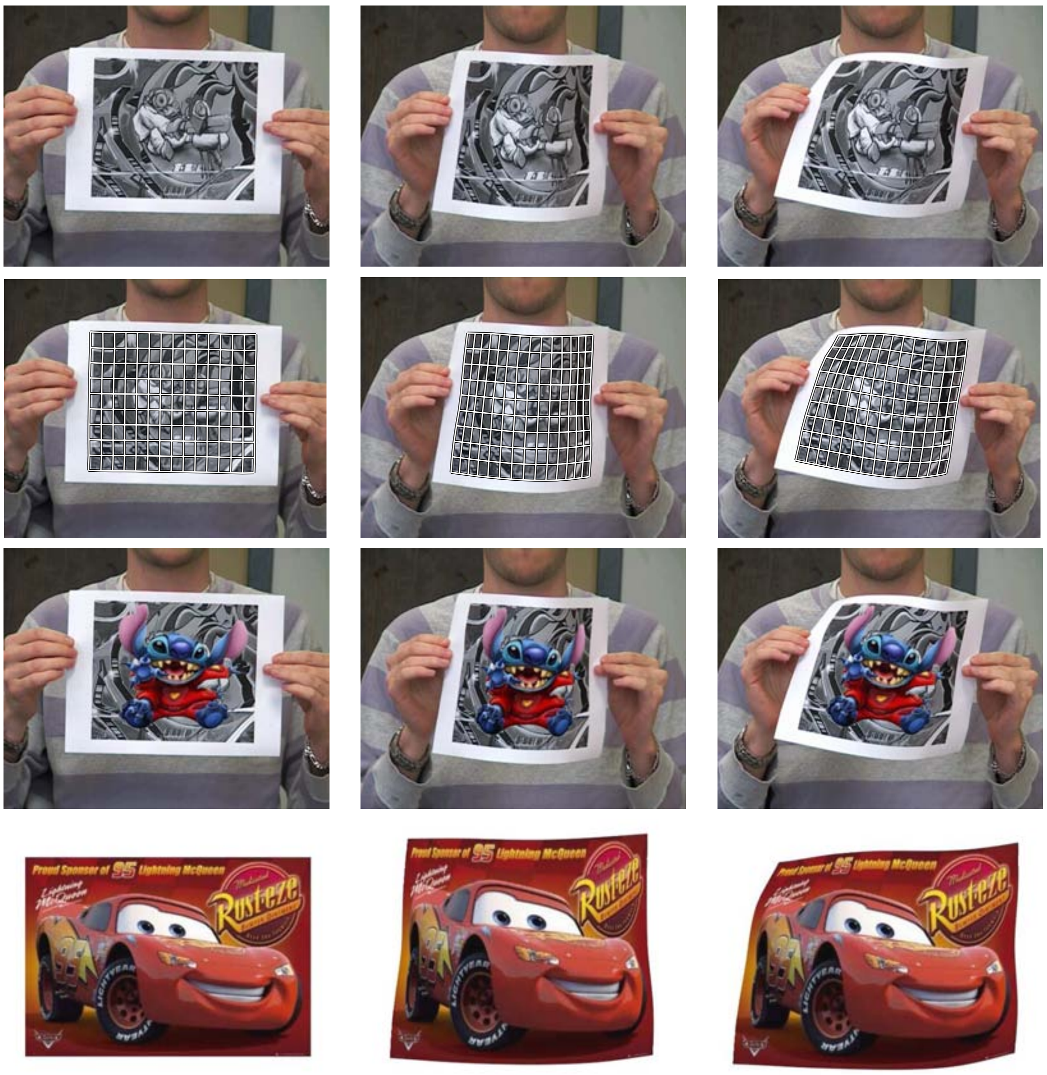


Figure A.21: First row: the paper sequence. Second row: registration results for FC-LE. Third row: surface augmentation with the Walt Disney character *Stitch*. Fourth row: deformations are captured and applied on a poster of the Walt Disney movie *Cars*.

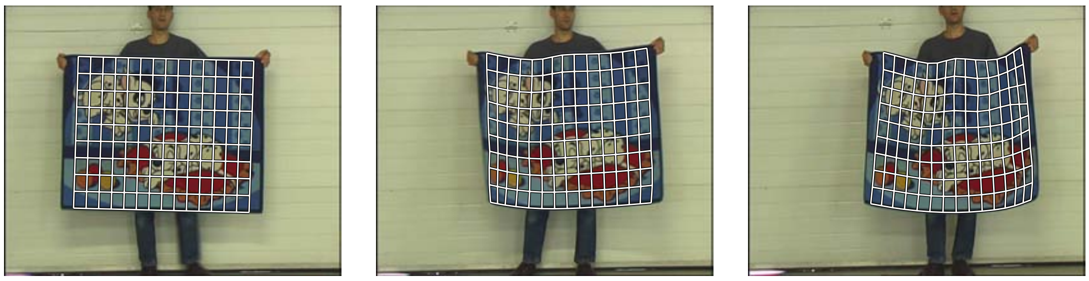


Figure A.22: Registration results for FC-LE on the rug sequence.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	**5.64/8.21**	538	118/2.8
FA-ESM	5.79/8.63	477	109/2.6
FC-LE	6.45/9.80	**149**	**17.13/0.4**

Table A.5: Results for the rug sequence. The IC-GN algorithm diverges on this sequence. Bold indicates best performances.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	**4.51/7.42**	3408	785/1.25
FA-ESM	4.53/7.49	3268	788/1.25
IC-GN	4.87/7.61	4407	381/0.61
FC-LE	4.61/7.70	**1757**	**247/0.39**

Table A.6: Results for the second T-shirt sequence. Bold indicates best performances.

A.7 Conclusions

We addressed an important issue for the problem of non-rigid registration. We proposed the Feature-Driven framework, relaxing the groupwise requirement for using efficient compositional algorithms such Inverse Compositional and Learning-Based algorithms. We also explained in details the Feature-Driven parameterization for the TPS and the FFD warps. Experiments show that Feature-Driven algorithms are more efficient compared to classical ones with additive update of the parameters. Overall, the best algorithm is the combination of the Feature-Driven framework, the Forward Compositional update of the parameters and the local registration based on Learning. The proposed algorithms make foreseeable accurate real-time surface registration.

Acknowledgements. We would like to thank Selim Benhimane for his useful advice, in particular for proposing the Gaussian Mixture Model in order to select the weights of the interaction matrices in the piecewise linear model used in the learning-based local registration.

Appendix: Learning-based Registration with a Piecewise Linear Model

A.7.1 Framework

Learning-based registration algorithms such as (Cootes et al., 1998; Jurie and Dhome, 2002) use a linear model *i.e.* a single interaction matrix. It has several drawbacks: if this interaction matrix covers a large domain of deformation magnitudes, registration accuracy is spoiled. On the other hand, if the matrix is learned for small deformations only, the convergence basin is dramatically reduced. We propose to learn a series $\mathsf{F}_1, \ldots, \mathsf{F}_\kappa$ of interaction matrices, each of them covering a different range of displacement magnitudes. This forms a piece-

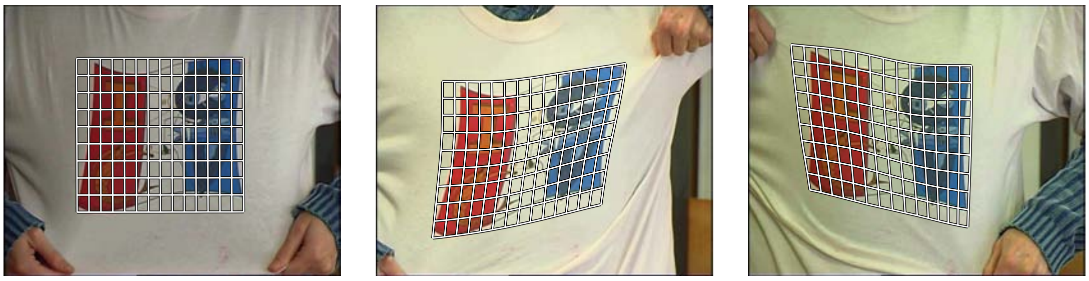


Figure A.23: Registration results for FC-LE on the second T-shirt sequence.

wise linear approximation to the true relationship. Experiments show that using a piecewise linear relationship solves these issues. We propose several ways to combine the interaction matrices $\{\mathsf{F}_i\}_{i=1}^{\kappa}$, that yield different piecewise linear relationships.

Trivial combination. One possibility is to apply all the matrices in turn (algorithm LOOP). The interaction matrix F_1 is first applied until convergence. Then, the other matrices $\{\mathsf{F}_i\}_{i=2}^{\kappa}$ are used one after the other. The last matrix F_κ, learned on the smallest displacement, ensures accuracy. The drawback is that all the matrices are used even for small displacements. It yields a dramatically high number of iterations to converge. Another approach is to try all the linear relationships at each iteration. The one resulting in the smallest residual error is kept (algorithm BEST). These piecewise linear relationships appear not to be the most discerning choice since they are not efficient. In fact, LOOP implies a large convergence rate whereas BEST yields high computational cost per iteration. They are less efficient when the number of interaction matrices κ increases.

Statistical map selection. Our goal is to select the most appropriate interaction matrix at each iteration (algorithm PROB). Each of those indeed has a specific domain of validity in the displacement magnitude. This can unfortunately not be determined prior to image registration. We thus propose to learn a relationship between the intensity error magnitude and displacement magnitude intervals. We express this relationship in terms of probabilities. The intensity error magnitude for a residual vector $\mathbf{d}$ is defined as its RMS: $e(\mathbf{d}) = \mathrm{rms}(\mathbf{d})$. Experimentally, we observed that $P(\mathsf{F}_i|e(\mathbf{d}))$ closely follows a Gaussian distribution, see figure A.24.

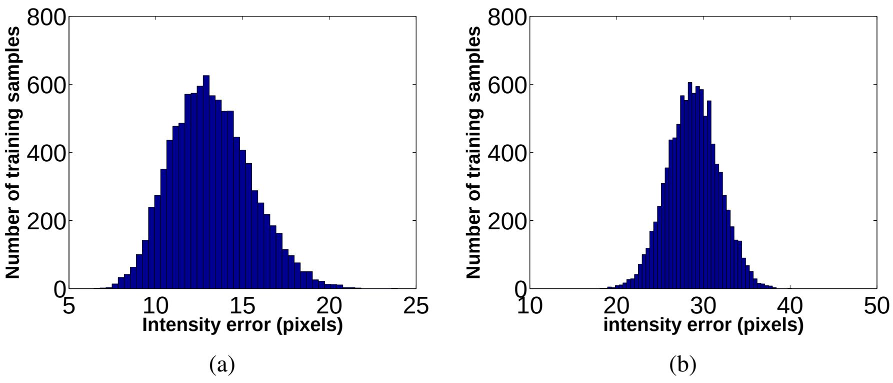


Figure A.24: Distribution of the intensity error magnitude for two different perturbation intervals: (a) $[2 \ldots 5]$ and (b) $[7 \ldots 13]$ pixels, respectively.

Given the current intensity error $e(\mathbf{d})$, finding the most appropriate interaction matrix F_s is simply achieved by solving:

$$s = \arg\max_t P(\mathsf{F}_t|e(\mathbf{d})). \tag{A.47}$$

Mixture models. We use an interaction matrix given by:

$$\mathsf{F} \leftarrow \sum_{i=1}^{\kappa} a_i \mathsf{F}_i \qquad \text{with} \sum_{i=1}^{\kappa} a_i = 1, \tag{A.48}$$

where $\{a_i\}_{i=1}^{\kappa}$ are the mixture proportions.

We compare two mixture models with different probability distributions:

- a Gaussian Mixture Model (GMM): $a_i = \frac{P(\mathsf{F}_i|e(\mathbf{d}))}{\sum_{k=1}^{\kappa} P(\mathsf{F}_k|e(\mathbf{d}))}$, where $P(\mathsf{F}_i|e(\mathbf{d}))$ follows a Gaussian distribution;

- a Constant Mixture Model (CMM): $a_i = \frac{1}{\kappa}$.

For all the piecewise linear relationships described above[6], the interaction matrix F_κ is applied for the last two iterations. This additional step ensures accuracy.

A.7.2 Experimental Results

We use a simple homographic warp to guarantee a fair comparison[7] between the five piecewise linear relationships. The setup described in section A.6.2.1 is used.

Convergence frequency. The results are shown in figure A.25. The five piecewise linear relationships have similar convergence basins. Their convergence frequency are over 95% and around 65% at a displacement magnitude of 20 pixels and 25 pixels respectively. CMM seems to have a slightly thiner convergence basin. GMM and PROB are quite sensitive to noise. It corrupts the probability distributions learned off-line. GMM is slightly better than PROB: at a noise magnitude of 7% the convergence frequency of PROB is only 80% whereas GMM always converges. CMM, BEST and LOOP are insensitive to noise.

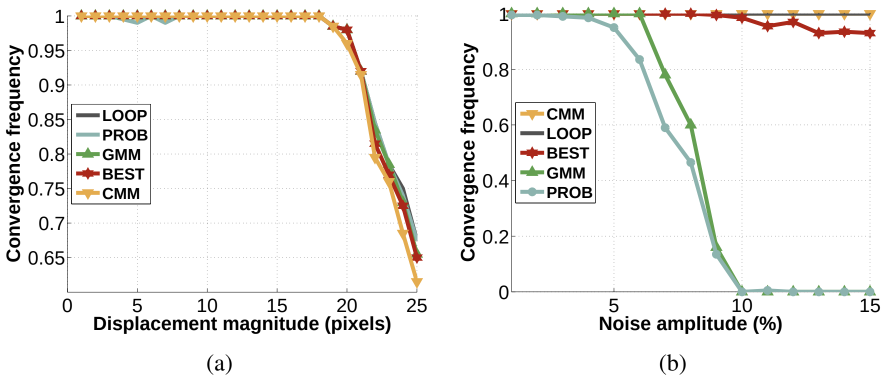


Figure A.25: Comparison of the five piecewise linear relationships in terms of convergence frequency against displacement magnitude (a) and noise amplitude (b). On graph (b), CMM and LOOP are indistinguishable.

Accuracy. The results against displacement magnitude are similar for the five piecewise linear relationships. The associated graph is thus not shown. The residual error is around 0.025 pixels for all tested magnitudes. GMM and PROB are less accurate against noise beyond an amplitude of 7% and 9% respectively than the other piecewise linear relationships. This is illustrated in figure A.26(b). Their residual errors are approximately 0.7 pixels against 0.3 pixels for CMM, BEST and LOOP (*i.e.* , a 2.5 ratio).

Convergence rate. The results are shown in figure A.26(a). BEST and LOOP are clearly not efficient against displacement magnitude: LOOP needs at least 30 iterations to converge while BEST requires 10 iterations with high computational cost. CMM, GMM and PROB do better with a convergence rate kept below 10. Convergence rates for the five piecewise linear relationships are similar against noise magnitude. The associated graph is thus not shown.

Discussion. Overall, CMM is the best piecewise relationship. It is much more efficient than LOOP and BEST and unaffected by noise contrary to CMM and PROB. Its convergence basin is only slightly smaller than those

[6]Except LOOP which naturally includes this additional step.
[7]The compositional update of the warp is not approximated since an homography belongs to a group.

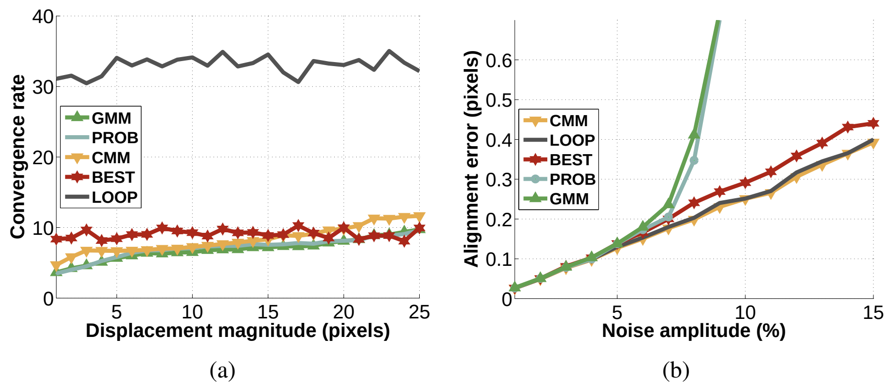


Figure A.26: (a) Comparison of the five piecewise linear relationships in terms of convergence rate against displacement magnitude. (b) Comparison of the five piecewise linear relationships in terms of accuracy against noise amplitude.

induced by the other relationships. We use the CMM piecewise linear relationship in the experiments of section A.6.2.

Bibliography

A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1), January 2006.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19: 716–723, 1974.

D. Allen. Mean square error of prediction as a criterion for selecting variables. *Technometrics*, 13:469–475, 1971.

J. Aloimonos. Shape from texture. *Biological Cybernetics*, 58:345–360, 1988.

S. Aouadi and L. Sarry. Accurate and precise 2D-3D registration based on X-ray intensity. *Computer Vision and Image Understanding*, 110:134–151, 2008.

S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4: 40–79, 2010.

K. V. Arya, P. Gupta, P. K. Kalra, and P. Mitra. Image registration using robust M-estimators. *Pattern Recognition Letters*, 28:1957–1968, 2007. ISSN 0167-8655.

M. Avriel and D.J. Wilde. Optimality proof for the symmetric fibonacci search technique. *Fibonacci Quarterly*, 4:265–269, 1966.

S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, 2004.

B. Bartczak, I. Schiller, C. Beder, and R. Koch. Integration of a Time-of-Flight camera into a mixed reality system for handling dynamic scenes, moving viewpoints and occlusions in real-time. *3D Data Processing, Visualization and Transmission*, 2008.

A. Bartoli. Groupwise geometric and photometric direct image registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2098–2108, December 2008a.

A. Bartoli. Maximizing the predictivity of smooth deformable image warps through cross-validation. *Journal of Mathematical Imaging and Vision*, 31(2-3):133–145, 2008b.

A. Bartoli. *Contributions to Image Registration and to the 3D Reconstruction of Rigid and Deformable Scenes (Habilitation à diriger des recherches)*. PhD thesis, LASMEA UMR6602 - Université Blaise Pascal (Clermont 2), 2008c.

A. Bartoli and P. Sturm. The 3D line motion matrix and alignment of line reconstruction. *International Journal of Computer Vision*, 15:501–517, 2004.

A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation and bundle adjustment. *Computer Vision and Image Understanding*, 100:416–441, 2005.

A. Bartoli and A. Zisserman. Direct estimation of non-rigid registrations. *British Machine Vision Conference*, 2004.

A. Bartoli, V. Gay-Bellile, U. Castellani, J. Peyras, S.I. Olsen, and P. Sayd. Coarse-to-fine low-rank structure-from-motion. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

A. Bartoli, M. Perriollat, and S. Chambon. Generalized thin-plate spline warps. *International Journal of Computer Vision*, 88:85–110, 2010.

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *European Conference on Computer Vision*, 2006.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110:346–359, 2008.

S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. *Proceedings of the International Conference on Intelligent Robots and Systems*, 2004.

Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989. ISSN 0162-8828.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1222–1239, 2001.

J. De Brabanter, K. Pelckmans, J. Suykens, J. Vandewalle, and B. De Moor. Robust cross-validation score functions with application to weighted least squares support vector machine function estimation. Technical report, Katholieke Universiteit Leuven, 2003.

M. Brand. A direct method for 3D factorization of nonrigid motion observed in 2D. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2690–2696, 2000.

F. Brunet. Convolutions binomiales et dérivation de fonctions discrètes bruitées. Master's thesis, Université Blaise Pascal, 2007.

F. Brunet, A. Bartoli, R. Malgouyres, and N. Navab. L-Tangent Norm: A low computational cost criterion for choosing regularization weights and its use for range surface reconstruction. *3D Data Processing, Visualization and Transmission*, 2008.

F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Ajustement automatique de surfaces paramétriques sur données de profondeur en présence d'un bruit hétérogène. *Actes des journées COmpression et REprésentation des Signaux Audiovisuels*, Mars 2009a.

F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Nurbs warps. *British Machine Vision Conference*, 2009b.

F. Brunet, A. Bartoli, J. Peyras, N. Navab, and R. Malgouyres. Découverte automatique de la région d'intérêt en recalage d'images direct. *Actes du onzime congrs francophone des jeunes chercheurs en vision par ordinateur*, 2009c.

F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Utilisation de l'information photométrique pour la sélection des hyperparamètres en recalage géométrique d'images. *Compression et reprsentation des signaux audiovisuels*, 2010a.

F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Pixel-based hyperparameter selection for feature-based image registration. *Vision, Modeling and Visualization Workshop*, 2010b.

F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Dcouverte automatique du recouvrement en recalage direct d'images. *Reconnaissance des Formes et Intelligence Artificielle*, 2010c.

F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Direct image registration without region of interest. *Vision, Modeling and Visualization Workshop*, 2010d.

F. Brunet, R. Hartley, A. Bartoli, N. Navab, and R. Malgouyres. Monocular template-based reconstruction of smooth and inextensible surfaces. *Proceedings of the Tenth Asian Conference on Computer Vision (ACCV)*, Queenstown (New Zealand), November 2010e.

R. Burachik, L. Mauricio, G. Drummond, A.N. Iusem, and E.D. Castorina. Full convergence of the steepest descent method with inexact line searches. *Optimization*, 32:137–146, 1996.

K. Burnham and D. Anderson. Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods Research*, 33(2):261–304, 2004.

N. Carlson. NURBS surface fitting with Gauss-Newton. Master's thesis, Luleå University of Technology, 2009.

A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus de Académie des Sciences*, 25:536–538, 1847.

CESAR laboratory (Oak Ridge National Laboratory). USF Range Image Database. `http://marathon.csee.usf.edu/range/DataBase.html`.

M. Cetin and A. Erar. Variable selection with Akaike information criteria: a comparative study. *Hacettepe Journal of Mathematics and Statistics*, 31:89–97, 2002.

G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the Association for Computing Machinery*, 13:547–569, 1966.

G. Charpiat, O. Faugeras, and R. Keriven. Image statistics based on diffeomorphic matching. *International Conference on Computer Vision*, 2005.

A. Collignon, D. Vandermeulen, P. Suetens, and G. Marchal. 3d multi-modality medical image registration based on information theory. *Computational Imaging and Vision*, 3:263–274, 1995.

T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *European Conference on Computer Vision*, 1998.

T. Corpetti, É. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:365–380, 2002.

P. Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

J.-C. Culioli. *Introduction l'optimisation*. Ellipses Marketing, 1994.

H. B. Curry and I. J. Schoenberg. On spline distributions and their limits: the pólya distributions. *Bulletin of the American Mathematical Society*, 53:1114, 1947.

C. Dachapak, S. Kanae, Z. J. Yang, and K. Wada. Study on radial basis function network in reproducing kernel Hilbert space. *Signal and Image Processing*, 2005.

T. A. Davis and W. W. Hager. Modifying a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20:606–627, 1999.

C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6:50–62, 1972.

C. de Boor. *A Practical Guide to Splines, Revised Edition*, volume 27 of *Applied Mathematical Sciences*. Springer, 2001.

A. Del Bue. A factorization approach to structure from motion with shape priors. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

A. Del Bue and L. Agapito. Non-rigid stereo factorization. *International Journal of Computer Vision*, 66(2): 193–207, 2006.

A. Del Bue and L. Agapito. *Scene Reconstruction, Pose Estimation and Tracking*, chapter 14, pages 243–264. 2007.

P. Dierckx. *Curve and Surface Fitting with Splines*. Monographs on Numerical Analysis. Oxford University Press, 1993.

G Donato and S Belongie. Approximation methods for thin plate spline mappings and principal warps. *European Conference on Computer Vision*. Springer, 2002.

A. Doshi, A. Hilton, and J. Starck. An empirical study of non-rigid surface feature matching. *European Conference on Visual Media Production*, 2008.

J. Erickson, S. Har-Peled, and D. Mount. On the least median square problem. *20th ACM Symposium on Computational Geometry*, pages 273–279, 2004.

D. Falie and V. Buzuloiu. Noise characteristics of 3D Time-of-Flight cameras. *International Symposium on Signals, Circuits and Systems*, 2007.

M. Farenzena, A. Bartoli, and Y. Mezouar. Efficient camera smoothing in sequential structure-from-motion using approximate cross-validation. *European Conference on Computer Vision*, 2008.

G. Farin. *Curves and surfaces for computer-aided geometric design 4th Edition*. Computer Science and Scientific Computing. Academic Press, 1997.

O. Faugeras. *Three-Dimensional Computer Vision*. MIT, 1993.

O. Faugeras, Q.-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2004.

M. Fischler and B. Robert. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

G. Fleury and M. Gourgand. Genetic algorithms applied to workshop problems. *International Journal of Computer Integrated Manufacturing*, 11:183–192, 1998.

M. Fornefett, K. Rohr, and H. Stiehl. Elastic registration of medical images using radial basis functions with compact support. *IEEE International Conference on Computer Vision and Pattern Recognition*, 1999.

M. Fornefett, K. Rohr, and H.S. Stiehl. Radial basis functions with compact support for elastic registration of medical images. *Image and Vision Computing*, 19:87–96, 2001.

V. Gay-Bellile. *Contributions au recalage et à la reconstruction 3D de surfaces déformables*. PhD thesis, Université Blaise Pascal, 2008.

V. Gay-Bellile, M. Perriollat, A. Bartoli, and P. Sayd. Image registration by combining thin-plate splines with a 3D morphable model. *IEEE International Conference on Image Processing*, 2006.

V. Gay-Bellile, A. Bartoli, and P. Sayd. Feature-driven direct non-rigid image registration. *British Machine Vision Conference*, 2007.

S. Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70:320–328, 1975.

C. P. Gendrich and M. M. Koochesfahani. A spatial correlation technique for estimating velocity fields using molecular tagging velocimetry (mtv). *Experiments in Fluids*, 22:67–77, 1996.

J.E. Gentle, W. Härdle, and Y. Mori. *Handbook of Computational Statistics*. Springer, 2004. URL `http://fedc.wiwi.hu-berlin.de/xplore/ebooks/html/csa/`.

S. George and M. T. Nair. Parameter choice by discrepancy principles for ill-posed problems leading to optimal convergence rates. *Journal of Optimization theory and applications*, 83(1):217–222, 1994.

P. Georgel, S. Benhimane, and N. Nassir. A unified approach combining photometric and geometric information for pose estimation. *British Machine Vision Conference*, 2008.

S. B. Gokturk, H. Yalcin, and C. Bamji. A time-of-flight depth sensor - system description, issues and solutions. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.

G. Golub, M. Health, and G. Wahba. Generalized cross validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

G.H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996a.

G.H. Golub and C. F. Van Loan. *Matrix Computations*, chapter 5.2, pages 223–236. The Johns Hopkins University Press, 1996b.

A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*, chapter The Gaussian and Mean Curvatures, pages 373–380. CRC Press, 1997.

M. Groher, M. Baust, D. Zikic, and N. Navab. Monocular deformable model-to-image registration of vascular structures. *Proceedings of the International Workshop on Biomedical Image Registration (WBIR)*, 2010.

N. Gumerov, A. Zandifar, R. Duraiswami, and L. S. Davis. Structure of applicable surfaces from single views. *European Conference on Computer Vision*, 2004.

E. Haber and J. Modersitzki. Intensity gradient based registration and fusion of multi-modal images. Springer Berlin, editor, *International Conference on Medical Image Computing and Computer Assisted Intervention*, volume 4191 of *Lecture Notes in Computer Science*, pages 726–733, 2006.

D. Hahn, V. Daum, and J. Hornegger. Automatic parameter selection for multi-modal image registration. *IEEE Transactions on Medical Imaging*, 29(5):1140–1155, 2010.

D. A. Hahn, Y. Sun, J. Hornegger, F. Sauer, G. Wolz, T. Kuwert, and X. Xu. A practical salient region feature based 3D multimodality registration method for medical images. *Medical Imaging 2006: Image Processing*, pages 870–879, 2006.

P.C. Hansen. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM Rev.*, 34(4):561–580, 1992. ISSN 0036-1445. doi: http://dx.doi.org/10.1137/1034115.

P.C. Hansen. The L-curve and its use in the numerical treatment of inverse problems. Technical report, Technical University of Denmark, 2005. URL `http://www.math.sintef.no/vskoler/2005/notes/Lcurve.pdf`.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2003a.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, chapter A3, pages 568–577. Cambridge University Press, second edition, 2003b.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, chapter 8, pages 225–256. Springer, 2001.

D. Heitz, P. Héaz, É. Mémin, and J. Carlier. Dynamic consistent correlation-variational approach for robust optical flow estimation. *Experiments on Fluids*, 45:595–608, 2008.

B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

S. Hsu, S. Acharya, A. Rafii, and R. New. Performance of a Time-of-Flight camera for intelligent vehicle safety applications. *Advanced Microsystems for Automotive Applications*. 2006.

P.J. Huber. *Robust Statistics*. Wiley, 1981.

M. Irani and P. Anandan. About direct methods. *Workshop on Vision Algorithms*, 1999.

H. J. Johnson and G. E. Christensen. Consistent landmark and intensity-based image registration. *IEEE Transactions on Medical Imaging*, 21(5):450–461, 2002.

S. Joshi and M. I. Miller. Landmark matching via large deformation diffeomorphisms. *IEEE Transactions on Image Processing*, 9:1357–1370, 2000.

F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–1000, 2002.

T. Kanade and M. Okutomi. A stereo matching algorithm with adaptive window: theory and experiment. *IEEE International Conference on Robotics and Automation*, 1991.

J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, volume 4, pages 502–506, 1953.

R. Koch, I. Schiller, B. Bartczak, F. Kellner, and K. Koeser. MixIn3D: 3D mixed reality with ToF-cameras. *Proceedings of the DAGM (German Association for Pattern Recognition) Syn3D Workshop*, 2009.

A. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmissions*, 1:1–7, 1965.

C. L. Lawson and R. J. Hanson. *Solving Least Squares Problem*. Prentice Hall, Englewood Cliffs, 1974.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944.

J. Lim and M.-H. Yang. A direct method for non-rigid motion with thin-plate spline. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

T. Lindeberg and B. ter Haar Romeny. *Linear scale-space*. Mathematical Imaging and Vision. ter Haar Romeny, 1994.

J. A. Little, D. L. G. Hill, and D. J. Hawkes. Deformations incorporating rigid structures. *Computer Vision and Image Understanding*, 66(2):223–232, 1997.

D. G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, pages 1150–1157, 1999.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

X. Luan. *Experimental Investigation of Photonic Mixer Device and Development of TOF 3D Ranging Systems Based on PMD Technology*. PhD thesis, University of Siegen, 2001.

K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems, 2nd edition. Technical University of Denmark, April 2004.

R. Malgouyres. *Algorithmes pour le synthèse d'images et l'animation 3D*. Collection Sciences Sup. Dunod, 2ème edition, 2005.

R. Malgouyres, F. Brunet, and S. Fourey. Binomial convolutions and derivatives estimation from noisy discretizations. *International Conference on Discrete Geometry for Computer Imagery*, 2008.

C.L. Mallows. Somme comments on cp. *Technometrics*, 15(4):661–675, 1973.

D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. *British Machine Vision Conference*, pages 384–396, 2002.

I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60 (2):135–164, November 2004.

P. J. Mc Carthy. Direct analytic model of the L-curve for thikhonov regularization parameter selection. *Inverse Problems*, 19:643–663, 2003.

C. R. Meyer, J. L. Boes, B. Kim, P. H. Bland, K. R. Zasadny, P. V. Kison, K. Koral, K. A. Frey, and R. L. Wahl. Demonstration of accuracy and clinical versatility of mutual information for automatic multimodality image fusion using affine and thin-plate spline warped geometric deformations. *Medical Image Analysis*, 1 (3):195–206, 1997.

J. Michot, A. Bartoli, F. Gaspard, and J.-M. Lavest. Algebraic line search for bundle adjustment. *British Machine Vision Conference*, 2009.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65:43–72, 2005.

J. Modersitzki. *Numerical Methods for Image Registration*. Oxford Science, 2004.

V.A. Morozov. The error principle in the solution of operational equations by the regularization method. *Computational Mathematics and Mathematical Physics*, 8:63–87, 1966.

V. Nannen. The paradox of overfitting. Master's thesis, Dutch National Research Institute for Mathematics and Informatics, 2003a.

V. Nannen. A short introduction to model selection, Kolmogorov complexity and Minimum Description Length (MDL). Technical report, Dutch National Research Institute for Mathematics and Informatics, 2003b.

S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

D. Nistér and H. Stewénius. Linear time maximally stable extremal regions. *European Conference on Computer Vision*, 2008.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

J.-M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation, 6(4):348-365, December 1995.*, 6:348–365, 1995.

S.I. Olsen and A. Bartoli. Implicit non-rigid structure-from-motion with priors. *Journal of Mathematical Imaging and Vision*, 31(2-3):233–244, 2008.

N. Papadakis and É. Mémin. Variational assimilation of fluid motion from image sequence. *SIAM Journal on Imaging Science*, 1:343–363, 2008.

J. Penne, K. Höller, M. Stürmer, T. Schrauder, A. Schneider, R. Engelbrecht, H. Feußner, B. Schmauss, and J. Hornegger. Time-of-Flight 3-D endoscopy. *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2009.

S. Pereverzev and E. Schock. Morozov's discrepancy principle for Tikhonov regularization of severely ill-posed problems in finite-dimensional subspaces. Technical report, University of Kaiserslautern, 1999.

M. Perriollat and A. Bartoli. A quasi-minimal model for paper-like surfaces. *Proceedings of the ISPRS International Workshop "Towards Benmarking Automated Calibration, Orientation, and Surface Reconstruction from Images"*, 2007.

M. Perriollat, R. Hartley, and A. Bartoli. Monocular template-based reconstruction of inextensible surfaces. *British Machine Vision Conference*, 2008.

M. Perriollat, R. Hartley, and A. Bartoli. Monocular template-based reconstruction of inextensible surfaces. *International Journal of Computer Vision*, 2010.

J. Pilet, V. Lepetit, and P. Fua. Real-time non-rigid surface detection. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

J. Pilet, V. Lepetit, and P. Fua. Fast non-rigid surface detection, registration and realistic augmentation. *International Journal of Computer Vision*, 76(2), 2008.

B. Pires and P. Aguiar. Registration of images with small overlap. *IEEE Workshop on Multimedia Signal Processing*, 2004.

D. Pizarro and A. Bartoli. Shadow resistant direct image registration. *Scandinavian Conference on Image Analysis*, 2007.

D. Pizarro and A. Bartoli. Feature-based deformable surface detection with self-occlusion reasoning. *3D Data Processing, Visualization and Transmission*, 2010.

J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual information based registration of medical images: A survey. *IEEE Transactions on Medical Imaging*, 22(8):986–1004, 2003.

T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

M. J. Powell. Five lectures on radial basis functions. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3600`.

M. Prasad, A. Zisserman, and A. W. Fitzgibbon. Single view reconstruction of curved surfaces. *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1345–1354, June 2006.

W.H. Press, S.A Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, 1992.

J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

S. Roberts and L. Stals. Discrete thin plate spline smoothing in 3D. Jagoda Crawford and A. J. Roberts, editors, *Proceedings of 11th Computational Techniques and Applications Conference CTAC-2003*, volume 45, pages C646–C659, 2004.

G. Rodriguez and D. Theis. An algorithm for estimating the optimal regularization parameter by the L-curve. *Rendiconti di Matematica*, 25:69–84, 2005.

S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D morphable model. *International Conference on Computer Vision*, 2003.

E. Ronchetti. Robustness aspects of model choice. *Statistica Sinica*, 7:327–338, 1997.

E. Ronchetti and R. Staudte. A robust version of mallows's cp. *Journal of the American Statistical Association*, 89(426):550–559, 1994.

P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388): 871–880, 1984.

D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes. Nonrigid registration using free-form deformations: Application to breast MR images. *IEEE Transactions on Medical Imaging*, 18:712–721, 1999.

M. Salzmann, R. Hartley, and P. Fua. Convex optimization for deformable surface 3-D tracking. *International Conference on Computer Vision*, 2007.

M. Salzmann, F. Moreno-Noguer, V. Lepetit, and P. Fua. Closed-form solution to non-rigid 3D surface registration. *European Conference on Computer Vision*, pages 581–594, 2008a.

M. Salzmann, R. Urtasun, and P. Fua. Local deformation models for monocular 3D shape recovery. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008b.

M. Salzmann, R. Urtasun, and P. Fua. Local deformation models for monocular 3D shape recovery. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174, 1998.

E. Schock. Parameter choice by discrepancy principles for the approximate solution of ill-posed problems. *Integral Equations and Operator Theory*, 7:895–898, 1984.

I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics*, 4:45–99, 1946.

L. Schumaker. *Spline functions: basic theory*. Wiley, 1981.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

F. Schweiger, B. Zeisl, P. Georgel, G. Schroth, E. Steinbach, and N. Navab. Maximum detector response markers for SIFT and SURF. *Vision, Modeling and Visualization Workshop*, 2009.

C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.

J. Shao. Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422): 486–494, 1993.

S. Shen, W. Shi, and Y. Liu. Monocular template-based tracking of inextensible deformable surfaces under L_2-norm. *Asian Conference on Computer Vision*, pages 214–223, 2009.

S. Shen, W. Shi, and Y. Liu. Monocular 3-D tracking of inextensible deformable surfaces under L_2-norm. *IEEE Transactions on Image Processing*, 19:512–521, 2010.

E.H. Shiguemori, H.F. de Campos Velho, and J.D.S. da Silva. Generalized discrepancy principle. *Inverse Problems, Design and Optimization Symposium*, 2004.

G. Silveira and E. Malis. Real-time visual tracking under arbitrary illumination changes. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2007.

R. Solomonoff. A formal theory of inductive inference, part 1 and 2. *Information and Control*, 7:1–22,224–254, 1964.

N. Sugiura. Further analysis of the data by Akaike' s information criterion and the finite corrections. *Communications in Statistics - Theory and Methods*, 7(1):13–26, 1978.

R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2:1–104, 2006.

U. Tautenhahn and U. Hämarik. The use of monotonicity for choosing the regularization parameter in ill-posed problems. *Inverse Problems*, 15:1487–1505, 1999.

A. Thompson and B. N. Taylor. *Guide for the Use of the International System of Units (SI)*, chapter 10, page 33. National Institute of Standards and Technology, 2008.

C. Tomasi and L. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.

P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. *Vision Algorithms*, 1999.

L. Torresani, A. Hertzmann, and C. Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):878–892, 2008.

A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 1936.

T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2007.

C. Ulrich, C. Schaller, J. Penne, and J. Hornegger. Evaluation of a time-of-flight based respiratory motion management system. *Bildverarbeitung für die Medizin*, 2010.

A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

L. Viarani, D. Stoppa, L. Gonzo, M. Gottardi, and A. Simoni. A CMOS smart pixel for active 3-D vision applications. *IEEE Sensors Journal*, 4(1):145–152, 2004.

P.A. Viola. *Alignment by Maximization of Mutual Information*. PhD thesis, Massachusetts Institute of Technology, 1995.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990.

G. Wahba and S. Wold. A completely automatic French curve: fitting spline functions by cross-validation. *Commun. Stat.*, 4:1–17, 1975.

J. Xiao, J. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. *International Journal of Computer Vision*, 67(2):233–246, 2006.

G. Xú and Z. Zhang. *Epipolar geometry in stereo, motion, and object recognition*. Springer, 1996.

Z. Xu, R. Schwarte, H. Heinol, B. Buxbaum, and T. Ringbeck. Smart pixel - photonic mixer device (PMD). Technical report, PMDTec, 2005.

R. Zabih and V. Kolmogorov. Spatially coherent clustering using graph cuts. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.

R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:690–706, 1999.

S. Zhang. *High-resolution, Real-time 3-D Shape Measurement*. PhD thesis, Stony Brook University, 2005.

Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15:59–76, 1997.

J. Zhu, S.C.H. Hoi, and M.R. Lyu. Nonrigid shape recovery by gaussian process regression. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

D. Zikic, M. Baust, A. Kamen, and N. Navab. Generalization of deformable registration in riemannian sobolev spaces. *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2010a.

D. Zikic, B. Glocker, O. Kutter, M. Groher, N. Komodakis, A. Kamen, N. Paragios, and N. Navab. Linear intensity-based image registration by markov random fields and discrete optimization. *Medical Image Analysis*, In Press, Accepted Manuscript, 2010b.

B. Zitová and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.