Variational modeling of curves and surfaces

Wesselink, J.W.

DOI: 10.6100/IR465317

Published: 01/01/1996

Citation for published version (APA):
Variational Modeling
of
Curves and Surfaces

Proefschrift

ter verkrijging van de graad van doctor aan
de Technische Universiteit Eindhoven, op gezag
van de Rector Magnificus, prof.dr. M. Rem,
voor een commissie aangewezen door het Col-
lege van Dekanen in het openbaar te verdedigen
op donderdag 12 september 1996 om 16.00 uur

door

Jan Willem Wesselink
geboren te Zutphen
Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.dipl.-ing. D.K. Hammer
prof.dr. H. Hagen

en de copromotor

dr. R.C. Veltkamp
Contents

1 Introduction ............................................. 1
  1.1 Organization ........................................... 2

2 Mathematical background ............................. 5
  2.1 Introduction ........................................... 5
  2.2 Notations .............................................. 6
  2.3 Tensor Analysis ....................................... 7
    2.3.1 Einstein convention ................................ 7
    2.3.2 Bases of a vector space ............................ 7
    2.3.3 Covariant and contravariant components .......... 7
    2.3.4 Dual space ....................................... 8
    2.3.5 Tensors .......................................... 8
    2.3.6 Tensor fields on $\mathbb{R}^n$ ..................... 9
    2.3.7 Differentiation operations on tensor fields .... 11
  2.4 Differential geometry ............................... 13
    2.4.1 Space curves ...................................... 13
    2.4.2 Surfaces ......................................... 18
    2.4.3 Space curves in conventional notation .......... 21
    2.4.4 Surfaces in conventional notation ............... 23
    2.4.5 Smoothness integrals for curves ................ 25
    2.4.6 Approximations of smoothness integrals ......... 25
    2.4.7 Iterative minimization of smoothness integrals . 29
    2.4.8 Smoothness integrals for surfaces ............... 32
    2.4.9 Approximations of smoothness integrals .......... 33
  2.5 Numerical methods .................................. 34
    2.5.1 Linear systems ................................... 34
    2.5.2 Unconstrained minimization ....................... 36
    2.5.3 Nonlinear programming ............................ 37
    2.5.4 Quadratic programming ............................ 38
    2.5.5 Numerical integration ............................ 40
  2.6 Miscellaneous ....................................... 42
    2.6.1 Distance between point and line ................ 42
    2.6.2 Distance between point and plane ............... 42


<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.3</td>
<td>Parallel projection</td>
<td>42</td>
</tr>
<tr>
<td>2.6.4</td>
<td>Quadratic functions</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>Curves</td>
<td>45</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>45</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Fairness</td>
<td>45</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Splines</td>
<td>46</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Other fairness functionals</td>
<td>48</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Modeling with minimal energy curves</td>
<td>49</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Data dependent fairness functionals</td>
<td>50</td>
</tr>
<tr>
<td>3.1.6</td>
<td>External energy operators</td>
<td>50</td>
</tr>
<tr>
<td>3.1.7</td>
<td>Curve representation</td>
<td>51</td>
</tr>
<tr>
<td>3.1.8</td>
<td>Minimizing the energy</td>
<td>54</td>
</tr>
<tr>
<td>3.2</td>
<td>Constraints</td>
<td>55</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Positional constraints</td>
<td>55</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Directional constraints</td>
<td>57</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Continuity constraints</td>
<td>58</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Other constraints</td>
<td>59</td>
</tr>
<tr>
<td>3.3</td>
<td>Internal energy</td>
<td>61</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Bend energy</td>
<td>61</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Stretch energy</td>
<td>64</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Curves with minimal bend or stretch energy</td>
<td>65</td>
</tr>
<tr>
<td>3.3.4</td>
<td>The effect of scaling and parameter transformations</td>
<td>66</td>
</tr>
<tr>
<td>3.4</td>
<td>External energy</td>
<td>68</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Positional attractors</td>
<td>68</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Directional attractors</td>
<td>73</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Repellors</td>
<td>75</td>
</tr>
<tr>
<td>3.4.4</td>
<td>The effect of scaling a curve and its parametrization</td>
<td>76</td>
</tr>
<tr>
<td>3.5</td>
<td>Curve representations</td>
<td>78</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Bézier curves</td>
<td>78</td>
</tr>
<tr>
<td>3.5.2</td>
<td>B-spline curves</td>
<td>80</td>
</tr>
<tr>
<td>3.6</td>
<td>Implementation and results</td>
<td>82</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Functionality of the program</td>
<td>82</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Uniform cubic B-spline curves</td>
<td>84</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Computing the constraints</td>
<td>85</td>
</tr>
<tr>
<td>3.6.4</td>
<td>Computing the internal energy</td>
<td>85</td>
</tr>
<tr>
<td>3.6.5</td>
<td>Computing the external energies</td>
<td>88</td>
</tr>
<tr>
<td>3.6.6</td>
<td>Solving the minimization problem</td>
<td>89</td>
</tr>
<tr>
<td>3.6.7</td>
<td>Examples</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>Surfaces</td>
<td>97</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>97</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Thin plate energy</td>
<td>98</td>
</tr>
</tbody>
</table>
CONTENTS

4.1.2 Other fairness functionals ................................................. 98
4.1.3 Modeling with minimal energy surfaces ............................... 100
4.1.4 Data dependent fairness functionals .................................. 101
4.1.5 External energy operators ................................................ 101
4.1.6 Surface representations ................................................... 102
4.2 Constraints ............................................................................ 104
4.2.1 Positional constraints ......................................................... 104
4.2.2 Directional constraints ....................................................... 105
4.2.3 Continuity constraints ......................................................... 106
4.2.4 Curve interpolation constraints .......................................... 106
4.3 Internal energy ...................................................................... 108
4.3.1 Bend energy ....................................................................... 108
4.3.2 Stretch energy .................................................................... 111
4.3.3 Surfaces with minimal bend or stretch energy ...................... 113
4.3.4 The effect of scaling and parameter transformations ............ 113
4.4 External energy ...................................................................... 116
4.4.1 Positional attractors ........................................................... 116
4.4.2 Directional attractors .......................................................... 118
4.4.3 The effect of scaling and parameter transformations .......... 119
4.5 Surface representations .......................................................... 121
4.5.1 Tensor product B-spline surfaces ....................................... 121
4.5.2 Triangular Bézier Surfaces ................................................. 122
4.5.3 Triangular B-spline surfaces .............................................. 124
4.6 Implementation and results ..................................................... 131
4.6.1 Computation ..................................................................... 131
4.6.2 Solving the minimization problem ...................................... 136
4.6.3 Interactive program ............................................................ 137
4.6.4 Implementation aspects ....................................................... 138
4.6.5 Examples .......................................................................... 138

5 Conclusions and further work .................................................. 147
5.1 Conclusions .......................................................................... 147
5.2 Limitations ............................................................................ 148
5.3 Improvements and further work ............................................. 148
Variational Modeling of Curves and Surfaces
Chapter 1

Introduction

This dissertation is concerned with curve and surface modeling. In the past, curves and surfaces were modeled by first making wooden models, from which the final results were obtained by means of ‘copy-milling’. Since the early 1960s, people have started to use computers to support the design of curves and surfaces. The increasing computing power and the increasing potential of graphical hardware in the last decade has opened many new possibilities for Computer Aided Geometric Design (CAGD). Especially the field of curve and surface modeling is rapidly evolving. This is reflected in the recently started series called ‘Geometric Design Publications’ that contains several books on curve and surface modeling (among others [Hag92a], [Hag92b] and [Sap94]).

The most important applications of curve and surface modeling are found in industrial design, for instance in the automobile, aerospace and shipbuilding industries. An often encountered application is the interpolation and approximation of data with B-splines and Bézier curves and surfaces or other representation schemes. Another related application is variational curve and surface modeling, where one is looking for smooth curves and surfaces that satisfy a number of geometric constraints. See figure 1.1 for an example of variational modeling.

![Example of variational modeling](image)

Figure 1.1: Example of variational modeling. The left figure shows a number of geometric interpolation constraints, specified by a designer. The figure on the right shows a smooth surface satisfying these constraints, that was automatically generated using a computer.

The goal of this dissertation is to develop efficient methods that can be used for interactive variational modeling of curves and surfaces. The following aspects related to this subject will be discussed:

1
CHAPTER 1 INTRODUCTION

- There are several \textit{fairness} functionals that can be used to measure the smoothness of a curve or surface. The thin plate functional is an example of a fairness functional that tends to yield high quality surfaces. However, this functional is far too complicated to use in interactive applications. On the other hand, the current quadratic approximations of the thin plate energy can be very poor. In this dissertation a \textit{data dependent} approximation to the thin plate functional is defined, which is based on an idea of Greiner (see [Gre91] and [GLW96]). This is a quadratic approximation that generally leads to much better results, provided that a suitable \textit{reference surface} is given.

- A shortcoming of current variational modeling applications is that interaction with constraints only is not always flexible enough. In this dissertation a number of design operators is introduced that can be used to deform a \textit{minimal energy} surface.

- Especially for surfaces, the number of degrees of freedom that determine the shape may be very large. The optimization problems corresponding to variational modeling are therefore not straightforward to solve. In this dissertation some attention is paid to numerical methods that can be used to solve these problems.

- The choice of the curve and surface representation is an important issue in variational modeling. In [DMS92] and [Pfe95] a surface representation scheme is discussed with some very pleasing properties: triangular B-spline surfaces. These piecewise polynomial surfaces are globally $C^{n-1}$ continuous (with $n$ the degree of the surface) and they consist of triangular patches. The combination of these two properties is very appropriate for variational modeling. The $C^{n-1}$ continuity is important for obtaining smooth surfaces and modeling with triangular patches is much more flexible than modeling with rectangular patches. An interactive variational modeling application is described that uses arbitrary degree triangular B-spline surfaces.

1.1 Organization

In chapter 2 a mathematical foundation of curves and surfaces is given. The most important geometric properties of curves and surfaces are given, and some properties of data dependent approximations of smoothness functionals are derived. In this chapter we are using notations from tensor analysis. This is rather unusual in geometric modeling, but it has some significant advantages. In the first place, this makes it very easy to determine which of the curve and surface properties are parametrization (in)dependent. In the second place, most of the definitions and proofs become much shorter and clearer. The most important properties are also formulated in a more conventional notation. This notation is adopted in the rest of the dissertation.

The chapters 3 and 4 deal with curves and surfaces. The structure of these chapters is similar. First an introduction is given (3.1 and 4.1), in both cases followed by sections on constraints (3.2 and 4.2), internal (3.3 and 4.3) and external energy (3.4 and 4.4). The sections on constraints describe the geometric constraints that can be applied to a curve
1.1 ORGANIZATION

or surface. The sections on internal energy describe the fairness functionals that are used to create smooth curves and surfaces. The sections on external energy describe the design operators that are used to deform a minimal energy curve or surface. In the sections (3.5 and 4.5) some curve and surface representation schemes and their suitability for variational modeling are discussed. The next sections (3.6 and 4.6) deal with implementations of variational modeling for curves and surfaces. Both chapters are finished with some conclusions and suggestions for further work.
Chapter 2

Mathematical background

2.1 Introduction

With geometric modeling of curves and surfaces a lot of mathematics is involved. This is particularly the case for variational modeling. The ‘fairness functionals’ that are used to measure the smoothness of a curve or surface have to be mathematically defined and for the computation of the optimal curve or surface all kinds of numerical methods are needed.

The sections 2.3 and 2.4 on tensor analysis and differential geometry are very technical. They are aimed at giving a thorough description of the geometrical properties of curves and surfaces and at the introduction of so-called data dependent fairness functionals. Section 2.5 gives an overview of the numerical methods used to solve the minimization problems that occur in this dissertation. This section will mainly be used as reference material.

For readers with a basic knowledge of differential geometry, it should be possible to skip most of these sections and go directly to chapter 3.
2.2 Notations

In this section the notations that are most often used in this dissertation are explained.

- For two elements $x, y \in \mathbb{R}^n$, the Euclidean inner product is denoted as $x \cdot y$.
- For two elements $x, y \in \mathbb{R}^2$, the cross product is denoted as $x \times y$.
- The first and second derivative of a function $x : I \subset \mathbb{R} \to \mathbb{R}^n$ are denoted as $\dot{x}$ and $\ddot{x}$.
- The partial derivatives of a function $F : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^n$ are denoted with subscripts. For example, $F_{uv} = \frac{\partial^2 F}{\partial u \partial v}$.
- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is denoted as $\nabla f$.
- The $n$-dimensional unit matrix is denoted as $I_n$.
- The $n$-dimensional simplex is denoted as $T_n$.
- The matrix representations of the first and second fundamental form of a surface $F$ are denoted as $I_F$ and $\Pi_F$. 
2.3 Tensor Analysis

This section gives a short introduction in tensor analysis (see [BG95]). The goal of this section is to prepare for the next section on differential geometry. Readers who are not interested in the mathematical derivations of the results of the differential geometry section may skip the tensor analysis section, since these results will also be formulated without using notations from tensor analysis.

Throughout this section we assume that $V$ is an $n$-dimensional vector space over $\mathbb{R}$ with an inner product $(.,.)$.

2.3.1 Einstein convention

In tensor analysis the Einstein summation convention is used. This convention states that if an index occurs in an expression as a superindex and as a subindex, this expression is considered as a summation. Thus, $\sum_{i=1}^n A^i_j x^i$ is written as $A^i_j x^i$ and $\sum_{j=1}^n T^i_j$ as $T^i_j$.

2.3.2 Bases of a vector space

Let $\{e_i\}$ be a basis of $V$. We define the numbers $g_{ij}$ as $g_{ij} = (e_i, e_j)$. The matrix $G = [g_{ij}]$, with $i$ the rowindex and $j$ the columnindex, is called the Gram matrix corresponding with the basis $\{e_i\}$. The elements of the inverse of $G$ are denoted as $g^{kl}$, so $G^{-1} = [g^{kl}]$. Note that $g_{ik}g^{kj} = \delta^i_j$ and $g^{ik}g_{ik} = \delta^i_i$, where $\delta^i_i = 1$ if $i = j$ and 0 otherwise. For the basis $\{e_i\}$ we define a second basis $\{e^i\}$ by $e^i = g^{ij}e_j$. This basis is called the reciprocal basis corresponding with $\{e_i\}$.

The vectors $e_i$ and $e^i$ have the following relationships:

\[
\begin{align*}
e^i &= g^{ij}e_j \\
e_i &= g_{ij}e^j \\
(e_i, e_j) &= g_{ij} \\
(e^i, e^j) &= g^{ij} \\
(e^i, e_j) &= \delta^i_j.
\end{align*}
\]

For every pair of bases $\{e_i\}$ and $\{e^j\}$ of $V$ there exist two unique, ordered collections of $n^2$ real numbers $A^\nu_i$ and $A_i^\nu$ such that

\[
\begin{align*}
e_i &= A_i^\nu e^\nu, \\
e^\nu &= A^\nu_i e_i.
\end{align*}
\]

2.3.3 Covariant and contravariant components

For every vector $x \in V$ there exists a unique ordered collection of real numbers $x^i$ such that $x = x^i e_i$. The numbers $x^i$ are called the contravariant components of $x$ with respect to
the basis \{e_i\}. There also exists a unique collection of real numbers \(x_i\) such that \(x = x_i e_i\). These numbers \(x_i\) are called the covariant components of \(x\) with respect to the basis \{e_i\}. Contravariant components are always stored in a column vector and covariant components are stored in a row vector.

### 2.3.4 Dual space

The vector space containing all linear functions from \(V\) to \(\mathbb{R}\) is called the dual space and is denoted as \(V^*\). The elements of \(V^*\) can be corresponded to the elements of \(V\) by the following theorem:

**Theorem 1 (Riesz)** For every linear function \(\Phi \in V^*\) there exists exactly one vector \(f \in V\) such that \(\Phi(x) = (f, x)\) for every \(x \in V\).

The function \(\Phi\) corresponding with \(f\) will be denoted as \(\hat{f}\). Because of this correspondence between \(V\) and \(V^*\), the elements of \(V^*\) are sometimes interpreted as elements of \(V\). Evidently, the set \{\(\hat{e}_i\)\} forms a basis of \(V^*\).

### 2.3.5 Tensors

A \(p\)-tensor, \(p \in \mathbb{N}\), is a mapping from \(V^p = V \times \ldots \times V\) (\(p\) times) to \(\mathbb{R}\) that is linear in all of its arguments. The set of all \(p\)-tensors is an \(n^p\)-dimensional vector space. We denote this set as \(T_p(V)\).

**0-tensors**

A 0-tensor is defined as a mapping from \(V\) to \(\mathbb{R}\) that adds to every \(x \in V\) the same real number. A 0-tensor is sometimes called a *scalar*.

**1-tensors**

A 1-tensor is a linear mapping from \(V\) to \(\mathbb{R}\). The set of all 1-tensors is therefore the dual space of \(V\).

**p-tensors**

The dimension of the set \(T_p(V)\) of all \(p\)-tensors is equal to \(n^p\). With respect to a basis \{\(e_i\)\} of \(V\), a \(p\)-tensor can therefore be represented as a \(p\)-dimensional matrix containing \(n^p\) elements. Such a ‘supermatrix’ is sometimes called a *holor*. There are different choices for this representation. One possible choice is the set of numbers \(T_{i_1 \ldots i_p} = T(e_{i_1}, \ldots, e_{i_p})\). These numbers are called the covariant components of the \(p\)-tensor \(T\) with respect to the basis \{\(e_i\)\}. Another choice is the set of numbers \(T^{i_1 \ldots i_p} = T(e^{i_1}, \ldots, e^{i_p})\), called the contravariant components of \(T\) with respect to the basis \{\(e_i\)\}. A more general choice is the set of numbers \(T^{k_1 \ldots k_r}_{i_1 \ldots i_s} = T(e_{k_1}, \ldots, e_{k_r}, e^{i_1}, \ldots, e^{i_s})\), with \(r + s = p\).
2.3 TENSOR ANALYSIS

Let $S$ be a mapping that adds to a tensor its matrix representation with respect to a given basis of $V$. The following theorem expresses the relation between representations with respect to two different bases of $V$.

**Theorem 2** The set of numbers $T^{k_1,\ldots,k_r}_{i_1,\ldots,i_s}$ form the components of an \((r\choose s)\)-tensor if the following holds. For any pair of bases of $V$, \(\{e_i\}\) and \(\{e'_i\}\)

\[
S(\{e_i\}) = \left[ T^{k_1,\ldots,k_r}_{i_1,\ldots,i_s} \right] \Rightarrow T^{k'_1,\ldots,k'_r}_{i'_1,\ldots,i'_s} = A^k_{i_1} \cdots A^{k_r}_{i_s} A^k_{i'_1} \cdots A^{k_r}_{i'_s} T^{k_1,\ldots,k_r}_{i_1,\ldots,i_s},
\]

where the numbers $A^k_{i}$ are given by (2.1).

A \((0\choose p)\)-tensor is also known as a covariant $p$-tensor and a \((q\choose 0)\)-tensor is also known as a contravariant $q$-tensor.

This alternative way of defining a tensor gives us an easy method to examine if a given set of numbers $T^{k_1,\ldots,k_r}_{i_1,\ldots,i_s}$ form the components of a tensor: we just have to check if these numbers behave like (2.2) under a transition to another basis from $V$.

As an example we give the **Kronecker tensor** on $V$. This is the mapping $F_1$ that adds to each basis of $V$ the matrix $[\delta^l_k]$, where $\delta^l_k = 1$ if $l = k$ and 0 otherwise. We can prove that this is a \((1\choose 1)\)-tensor as follows: let \(\{e_i\}\) and \(\{e'_i\}\) be two bases from $V$, such that $F_1(\{e_i\}) = [\delta^l_k]$ and $F_1(\{e'_i\}) = [\delta'^{l'}_{k'}]$. Then we find

\[
\delta'^{l'}_{k'} = A^k_{l'} A^k_l = A^k_{l'} A^k_l \delta^l_k,
\]

which completes the proof. We will now show (for the case $n = 2$) that the Kronecker tensor is not a \((0\choose 2)\)-tensor. Let the bases \(\{e_i\}\) and \(\{e'_i\}\) be given as follows: $e_1 = e'_1 = \binom{1}{0}$, $e_2 = \binom{0}{1}$ and $e'_2 = \binom{1}{1}$. Then $T(e'_1, e'_2) = T(e_1, e_1 + e_2) = T(e_1, e_1) + T(e_1, e_2) = \delta^1_1 + \delta^1_2 = 1 + 0 = 1$. This is a contradiction, since for a \((0\choose 2)\)-tensor $T(e'_1, e'_2)$ would be 0.

2.3.6 Tensor fields on $\mathbb{R}^n$

In this section we will define tensor fields on $\mathbb{R}^n$. Intuitively speaking, a tensor field on $\mathbb{R}^n$ adds a tensor to each point $X$ of $\mathbb{R}^n$. This tensor is defined on the tangent space $T_X(\mathbb{R}^n)$ at the point $X$, which is just a copy of $\mathbb{R}^n$, labeled with the point $X$. The vector space of all $(r + s)$-dimensional holors with $s$ subindices and $r$ superindices will be denoted as $T^{r,s}_X(\mathbb{R}^n)$.

**Coordinate systems**

Let $\Omega$ be an open subset of $\mathbb{R}^n$. A system of $n$ real valued functions $\{f^i(X)\}$, defined on $\Omega$, is called a **curvilinear coordinate system** on $\Omega$ if:

- The mapping $f = (f^1, \ldots, f^n)^T$ from $\Omega$ to $\mathbb{R}^n$ is injective. We denote $u^i = f^i(X) = f^i(x^jE_j)$. 
• The set \( U = f(\Omega) \) is open.

• The mapping \( f \) is differentiable in each point \( X \in \Omega \) and \( \det \left[ \frac{\partial f^i}{\partial \sigma^j} \right] \neq 0 \) for each \( X \in \Omega \).

The inverse mapping \( f^{-1} : U \to \Omega \) is called a parametrization of \( \Omega \). The variables \( x^j \) are functions of the variables \( u^i \). The vectors \( \partial_i X = \frac{\partial X}{\partial u^i} \) form, in a natural way, a basis of the tangent space \( T_X(\mathbb{R}^n) \) at the point \( X \).

For example, let \( \Omega = \mathbb{R}^2 \setminus \{(x, 0) | x \in [0, \infty)\} \). Polar coordinates on \( \Omega \) are defined by the parametrization

\[
x = r \cos \varphi, \quad y = r \sin \varphi,
\]

with \( x = x^1, y = x^2, r = u^1 \) and \( \varphi = u^2 \). A basis of the tangent space in \( X = (x, y)^T \) is given by \( \partial_1 X = (\cos \varphi, \sin \varphi)^T \) and \( \partial_2 X = (-\sin \varphi, \cos \varphi)^T \). With some effort, these vectors may be expressed in \( x \) and \( y \).

**Tensor fields**

An \( \left( \frac{r}{s} \right) \)-tensor field \( \varphi \) on \( \mathbb{R}^n \) is a mapping from \( \mathbb{R}^n \) to \( T^r_s(T_X(\mathbb{R}^n)) \) that adds an \( \left( \frac{r}{s} \right) \)-tensor to each point \( X \in \mathbb{R}^n \). We will give four examples of tensor fields: scalar fields, vector fields, the Kronecker tensor field and the (first) fundamental tensor field.

**Scalar fields**

A scalar field on \( \mathbb{R}^n \) is a tensor field that adds a \( \left( \frac{0}{0} \right) \)-tensor to each point \( X \in \mathbb{R}^n \). Intuitively speaking, a scalar field is just a mapping from \( \mathbb{R}^n \) to \( \mathbb{R} \).

**Vector fields**

A vector field or \( \left( \frac{1}{0} \right) \)-tensor field \( \mathbf{a} \) on \( \mathbb{R}^n \) is a mapping from \( \mathbb{R}^n \) to \( \cup_{X \in \mathbb{R}^n} T_X(\mathbb{R}^n) \). To each point \( X \in \mathbb{R}^n \) a vector \( \mathbf{a}(X) \) is attached that lies in the tangent space \( T_X(\mathbb{R}^n) \) corresponding to \( X \). There belong \( n \) functions \( a^i \) on \( \mathbb{R}^n \) to the vector field \( \mathbf{a} \) such that \( \mathbf{a}(X) = a^i(x^k) \frac{\partial}{\partial x^i} \), with \( \frac{\partial}{\partial x^i} \) the \( i \)-th basis vector.

A covector field or \( \left( \frac{0}{1} \right) \)-tensor field \( \alpha \) on \( \mathbb{R}^n \) is a mapping from \( \mathbb{R}^n \) to \( \cup_{X \in \mathbb{R}^n} T^*_X(\mathbb{R}^n) \). To each point \( X \in \mathbb{R}^n \) an element \( \alpha(X) \) from the dual space \( T^*_X(\mathbb{R}^n) \) of \( T_X(\mathbb{R}^n) \) is attached. There belong \( n \) functions \( \alpha_i \) on \( \mathbb{R}^n \) to the covector field \( \alpha \) such that \( \alpha(X) = \alpha_i(x^k)dx^i \), with \( dx^i \) the \( i \)-th reciprocal basis vector.

Using the theorem from Riesz, both a vector field on \( \mathbb{R}^n \) and a covector field on \( \mathbb{R}^n \) may be interpreted as a mapping that attaches an \( n \)-dimensional vector to each element from \( \mathbb{R}^n \).

**The Kronecker tensor field**

The Kronecker tensor field is the \( \left( \frac{1}{1} \right) \)-tensor field on \( \mathbb{R}^n \) that adds to each \( X \in \mathbb{R}^n \) the Kronecker tensor in \( T^1(T_X(\mathbb{R}^n)) \).
2.3 TENSOR ANALYSIS

The fundamental tensor field

Let \((\cdot, \cdot)\) be a symmetric inner product on \(\mathbb{R}^n\). Let \(x, y \in T_X(\mathbb{R}^n)\) and define the inner product \((\cdot, \cdot)_X\) by

\[(x, y)_X = \delta_{ij}x^i y^j,
\]

where \(x^i\) and \(y^i\) are the components of \(x\) and \(y\) with respect to the standard basis in \(T_X(\mathbb{R}^n)\). The inner product \((\cdot, \cdot)_X\) is a \(\left( \begin{smallmatrix} 0 \\ 2 \end{smallmatrix} \right)\)-tensor on \(T_X(\mathbb{R}^n)\). The fundamental tensor field \(g\) is the \(\left( \begin{smallmatrix} 0 \\ 2 \end{smallmatrix} \right)\)-tensor field on \(\mathbb{R}^n\) that adds this inner product \((\cdot, \cdot)_X\) to each point \(X \in \mathbb{R}^n\).

2.3.7 Differentiation operations on tensor fields

In this section some differentiation operations on scalar and vector fields will be defined.

The gradient

Let \(f\) be a scalar field on \(\mathbb{R}^n\), let \(\{x^i\}\) be a curvilinear coordinate system on \(\mathbb{R}^n\) and let \(\{x'^i\}\) be another curvilinear coordinate system on \(\mathbb{R}^n\). Since

\[
\frac{\partial f}{\partial x'^i} = \frac{\partial x'^i}{\partial x^i} \frac{\partial f}{\partial x^i},
\]

the functions \(\partial f = \frac{\partial f}{\partial x^i}\) form the components of a \(\left( \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right)\)-tensor field. This covector field is called the gradient field of \(f\) and will be denoted as

\[
\text{grad}(f) = \partial_i f dx^i,
\]

where \(dx^i\) is the \(i\)-th reciprocal basis vector with respect to the standard basis of \(T_X(\mathbb{R}^n)\).

The directional derivative

Let \(a\) be a vector field. Let \(a^i\) be the components of this vector field with respect to the curvilinear coordinates \(x^i\). The functions \(a^i \partial_i f\) then form the components of a scalar field. This scalar field is called the directional derivative of \(f\) in the direction \(a\) and will be denoted as

\[
\mathcal{L}_a f = a^i \partial_i f.
\]

The covariant derivative

Let \(a\) be a vector field. Let \(a^i\) be the components of this vector field with respect to the curvilinear coordinates \(x^i\). The second derivative vectors \(\partial_j \partial_k X = \frac{\partial^2 X}{\partial u^j \partial u^k}\) can be expressed in the basis \(\{\partial_i X\}\) of the tangent space \(T_X(\mathbb{R}^n)\) as follows:

\[
\partial_j \partial_k X = \Gamma^i_{jk} \partial_i X,
\]
where the $n^3$ functions $\Gamma^i_{jk}$ are called Christoffel symbols. It can be proven that the $n^2$ functions $\nabla_j a^i$ defined by

$$\nabla_j a^i = \partial_j a^i + \Gamma^i_{jk} a^k$$

form the components of a $(1,1)$-tensor field. This tensor field is called the covariant derivative of the vector field $\mathbf{a}$, and will be denoted as $\nabla \mathbf{a}$. 

2.4 Differential geometry

To give geometric modeling a mathematical foundation, we need to turn to differential geometry. The goal of this section is twofold. In the first place some important notions like the curvature and the first and second fundamental forms are defined, which can be used to characterize the local geometry in a point of a curve or surface. In the second place some smoothness or fairness integrals will be defined that will be used later as a measure for the smoothness of a curve or surface. An important part of the section is devoted to the derivation of so-called data dependent approximations to these integrals, which is an idea of Greiner (see [Gre94] and [GLW96]).

This section uses tensor analysis. The advantage of this approach is that notions like the gradient and the Hessian become easier to define, and that the proofs of some results will be shorter and clearer. Another advantage is that it is immediately clear if the curve and surface properties that will be defined are parametrization dependent. If a property is defined as a tensor (on the tangent space) then it doesn’t depend on the basis of this tangent space, and hence it doesn’t depend on the parametrization. A disadvantage is that this forces the reader to get used to the notations of tensor analysis. Therefore, in the sections 2.4.3 and 2.4.4 a short review will be given in a more conventional notation. The same notation is also used in the rest of this dissertation.

2.4.1 Space curves

Let \( \{E_i\} \) be the standard basis of \( \mathbb{R}^3 \), such that any point \( X \in \mathbb{R}^3 \) can be written as \( X = x^i E_i \). Now let \( x^i \) be real valued functions from a real parameter \( t \), where \( t \) traverses a certain interval \( I \). We assume that the functions \( x^i \) are sufficiently smooth, such that there are no problems regarding differentiation. Furthermore we assume that the derivatives \( \frac{dx^i}{dt} \) are not simultaneously zero for all \( t \in I \).

Definition of a space curve

A space curve \( K \) (also called a parametric curve in \( \mathbb{R}^3 \)) is the set of points \( X = X(t) = x^i(t) E_i \), where \( t \) traverses the interval \( I \) and where the mapping \( t \mapsto X(t) \) is injective.

The representation \( x^i(t) \) of a space curve \( K \) is called a parametrization. The vector \( \frac{dx^i}{dt} E_i \) is the tangent vector at the space curve in the point \( X \), which will be denoted as \( \text{d} X / \text{d} t \). Another parametrization \( K \) can be obtained by replacing \( t \) with \( f(u) \), where \( f \) is a monotonous function, such that \( f(u) \) traverses the interval \( I \) as the parameter \( u \) traverses another interval \( J \). We assume that the function \( f \) is sufficiently smooth and has a derivative that is nowhere zero. The transition to another parametrization, via \( t = f(u) \), is called a parameter transformation.
CHAPTER 2 MATHEMATICAL BACKGROUND

Arc length parametrization

The length of a (finite) curve $K$, represented by the parametrization $x^i(\tau)$, with $t_0 \leq \tau \leq t$, is given by

$$s(t) = \int_{t_0}^{t} \sqrt{\left(\frac{dx^1(\tau)}{d\tau}\right)^2 + \left(\frac{dx^2(\tau)}{d\tau}\right)^2 + \left(\frac{dx^3(\tau)}{d\tau}\right)^2} \, d\tau. \quad (2.3)$$

The in this way introduced function $s$ can be used as a parameter for space curves. The parameter is then $s$ and is called arc length. Since the integral in $(2.3)$ is often difficult or impossible to compute, the arc length parametrization is mostly used for theoretical goals.

In the sequel we will use the Euclidean inner product $(\cdot, \cdot)$ on $\mathbb{R}^3$. The arc length parametrization is special in the sense that the tangent vector has unit length:

$$\left(\frac{dX}{ds}, \frac{dX}{ds}\right) = 1.$$

Tangent line and osculation plane

The tangent line at a curve $K$ in a point $X$ is the straight line given by the parameter representation

$$Y = X + \lambda \frac{dX}{ds}.$$

The parameter $\lambda$ is such that $|\lambda|$ is the distance from the tangent point $X$ along the tangent line. The tangent line is the straight line that best approximates the curve at $X$.

The osculation plane at a curve $K$ in a point $X$ is the plane given by the parameter representation

$$Y = X + \lambda \frac{dX}{ds} + \mu \frac{d^2X}{ds^2},$$

or by the equivalent equation

$$\det \left( Y - X, \frac{dX}{ds}, \frac{d^2X}{ds^2} \right) = 0.$$

Here we have assumed that the vectors $\frac{dX}{ds}$ and $\frac{d^2X}{ds^2}$ are independent vectors, so in particular $\frac{d^2X}{ds^2} \neq 0$. In geometric terms, it means that $X$ is not an inflection point. However, also in inflection points it is possible to define the osculation plane:

$$\det \left( Y - X, \frac{dX}{ds}, \frac{d^3X}{ds^3} \right) = 0.$$

For an arbitrary parametrization of a space curve $K$, with parameter $t$, the parameter representations of the tangent line and the osculation plane in $X$, with $\frac{d^2X}{dt^2} \neq 0$, are given by

$$Y = X + \lambda \frac{dX}{dt}$$

and

$$Y = X + \lambda \frac{dX}{dt} + \mu \frac{d^2X}{dt^2}.$$
2.4 Differential Geometry

Frenet frame

We will now introduce a special local basis, linked to a point \( X(t) \) of a space curve \( K \), that will significantly facilitate the description of local curve properties at that point. Suppose that \( K \) is parametrized by the arc length \( s \). Let \( X \) be a fixed point on \( K \) that is not an inflection point. The tangent vector \( \frac{dX}{ds} \) has unit length and will be denoted as \( t \), so \( \frac{dX}{ds} = t \). A straight line through the point \( X \), perpendicular to the tangent line in \( X \), is called a normal. The normals in \( X \) form a plane, the normal plane in \( X \). The normal in \( X \) that lies in the osculation plane is called the main normal. The normal that is perpendicular to the main normal is called binormal. Now we can introduce vectors \( n \) and \( b \) that are unit vectors on the main normal and binormal lines. The directions of these vectors are chosen such that \( n \) points in the direction of \( \dot{X} \) and \( b \) points in the direction of \( \frac{dX}{ds} \times \frac{d^2X}{ds^2} \). The three vectors \( t, n \) and \( b \) form in every point an orthonormal basis, called Frenet frame. The Frenet frame varies its orientation as \( t \) traces out the curve (see figure 2.1).

Curvature and torsion

The local geometry in a point \( X \) of a space curve \( K \) can be characterized by two properties, the curvature and the torsion. Intuitively speaking, one can say that the curvature is a measure for how strongly the curve is bent, and the torsion is a measure for the ‘non-planarity’ of the curve at the point \( X \). For example, on a straight line the curvature is everywhere zero, and on a planar curve the torsion is everywhere zero. Let \( Y \) be a point on the space curve \( K \) in the neighborhood of \( X \). Let \( \Delta \varphi \) be the angle between the tangent lines in \( X \) and \( Y \) and let \( \Delta \psi \) be the angle between the binormals in \( X \) and \( Y \). The curvature \( \kappa \) and the torsion \( \tau \) of the space curve \( K \) in the point \( X \) are defined by

\[
\kappa^2 = \left( \frac{d\varphi}{ds} \right)^2 = \lim_{\Delta s \to 0} \left( \frac{\Delta\varphi}{\Delta s} \right)^2,
\]

\[
\tau^2 = \left( \frac{d\psi}{ds} \right)^2 = \lim_{\Delta s \to 0} \left( \frac{\Delta\psi}{\Delta s} \right)^2.
\]

The sign of \( \kappa \) is always taken positive, and the sign of \( \tau \) follows from the Frenet formulas, which will be given later. The curvature is a measure for the change in direction of the tangent line. The value \( \rho = \frac{1}{\kappa} \) is called the radius of curvature. In inflection points, the curvature is defined as \( \kappa = 0 \). The converse is also true: the curvature is only zero in an inflection point of the curve. The torsion is zero if and only if the curve is flat at that point.

Since the Frenet frame is an orthonormal basis, the derivative of each of the vectors \( t, n \) and \( b \) is a linear combination of the other two. These relations are called the Frenet formulas:

\[
\frac{dt}{ds} = \kappa n,
\]

\[
\frac{dn}{ds} = -\kappa t + \tau b,
\]

\[
\frac{db}{ds} = -\tau n.
\]
The Frenet formulas determine the sign of $\tau$.

We will conclude this section on curves with the definition of a number of tensor fields. First we have to adopt some notations: the first and second derivative $\frac{dX}{dt}$ and $\frac{d^2X}{dt^2}$ of the space curve $K$ are denoted as $\partial X$ and $\partial^2 X$. The first and second derivatives of a scalar function $f$ are denoted similarly as $\partial f$ and $\partial^2 f$. The vector $\partial X$ forms a basis of the tangent line $T_X(K)$. The $(1 \times 1)$ Gram matrix $G$ with respect to this basis is denoted as $G = [g] = [(\partial X, \partial X)]$ and its inverse is given by $G^{-1} = [g^{-1}] = \left[ \frac{1}{g} \right]$.

The second fundamental form

The second derivative vector $\partial^2 X$ can be expressed in the Frenet frame $\{t, n, b\}$ as follows:

$$\partial^2 X = \Gamma t + h n,$$

where

$$h = \left( \partial^2 X, n \right),$$

$$\Gamma = g^{-1} \left( \partial^2 X, \partial X \right).$$

It can be shown that the function $h$ is the component of a covariant 2-tensor field, and hence independent of the parametrization of $K$. This tensor field is called the second fundamental form. To the 2-tensor $h$ corresponds a linear mapping on the tangent line $T_X(K)$ with the matrix representation $[g^{-1}h]$. This mapping is called the Weingarten map. It can be proven that $g^{-1}h$ is equal to the curvature $\kappa$.

The covariant derivative

A tangential vector field on a curve $K$ is a vector field $v$ such that $v(t)$ is an element of the tangent line $T_{X(t)}(K)$ for all $t$. Now let $v$ be such that $v = v\partial X$, then we define

$$\nabla v = \partial v + v\Gamma.$$  \hfill (2.4)

The function $\nabla v$, given by (2.4), forms the component of a $(1)$-tensor field on $K$. This tensor field is called the covariant derivative from $v$ on $K$.

The gradient

Let $f$ be a scalar field on the space curve $K$. The gradient field of $f$ is a covariant tensor field (see section 2.3). The gradient with respect to the basis $\{\partial X\}$ of the tangent line is given by

$$\text{grad}(f) = g^{-1} \partial f \partial X.$$

Since it is a tensor field, the gradient of $f$ is independent of the parametrization of $K$. 

2.4 DIFFERENTIAL GEOMETRY

The Hessian

Let $f$ be a scalar field on the space curve $K$. The Hessian of $f$ is defined as the covariant derivative of the gradient field $\nabla f$. We will now derive an expression for the component Hess($f$) of the Hessian. Let the gradient be given by $\nabla f = g^{-1} \partial f \partial X$. Then

$$\text{Hess}(f) = \nabla \left( g^{-1} \partial f \right)$$
$$= \partial (g^{-1} \partial f) + g^{-1} \partial f \Gamma$$
$$= g^{-1} \partial^2 f - 2g^{-1} \partial f \Gamma + g^{-1} \partial f \Gamma$$
$$= g^{-1} (\partial^2 f - \partial f \Gamma).$$

Properties of the gradient and Hessian

If we apply the gradient and the Hessian to the coordinate functions $x^i$ of a space curve $K$, we find the following properties:

$$\sum_{i=1}^{3} (\text{grad}(x^i), \text{grad}(x^i)) = 1,$$
$$\sum_{i=1}^{3} \text{Hess}(x^i)^2 = \kappa^2.$$

**Proof.** Let $f$ be a scalar field on the space curve $K$. Then,

$$(\text{grad}(f), \text{grad}(f)) = \left( g^{-1} \partial f \partial X, g^{-1} \partial f \partial X \right)$$
$$= (g^{-1})^2 (\partial f)^2 (\partial X, \partial X)$$
$$= g^{-1} (\partial f)^2.$$

Therefore

$$\sum_{i=1}^{3} (\text{grad}(x^i), \text{grad}(x^i)) = \sum_{i=1}^{3} g^{-1} \left( \partial x^i \right)^2$$
$$= \sum_{i=1}^{3} g^{-1} \left( \partial X, E^i \right)^2$$
$$= g^{-1} (\partial X, \partial X)$$
$$= 1,$$

which proves the first property. The second property is proven as follows:

$$\text{Hess}(x^i) = g^{-1} (\partial^2 x^i - \partial x^i \Gamma)$$
$$= g^{-1} (\partial^2 X - \Gamma \partial X, E^i)$$
$$= g^{-1} h(n, E^i),$$
and thus
\[ \sum_{i=1}^{3} \text{Hess}(x^i)^2 = \left( g^{-1} h \right)^2 = \kappa^2. \]

2.4.2 Surfaces

Let \( \{ E_i \} \) be the standard basis of \( \mathbb{R}^3 \), such that any point \( X \in \mathbb{R}^3 \) can be written as \( X = x^i E_i \). Now let \( x^i \) be real valued functions from two real parameters \( u^1 \) and \( u^2 \), where \( (u^1, u^2) \in \Omega \subset \mathbb{R}^2 \), with \( \Omega \) open. We assume that the functions \( x^i \) are sufficiently smooth with respect to both variables. Furthermore we assume that the matrix formed by the partial derivatives \( \frac{\partial x^i}{\partial \sigma^j} \) has rank two. Therefore the vectors \( \partial_1X \) and \( \partial_2X \) are linearly independent in every point \( X \) (with \( \partial_j \) shorthand for \( \frac{\partial}{\partial \sigma^j} \)).

Definition of a surface

A surface \( S \) in \( \mathbb{R}^3 \) is the set of points \( X = X(u^1, u^2) = x^i(u^1, u^2)E_i \), where \( (u^1, u^2) \in \Omega \).

The functions \( x^i = x^i(u^j) \) are called a parametrization of the surface \( S \). There are infinitely many parametrizations to describe a surface \( S \). By the substitution \( u^1 = u^1(u^1', u^2') \), \( u^2 = u^2(u^1', u^2') \), where we assume that the determinant of the matrix \( \left[ \frac{\partial x^j}{\partial \sigma^i} \right] \) is nonzero, a new parametrization of the surface \( S \) is obtained. A parameter curve is a curve on the surface that can be described by keeping one of the parameters \( u^i \) constant. Since the determinant of \( \left[ \frac{\partial x^j}{\partial \sigma^i} \right] \) is nonzero, two parameter curves of the form \( u^1 = \text{const} \) and \( u^2 = \text{const} \) will never coincide.

The tangent plane

Let \( S \) be a surface in \( \mathbb{R}^3 \) with coordinates \( u^i \). A space curve \( K \) on the surface \( S \) can be described by \( u^i = u^i(t) \), where \( t \) is a parameter for \( K \). The tangent vector in a point \( X \) on this curve is given by
\[ \frac{dX}{dt} = \frac{dw^i}{dt} \partial_i X, \]
and forms thus a linear combination of the vectors \( \partial_1X \) and \( \partial_2X \). The tangent lines in \( X \) at all curves through \( X \) on the surface lie in a plane. This plane is called the tangent plane in \( X \) at \( S \), and is denoted as \( T_X(S) \). This tangent plane is a 2-dimensional subspace of the tangent space \( T_X(\mathbb{R}^3) \). The vectors \( \partial_1X \) and \( \partial_2X \) form, in a natural way, a basis of this subspace.

We will now introduce a special local basis, linked to a point \( X(u^1, u^2) \) of a surface \( S \). This basis plays the same role as the Frenet frame does for curves. Let \( N_X \) be the vector that is at \( X \) perpendicular to \( T_X(S) \), has length 1, and points in the direction of \( \partial_1X \times \partial_2X \). This vector is called the normal vector in \( X \) at the surface \( S \). The vectors \( \partial_1X \), \( \partial_2X \) and \( N_X \) form a basis of \( T_X(\mathbb{R}^3) \), see figure 2.2. Note that this basis depends on the parametrization of \( S \), as opposed to the Frenet frame for curves, which is a geometric property.
2.4 Differential Geometry

Like in the section on space curves, we will now define some important tensor fields: the first and second fundamental tensor field, and the gradient and Hessian of a scalar field on the surface $S$.

The first fundamental tensor field

The first fundamental tensor field is the tensor field that attaches to each point $X$ the inner product $(\cdot, \cdot)_X$ on $T_X(S)$. The components of the first fundamental tensor field with respect to the basis $\{\partial_1X, \partial_2X\}$ are given by

$$g_{ij} = (\partial_iX, \partial_jX)_X.$$

The parameter curves of the surface $S$ intersect each other orthogonally if $g_{12} = 0$. For this reason, a parametrization $u^i$ is called orthogonal if $g_{12} = 0$. If the matrix $[g_{ij}]$ is the unit matrix, the parametrization is called isometric. For a surface $S$, always an orthogonal parametrization can be found. An isometric parametrization, however, does only exist for developable surfaces, which is a well-known class of surfaces in differential geometry.

The second fundamental tensor field

Let $S$ be a surface in $\mathbb{R}^3$ with parametrization $x^i = x^i(u^j)$. In a point $X$ the second derivative vectors $\partial_i\partial_jX$ can be expressed in the basis $\{\partial_1X, \partial_2X, N_X\}$ of the tangent space $T_X(\mathbb{R}^3)$. We denote this as follows:

$$\partial_i\partial_jX = \Gamma^k_{ij}\partial_kX + h_{ij}N_X.$$  \hfill (2.5)

The functions $\Gamma^k_{ij}$ are called Christoffel symbols (see also section 2.3). The functions $h_{ij}$ are the components of a covariant 2-tensor field, called the second fundamental form. Some properties of $\Gamma^k_{ij}$ and $h_{ij}$ are the following:

$$h_{ij} = (\partial_i\partial_jX, N_X),$$

$$\Gamma^k_{ij} = g^{kl}(\partial_i\partial_jX, \partial_kX),$$

$$\Gamma^k_{ij} = \frac{1}{2}g^{kl}(\partial_jg_{il} + \partial_lg_{ij} - \partial_ig_{lj}),$$

$$\partial_k g^{ij} = -g^{ij}\Gamma^k_{ik} - g^{lj}\Gamma^k_{il}. \hfill (2.6)$$

Analogous to curves (section 2.4.1), to the 2-tensor $h$ corresponds a linear mapping on the tangent space $T_X(S)$ with the matrix representation $[g^{ik}h_{kj}]$. This mapping is called the Weingarten map. The eigenvalues $\kappa_1$ and $\kappa_2$ of this matrix are called the principal curvatures of the surface $S$ at the point $X$. Often the following related quantities are used: the Gaussian curvature $\kappa_1\kappa_2$, the mean curvature $(\kappa_1 + \kappa_2)/2$ and the total curvature $\kappa_1^2 + \kappa_2^2$. Note that these quantities can be defined as $\det(A), \frac{1}{2}\text{trace}(A)$ and $\text{trace}(A^2)$ respectively, where $A$ is the matrix $[g^{ik}h_{kj}]$. 

CHAPTER 2 MATHEMATICAL BACKGROUND

The covariant derivative

A \textit{tangential vector field} is a vector field \( \mathbf{v} \) such that \( \mathbf{v}(u^1, u^2) \) is an element of the tangent space \( T_{X(u^1, u^2)}(S) \) for all \( (u^1, u^2) \in \Omega \). Now let \( \mathbf{v} = v^j \partial_j X \) be a tangential vector field, defined on the surface \( S \). Then we define

\[
\nabla_k v^j = \partial_k v^j + v^l \Gamma^j_{kl}
\]

(2.7)

The functions \( \nabla_k v^j \), given by (2.7) form the components of a \( \text{\(1\)}{\text{\(1\)}} \)-tensor field on \( S \). This tensor field is called the \textit{covariant derivative} from \( \mathbf{v} \) on \( S \).

The gradient

Let \( f \) be a scalar field on the surface \( S \). The gradient field of \( f \) is a covariant tensor field (see section 2.3). With respect to the basis \( \{ \partial_1 X, \partial_2 X \} \) of the tangent plane, the gradient is given by

\[
\text{grad}(f) = g^{ij} \partial_i f \partial_j X.
\]

The Hessian

Let \( f \) be a scalar field on \( S \). The covariant derivative of the gradient field \( \text{grad}(f) \) on \( S \) is a \( \text{\(1\)}{\text{\(1\)}} \)-tensor field called the \textit{Hessian} of \( f \). We will now derive an expression for the components \( \text{Hess}(f)^i_j \) of the Hessian. Let the gradient of \( f \) be given by \( \text{grad}(f) = g^{ij} \partial_i f \partial_j X \). Then,

\[
\begin{align*}
\text{Hess}(f)^i_j &= \nabla_k (g^{ij} \partial_k f) \\
&= \partial_k (g^{ij} \partial_k f) + g^{ij} \partial_k f \Gamma^i_{kl} \\
&= g^{ij} \partial_k \partial_k f + \partial_k f \left( \partial_k g^{ij} + g^{il} \Gamma^j_{kl} \right) \\
\{ \text{property (2.6)} \} &= g^{ij} \partial_k \partial_k f + \partial_k f \left( -g^{il} \Gamma^j_{lk} - g^{jl} \Gamma^i_{lk} + g^{il} \Gamma^j_{kl} \right) \\
&= g^{ij} \partial_k \partial_k f - g^{ij} \partial_k f \Gamma^i_{lk} \\
&= g^{ij} \left( \partial_k \partial_k f - \partial_k f \Gamma^i_{lk} \right)
\end{align*}
\]

Properties of gradient and Hessian

If we apply the gradient and the Hessian to the coordinate functions \( x^i \) of a surface \( S \), we find the following properties:

\[
\begin{align*}
\sum_{i=1}^{3} \text{grad}(x^i) \cdot \text{grad}(x^i) &= 2, \\
\sum_{i=1}^{3} \text{trace} \left( [\text{Hess}(x^i)]^i_j \right) &= \kappa_1^2 + \kappa_2^2
\end{align*}
\]
2.4 DIFFERENTIAL GEOMETRY

Proof. Let $f$ be a scalar function defined on $S$. Then,
\[
\text{grad}(f) \cdot \text{grad}(f) = \left( \partial_i f g^{ij} \partial_j X \right) \cdot \left( \partial_k f g^{kl} \partial_l X \right)
= \partial_i f \partial_k f g^{ij} \left( \partial_j X \cdot \partial_l X \right) g^{kl}
= \partial_i f \partial_k f g^{ij} g_{jl} g^{kl}
= g^{ki} \partial_i f \partial_k f.
\]

Therefore,
\[
\sum_{\alpha=1}^{3} \text{grad}(x^\alpha) \cdot \text{grad}(x^\alpha) = \sum_{\alpha=1}^{3} g^{ki} \partial_k x^\alpha \partial_i x^\alpha
= \sum_{\alpha=1}^{3} g^{ki} (\partial_k X \cdot E_\alpha) (\partial_i X \cdot E_\alpha)
= g^{ki} (\partial_k X \cdot \partial_i X)
= g^{ki} g_{ik}
= 2. \tag{2.8}
\]

The second property is proven as follows:
\[
\text{Hess}(x^\alpha)^j_k = g^{ij} (\partial_k \partial_i x^\alpha - \partial_k x^\alpha \Gamma^i_{ik})
= g^{ij} (\partial_k \partial_i X - \Gamma^i_{ik} \partial_j X, E^\alpha)
\{\text{property (2.5)}\} = g^{ij} h_{kl} (N_X, E^\alpha).
\]

Therefore,
\[
\sum_{\alpha=1}^{3} \text{trace} \left( [\text{Hess}(x^\alpha)^j_k]^2 \right) = \text{trace} \left( \sum_{\alpha=1}^{3} [\text{Hess}(x^\alpha)^j_k]^2 \right)
= \text{trace} \left( \sum_{\alpha=1}^{3} [g^{ij} h_{kl}]^2 (N_X, E^\alpha)^2 \right)
= \text{trace} \left( [g^{ij} h_{kl}]^2 \right)
= \kappa_1^2 + \kappa_2^2.
\]

\[\blacksquare\]

2.4.3 Space curves in conventional notation

In texts on geometric modeling, the notations from tensor analysis are rarely used. From now on we will proceed with a more conventional notation, that is comparable to the notation used in [Far93]. This notation will also be used in the rest of this dissertation! Also the most important results for curves are reformulated in this notation.
Let $I \subset \mathbb{R}$ be an interval. A space curve $K$ is a smooth map $\mathbf{x} : I \to \mathbb{R}^3$, $(t) \mapsto \mathbf{x}(t) = (x_1(t), x_2(t), x_3(t))$. The derivative of $\mathbf{x}$ with respect to the arc length parameter $s$ is from now on denoted with a prime ($'\mathbf{x}$') and the derivative of $\mathbf{x}$ with respect to an arbitrary parameter $t$ with a dot ($\dot{\mathbf{x}}$). We assume that the curve $\mathbf{x}$ is regular, i.e. the derivative vector $\dot{\mathbf{x}}(t)$ is nonzero for all $t \in I$.

The Frenet frame $\{\mathbf{t}, \mathbf{n}, \mathbf{b}\}$ has the following representation with respect to an arbitrary parameter $t$:

$$
\mathbf{t} = \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||}, \quad \mathbf{b} = \frac{\dot{\mathbf{x}} \times \ddot{\mathbf{x}}}{||\dot{\mathbf{x}} \times \ddot{\mathbf{x}}||}, \quad \mathbf{n} = \mathbf{b} \times \mathbf{t}.
$$

The curvature $\kappa$ and the torsion $\tau$ in a point of a curve $\mathbf{x}$ may be defined both in terms of the arc length parameter $s$ and in terms of an arbitrary parameter $t$:

$$
\kappa(s) = ||\mathbf{x}''||, \quad \tau(s) = \frac{\det[\mathbf{x}', \mathbf{x}'', \mathbf{x}''']}{\kappa^2},
$$

$$
\kappa(t) = \frac{||\dot{\mathbf{x}} \times \ddot{\mathbf{x}}||}{||\dot{\mathbf{x}}||^3}, \quad \tau(t) = \frac{\det[\dot{\mathbf{x}}, \ddot{\mathbf{x}}, \frac{d^2\mathbf{x}}{dt^2}]}{||\dot{\mathbf{x}} \times \ddot{\mathbf{x}}||^2}.
$$

(2.9)

See figure 2.1 for an illustration of the Frenet frame and the curvature.

![Frenet frame illustration](image)

Figure 2.1: The Frenet frame $\{\mathbf{t}, \mathbf{n}, \mathbf{b}\}$ is a local orthonormal basis at each point of the curve $\mathbf{x}$. The osculation circle is the circle with radius $\rho = 1/\kappa$ that lies in the plane spanned by $\mathbf{t}$ and $\mathbf{n}$. This circle is locally a ‘second order’ approximation of the curve.

Let $f : I \to \mathbb{R}$ be a scalar function defined on the curve $\mathbf{x}$. The gradient of $f$ is the smooth tangential vector field $\text{grad}(f)$ on $\mathbf{x}$ given by

$$
\text{grad}_{\mathbf{x}}(f) = \frac{\dot{f}}{\dot{\mathbf{x}}} \cdot \mathbf{t}.
$$
2.4 DIFFERENTIAL GEOMETRY

The Hessian of \( f \) is a bilinear function on the tangent space \( T_x(K) \). It has the following representation with respect to the basis \( \{ \hat{x} \} \) of \( T_x(K) \):

\[
\text{Hess}_x(f) = \frac{\partial^2 f}{\partial \hat{x} \cdot \hat{x}} = \frac{\partial^2 f}{(\hat{x} \cdot \hat{x})^2},
\]

(2.10)

Let \( y : I \rightarrow \mathbb{R}^3 \) be another space curve defined on the same interval \( I \). The Laplace operator \( \triangle(\cdot) \) is defined as (see [Adl93])

\[
\triangle_x(y) = \frac{\ddot{\hat{y}}}{\hat{x} \cdot \hat{x}} - \frac{(\hat{x} \cdot \hat{x})\ddot{\hat{y}}}{(\hat{x} \cdot \hat{x})^2}.
\]

(2.11)

Some important properties of the gradient, Hessian and Laplace operator are

\[
\sum_{i=1}^{3} \text{grad}_x(x_i) \cdot \text{grad}_x(x_i) = 1,
\]

(2.12)

\[
\sum_{i=1}^{3} \text{Hess}_x(x_i)^2 = \kappa^2,
\]

(2.13)

\[
\triangle_x(\hat{\kappa}) \cdot \triangle_x(\hat{x}) = \kappa^2.
\]

(2.14)

The first and second property were proven in the previous section and the last property follows straightforwardly from a comparison between (2.10) and (2.11).

2.4.4 Surfaces in conventional notation

Just like for curves, we will introduce a more conventional notation for surfaces, that is used in the rest of this dissertation. Also the most important results for surfaces are reformulated in this notation.

Let \( \Omega \) be an open connected subset of \( \mathbb{R}^2 \). A parametric surface \( S \) in \( \mathbb{R}^3 \) is a smooth map \( F : \Omega \rightarrow \mathbb{R}^3 \), \((u, v) \mapsto F(u, v) = (F_1(u, v), F_2(u, v), F_3(u, v)) \). The surface \( S \) is called regular if the partial derivative vectors \( F_u \) and \( F_v \) are linearly independent for all \((u, v) \in \Omega \). To avoid potential problems concerning the parametrization of the surface, we shall assume in the sequel that the surface \( F \) is always regular.

The tangent space \( T_F(S) \) in a regular point \( F(u, v) \) of the surface is the vector space spanned by the vectors \( F_u \) and \( F_v \). The unit normal \( N_F \) is given by \( N_F = F_u \times F_v / \|F_u \times F_v\| \).

At the point \( F(u, v) \) the frame with origin \( F \) and axes \( F_u, F_v \) and \( N_F \) forms a local basis at the point \( F(u, v) \) (see figure 2.2).

The first and second fundamental forms are two bilinear functions on the tangent space \( T_F(S) \) that can be used to describe the local geometry in the neighborhood of a point \( F(u, v) \).

The matrix representation of the first fundamental form with respect to the basis \( \{ F_u, F_v \} \) of the tangent space is given by

\[
I_F = \begin{bmatrix}
F_u \cdot F_u & F_u \cdot F_v \\
F_v \cdot F_u & F_v \cdot F_v
\end{bmatrix}.
\]
Figure 2.2. In a regular point of a surface $\mathbf{F}$, a local basis can be formed by the tangent vectors $\mathbf{F}_u, \mathbf{F}_v$ and the normal vector $\mathbf{N}_F$.

The matrix representation of the second fundamental form with respect to the basis $\{\mathbf{F}_u, \mathbf{F}_v\}$ of the tangent space is given by

$$\mathbb{II}_F = \begin{pmatrix} \mathbf{N}_F \cdot \mathbf{F}_{uu} & \mathbf{N}_F \cdot \mathbf{F}_{uv} \\ \mathbf{N}_F \cdot \mathbf{F}_{vu} & \mathbf{N}_F \cdot \mathbf{F}_{vv} \end{pmatrix}.$$ 

The principal curvatures $\kappa_1$ and $\kappa_2$ are the eigenvalues of the matrix $I_F^{-1} \mathbb{II}_F$. The corresponding unit eigenvectors in the $(u, v)$ parameter space are called the principal directions (see also [Far93]).

Let $f : \Omega \to \mathbb{R}$ be a scalar function defined on the surface $\mathbf{F}$. The gradient field of $f$ is a vector field on the surface $S$, with the following interpretation:

- the gradient in the point $\mathbf{F}(u, v)$ is the vector in the tangent space $T_F(S)$ pointing in the direction of steepest ascent

- the length of the gradient is the directional derivative in the corresponding parametric direction.

The Hessian of $f$ is a mapping that attaches to each point $\mathbf{F}(u, v)$ of $S$ a bilinear function on the tangent space $T_F(S)$. Just like the Hessian of an ordinary function from $\mathbb{R}^n$ to $\mathbb{R}$, this bilinear function can be viewed as a quadratic approximation of the function $f$.

If we apply the gradient and Hessian to the components $F_i$ of a surface $\mathbf{F}$, the following
2.4 DIFFERENTIAL GEOMETRY

properties are found:

\[
\sum_{i=1}^{3} \text{grad}_F(F_i)^T \text{grad}_F(F_i) = 2,
\]

\[
\sum_{i=1}^{3} \text{trace}(\text{Hess}_F(F_i)) = \text{trace}\left( (I_F^{-1}I_F)^2 \right) = \kappa_1^2 + \kappa_2^2, \tag{215}
\]

\[
\sum_{i=1}^{3} \det(\text{Hess}_F(F_i)) = \det( (I_F^{-1}I_F)^2 ) = \kappa_1\kappa_2,
\]

\[
\sum_{i=1}^{3} \text{trace}(\text{Hess}_F(F_i)^2) = \text{trace}\left( (I_F^{-1}I_F)^2 \right)^2 = (\kappa_1 + \kappa_2)^2.
\]

2.4.5 Smoothness integrals for curves

The goal of variational curve modeling is to find smooth curves. The length of a curve

\[
F_1(x) = \int_0^L \| \dot{x} \| \, dt
\]

and the integral of the squared curvature

\[
F_2(x) = \int_0^L \kappa^2 \| \dot{x} \| \, dt
\]

are two integrals that are often used to measure the smoothness of a curve \( x \), defined on the interval \([0, L]\) (see also section 3.3). The integrals \( F_1(x) \) and \( F_2(x) \) are both geometric properties of the curve \( x \), i.e. they are invariant under a parameter transformation \( \varphi \):

\[
F_i(x \circ \varphi) = F_i(x), \quad i = 1, 2
\]

2.4.6 Approximations of smoothness integrals

Unfortunately, the smoothness integrals (2.16) and (2.17) are too complicated for most practical applications. In this section we will define data dependent approximations to these integrals. A data dependent approximation for the integral \( F_1(x) \) is an approximation that depends on a curve \( c \). The idea behind this is the following: suppose we have a curve \( c \) that is 'close' to the curve \( x \). Then we can use this curve \( c \) to construct approximations of the integrals \( F_1(x) \) and \( F_2(x) \) that are less complicated expressions in \( x \).

Let \( M \) be the set of \( G^1 \) or tangent continuous curves in \( \mathbb{R}^3 \) with piecewise continuous curvature, defined on the interval \([0, L]\). Let \( x \) and \( c \) be two curves in \( M \). Then we define the data dependent approximation \( \hat{F}_1(x; c) \) of \( F_1(x) \) as

\[
\hat{F}_1(x; c) = \int_0^L \sum_{i=1}^{3} \text{grad}_c(x_i) \cdot \text{grad}_c(x_i) \| \dot{c} \| \, dt
\]

\[
= \int_0^L \frac{\dot{x} \cdot \dot{x}}{\dot{c} \cdot \dot{c}} \| \dot{c} \| \, dt
\]
and the data dependent approximation $\hat{F}_2(x; c)$ of $F_2(x)$ as

$$\hat{F}_2(x; c) = \int_0^L \sum_{i=1}^3 \text{Hess}_c(x_i)^2 \|\hat{c}\| dt.$$  

The latter integral can be more conveniently expressed in the Laplace operator $\Delta$:

$$\hat{F}_2(x; c) = \int_0^L \Delta_c(x) \cdot \Delta_c(x) \|\hat{c}\| dt.$$  

The approximations $\hat{F}_1$ and $\hat{F}_2$ have the following properties:

- They are exact for $x = c$, i.e. $\hat{F}_i(c; c) = F_i(c)$, $i = 1, 2$.
- They are invariant under a parameter transformation $\varphi$ of $x$ and $c$:

$$\hat{F}_i(x \circ \varphi; c \circ \varphi) = \hat{F}_i(x; c), \quad i = 1, 2.$$  

The first property follows directly from the properties (2.12) and (2.13) of the gradient and the Hessian. The second property is a consequence of the fact that the gradient and the Hessian are independent of the parametrization of a curve, since they were defined as tensors.

A desirable property of the approximations $\hat{F}_i$, $i = 1, 2$ is that the error $\left| \hat{F}_i(x; c) - F_i(x) \right|$ is small when the distance between $x$ and $c$ is small. To measure the distance between $x$ and $c$, we need to define a suitable norm. This norm will depend on $c$. Let the inner product $\langle \cdot, \cdot \rangle_c$ be defined as

$$\langle x, y \rangle_c = x(0) \cdot y(0) + x(L) \cdot y(L) + \int_0^L \Delta_c(x) \cdot \Delta_c(y) \|\hat{c}\| dt.$$  

This inner product induces a norm on $M$, given by

$$\|x\|_c = \sqrt{\langle x, x \rangle_c}.$$  

The extra terms $x(0) \cdot y(0)$ and $x(L) \cdot y(L)$ in (2.18) are introduced to guarantee that for two curves $x_0$ and $x_1$ the relation $\|x_0 - x_1\|_c = 0$ implies $x_0 = x_1$. Another way to enforce this property is to always fix the end points of the curve $x$ to the end points of $c$. However, this would be an unnecessary restriction to the function space $M$.

For the integrals $\hat{F}_i(x)$ and their approximations $\hat{F}_i(x; c)$, the norm $\|x\|_c$ is invariant under a parameter transformation $\varphi$ of $x$ and $c$, i.e. $\|x \circ \varphi\|_{c \circ \varphi} = \|x\|_c$. We may therefore assume, without losing generality, that the curve $c$ is arc length parametrized. Therefore, from now on we will assume that $\hat{c} \cdot \hat{c} = 1$ (and hence $\hat{c} \cdot \hat{c} = 0$).

Under this assumption, the expressions $\|x\|_c$, $\hat{F}_1(x; c)$ and $\hat{F}_2(x; c)$ can be simplified to

$$\|x\|_c = x(0) \cdot x(0) + x(L) \cdot x(L) + \int_0^L \hat{x} \cdot \hat{x} dt,$$

$$\hat{F}_1(x; c) = \int_0^L \hat{x} \cdot \hat{x} dt,$$

$$\hat{F}_2(x; c) = \int_0^L \hat{x} \cdot \hat{x} dt.$$
2.4 DIFFERENTIAL GEOMETRY

Now we will use the norm \( \| \cdot \|_c \) to compare \( F_1 \) and \( F_2 \) with their approximations \( \hat{F}_1 \) and \( \hat{F}_2 \). First a lemma is derived that is needed for this comparison.

**Lemma 3** Let \( y = y(x) \), \( x \in [0, L] \), be a twice differentiable function with second derivatives that are \( L_2 \)-integrable on the interval \( [0, L] \). Then we can make the following estimates for \( y \) in an arbitrary parameter value \( a \in [0, L] \):

\[
|y(a)| \leq C_0 \left\{ |y(0)|^2 + |y(L)|^2 + \int_0^L \bar{y}(s)^2 ds \right\}^{1/2},
\]
\[
|\dot{y}(a)| \leq C_1 \left\{ |y(0)|^2 + |y(L)|^2 + \int_0^L \bar{y}(s)^2 ds \right\}^{1/2},
\]

where \( C_0 = 2 + L \sqrt{L} \) and \( C_1 = \frac{2}{L} + \sqrt{L} \).

**Proof.** The value \( y(x) \) can be expressed as

\[
y(x) = \frac{\beta - \alpha}{L}x + \alpha + \frac{x - L}{L} \int_0^x \bar{y}(s) ds + \frac{x}{L} \int_x^L (s - L) \bar{y}(s) ds.
\]

Therefore,

\[
|y(a)| = \left| \frac{\beta - \alpha}{L}a + \alpha + \frac{a - L}{L} \int_0^a \bar{y}(s) ds + \frac{a}{L} \int_a^L (s - L) \bar{y}(s) ds \right|
\]
\[
\leq |\beta - \alpha| \frac{a}{L} + |\alpha| + \frac{a - L}{L} \int_0^a L |\bar{y}(s)| ds + \frac{a}{L} \int_a^L L |\bar{y}(s)| ds
\]
\[
\leq |\beta - \alpha| + |\alpha| + L \sqrt{L} \int_0^L |\bar{y}(s)|^2 ds
\]
\[
\leq (1 + 1 + L \sqrt{L}) \left\{ |y(0)|^2 + |y(L)|^2 + \int_0^L \bar{y}(s)^2 ds \right\}^{1/2}
\]

and

\[
|\dot{y}(a)| = \left| \frac{\beta - \alpha}{L} + \frac{1}{L} \int_0^a \bar{y}(s) ds + \frac{1}{L} \int_a^L (s - L) \bar{y}(s) ds \right|
\]
\[
\leq |\beta - \alpha| \frac{1}{L} + \frac{1}{L} \int_0^a L |\bar{y}(s)| ds + \frac{1}{L} \int_a^L L |\bar{y}(s)| ds
\]
\[
\leq \frac{|\alpha| + |\beta|}{L} + \int_0^L |\bar{y}(s)| ds
\]
\[
\leq \frac{|\alpha|}{L} + \frac{|\beta|}{L} + \sqrt{L} \int_0^L |\bar{y}(s)|^2 ds
\]
\[
\leq (\frac{1}{L} + \frac{1}{L} + \sqrt{L}) \left\{ |y(0)|^2 + |y(L)|^2 + \int_0^L \bar{y}(s)^2 ds \right\}^{1/2}
\]
Corollary 4  For a curve $\mathbf{x}$ defined on the interval $[0, L]$, the following holds:
\[
\|\dot{\mathbf{x}}(a)\| \leq \left(\frac{2}{L} + \sqrt{L}\right)\|\mathbf{x}\|_c, \quad a \in [0, L],
\]
where $L$ is the length of the curve $\mathbf{c}$.

In the following proposition the errors $\left| \hat{F}_1(\mathbf{c} + \xi; \mathbf{c}) - F_1(\mathbf{c} + \xi) \right|$ are expressed in the norm $\|\xi\|_c$.

Proposition 5  The functionals $\hat{F}_1$ and $\hat{F}_2$ satisfy the following relations:
\[
\left| \hat{F}_1(\mathbf{c} + \xi; \mathbf{c}) - F_1(\mathbf{c} + \xi) \right| \leq K_1 \|\xi\|_c
\]
\[
\left| \hat{F}_2(\mathbf{c} + \xi; \mathbf{c}) - F_2(\mathbf{c} + \xi) \right| \leq K_2 \|\xi\|_c
\]
for small values of $\|\xi\|_c$, with $K_1$ and $K_2$ constants that depend on $\mathbf{c}$, but not on $\xi$.

Proof.  We will assume that the curve $\mathbf{c}$ is arc length parametrized. Let $\mathbf{x} = \mathbf{c} + \xi$, then
\[
\left| \hat{F}_1(\mathbf{x}; \mathbf{c}) - F_1(\mathbf{x}) \right| = \left| \hat{F}_1(\mathbf{x}; \mathbf{c}) - \hat{F}_1(\mathbf{x}; \mathbf{x}) \right|
\]
\[
= \int_0^L \ddot{x} \cdot \ddot{x} - \sqrt{x} \cdot \ddot{x} dt
\]
\[
= \int_0^L 1 + h - \sqrt{1 + h} dt,
\]
where $h$ is given by
\[
h = \ddot{x} \cdot \ddot{x} - \dot{\mathbf{c}} \cdot \dot{\mathbf{c}} = (\ddot{\mathbf{c}} + \dddot{\xi}) \cdot (\dddot{\mathbf{c}} + \dddot{\xi}) - \dot{\mathbf{c}} \cdot \dot{\mathbf{c}} = 2\dddot{\mathbf{c}} \cdot \dddot{\xi} + \dddot{\mathbf{c}} \cdot \dddot{\xi}.
\]
From the Taylor expansion
\[
1 + h - \sqrt{1 + h} = \frac{h}{2} + \frac{h^2}{8} - \frac{h^3}{16} + \cdots
\]
we can derive that for any fixed value $K > \frac{1}{2}$
\[
\left| 1 + h - \sqrt{1 + h} \right| \leq K |h|
\]
for small values of $h$. Hence, under the assumption that $h$ is small, we find
\[
\left| \hat{F}_1(\mathbf{x}; \mathbf{c}) - F_1(\mathbf{x}) \right| \leq K \int_0^L \left| 2\dddot{\mathbf{c}} \cdot \dddot{\xi} + \dddot{\mathbf{c}} \cdot \dddot{\xi} \right| dt
\]
\[
\leq 2K \int_0^L \|\dddot{\mathbf{c}}\| \|\dddot{\xi}\| dt + K \int_0^L \|\dddot{\xi}\|^2 dt
\]
\[
\begin{aligned}
\text{Schwarz} & \geq 2K \int_0^L \|\dddot{\mathbf{c}}\| \|\dddot{\xi}\| dt + K \int_0^L \|\dddot{\xi}\|^2 dt \\
\{ \|\dddot{\mathbf{c}}\| = 1 \} & = 2K \int_0^L \dddot{\xi} \ dt + K \int_0^L \|\dddot{\xi}\|^2 dt \\
\text{corollary 4} & \leq 2K \int_0^L \left(\frac{2}{L} + \sqrt{L}\right) \|\xi\|_c dt + K \int_0^L \left(\frac{2}{L} + \sqrt{L}\right)^2 \|\xi\|_c^2 dt \\
& = 2KL \left(\frac{2}{L} + \sqrt{L}\right) \|\xi\|_c + KL \left(\frac{2}{L} + \sqrt{L}\right)^2 \|\xi\|_c^2.
\end{aligned}
\]
2.4 DIFFERENTIAL GEOMETRY

From corollary 4 it follows that the condition that $h$ is small is met when $\|\xi\|_c$ is small. Therefore, the first statement is true for $K_1 > 2 + L \sqrt{L}$. The second statement can be proven analogously:

$$\left|\hat{F}_2(x; c) - F_2(x)\right| = \left|\int_0^L \triangle_c(x) \cdot \triangle_c(x) - \int_0^L \kappa^2 \|\dot{x}\| \, dt\right| = \int_0^L \dot{x} \cdot \ddot{x} - \frac{(\ddot{x} \cdot \ddot{x}) (\dot{x} \cdot \ddot{x}) - (\ddot{x} \cdot \ddot{x})^2}{(\ddot{x} \cdot \ddot{x})^5/2} \, dt = \int_0^L \ddot{c} \cdot \ddot{c} + h_3 \cdot \frac{(\ddot{c} \cdot \ddot{c} + h_3) (1 + h_1) - h_2^2}{(1 + h_1)^5/2} \, dt,$$

where $h_1$, $h_2$ and $h_3$ are given by

$$\begin{align*}
h_1 &= \ddot{x} \cdot \ddot{x} - \dot{\hat{c}} \cdot \hat{c} = (\hat{c} + \hat{\xi}) \cdot (\hat{c} + \hat{\xi}) - \hat{c} \cdot \hat{c} = 2\hat{c} \cdot \hat{\xi} + \hat{c} \cdot \hat{c} = 1 + h_1, \\
h_2 &= \ddot{x} \cdot \ddot{x} = (\hat{c} + \hat{\xi}) \cdot (\hat{c} + \hat{\xi}) = \hat{c} \cdot \hat{c} + \hat{c} \cdot \hat{c} + \hat{c} \cdot \hat{c} = h_2, \\
h_3 &= \ddot{x} \cdot \ddot{x} - \dot{\hat{c}} \cdot \hat{c} = (\hat{c} + \hat{\xi}) \cdot (\hat{c} + \hat{\xi}) - \hat{c} \cdot \hat{c} = 2\hat{c} \cdot \hat{\xi} + \hat{c} \cdot \hat{c} = \hat{c} \cdot \hat{c} + h_3.
\end{align*}$$

From the Taylor expansion

$$\hat{c} \cdot \hat{c} + h_3 \cdot \frac{(\ddot{c} \cdot \ddot{c} + h_3) (1 + h_1) - h_2^2}{(1 + h_1)^5/2} = \frac{3\ddot{c} \cdot \ddot{c}}{2} h_1 + \left(\frac{15\ddot{c} \cdot \ddot{c}}{8} h_1^2 + h_2^2 + \frac{3}{2} h_1 h_3\right) + \cdots$$

we can derive that for any fixed value $K > \frac{3}{2} \max_{t \in [0, L]} (\ddot{c} \cdot \ddot{c})$

$$\left|\hat{c} \cdot \hat{c} + h_3 \cdot \frac{(\ddot{c} \cdot \ddot{c} + h_3) (1 + h_1) - h_2^2}{(1 + h_1)^5/2}\right| \leq K \left|\hat{c} \cdot \hat{c}\right|$$

for small values of $h_1$, $h_2$ and $h_3$. Hence, under the assumption that $h_1$, $h_2$ and $h_3$ are small, we find

$$\left|\hat{F}_2(x; c) - F_2(x)\right| \leq K \int_0^L \left|2\hat{c} \cdot \hat{\xi} + \hat{c} \cdot \hat{\xi}\right| \, dt \leq 2KL \left(\frac{2}{L} + \sqrt{L}\right) \|\xi\|_c + KL \left(\frac{2}{L} + \sqrt{L}\right)^2 \|\xi\|_c^2.$$

From corollary 4 it follows that the condition that ‘$h_1$, $h_2$ and $h_3$ are small’ is met when $\|\xi\|_c$ is small. Therefore, the second statement is true for $K_2 > \left(6 + 3L \sqrt{L}\right) \max_{t \in [0, L]} (\hat{c} \cdot \hat{c})$. □

2.4.7 Iterative minimization of smoothness integrals

We will now describe a technique to iteratively minimize the smoothness integrals $F_1$ and $F_2$. For this technique the data dependent functionals $\hat{F}_i(\cdot; c)$ and $\hat{F}_2(\cdot; c)$ are used. Since the technique is similar to Newton’s method for minimizing a function, we will give a short description of Newton’s method first (see also [Lue89]).
Newton’s method

Suppose that the function \( f \) of a single variable \( x \) is to be minimized, and suppose the point \( x_k \) is given. It is possible to construct a quadratic function \( f_k \) which at \( x_k \) agrees with \( f \) up to second derivatives, that is

\[
f_k(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2.
\]

We may then calculate an estimate \( x_{k+1} \) of the minimum point of \( f \) by computing the minimum of \( f_k \). This process, which is illustrated in figure 2.3, can then be repeated at \( x_{k+1} \). In general, Newton’s method has order two convergence.

![Figure 2.3: Newton’s method for determining a minimum of the function \( f(x) \).](image)

Iterative method

In a completely analogous way we will now define an iterative method for computing a minimum of the integrals \( F_i \). First we have to define the space in which the curves lie. Let \( M \) be the set of tangent continuous curves in \( \mathbb{R}^3 \) with piecewise continuous curvature, defined on the interval \( [0, L] \). Let \( M_1 \) be the set of all curves \( x \) of \( M \) that satisfy the constraints \( x(0) = x_0 \) and \( x(L) = x_L \) and let \( M_2 \) the set of all curves \( x \) of \( M_1 \) that in addition satisfy the constraints \( \dot{x}(0) = t_0 \) and \( \dot{x}(L) = t_L \). Of course, there are also other boundary conditions possible. The two problems can now be stated as:

\[
\min_{x \in M_i} F_i(x).
\]

An iterative method to solve this problem works as follows. Suppose a curve \( x_k \) is given. At \( x_k \) we can construct the data dependent approximation \( \hat{F}_i(\cdot; x_k) \) of \( F_i \). We define the new
2.4 DIFFERENTIAL GEOMETRY

estimate \( x_{k+1} \) of the minimum of the functional \( F_i \) as the minimum of the function \( F_i(\cdot; x_k) \). The question is now: does this iterative method converge? (under proper start conditions). Below, we will try to answer this question.

Euler-Lagrange equations

Let \( F \) be the functional

\[
F(t, x, \dot{x}, \ddot{x}) = \int_0^L f(t, x, \dot{x}, \ddot{x}) dt.
\]

A necessary condition for \( x = x(t) \) to minimize the functional \( J \) is that the components \( x_i \) of \( x \) satisfy the Euler Lagrange differential equations

\[
\frac{\partial f}{\partial x_i} - \frac{d}{dt} \frac{\partial f}{\partial \dot{x}_i} + \frac{d^2}{dt^2} \frac{\partial^2 f}{\partial \dot{x}_i^2} = 0,
\]

where \( x \) has fixed values and derivatives at the end-points.

The integrals \( F_1, F_2, \hat{F}_1 \) and \( \hat{F}_2 \) can be written in the form (2.21) with the following choices for the function \( f \):

\[
\begin{align*}
\hat{f}_1(x; \mathbf{c}) &= \frac{(\ddot{x} \cdot \dot{x})^{1/2}}, \\
\hat{f}_2(x; \mathbf{c}) &= \frac{(\ddot{x} \cdot \dot{x})}{(\hat{c} \cdot \hat{c})^{3/2}}, \\
\hat{f}_1(x; \mathbf{c}) &= \frac{(\ddot{x} \cdot \dot{x})}{(\hat{c} \cdot \hat{c})^{3/2}}, \quad \hat{f}_2(x; \mathbf{c}) = \frac{-(\ddot{x} \cdot \dot{x})(\hat{c} \cdot \hat{c})}{(\hat{c} \cdot \hat{c})^{3/2}} + \frac{(\ddot{x} \cdot \dot{x})(\hat{c} \cdot \hat{c})^2}{(\hat{c} \cdot \hat{c})^{3/2}}. 
\end{align*}
\]

Necessary conditions

A necessary condition for the convergence of the iterative method is the following: if the \( k \)-th iteration \( x_k \) is a minimum of the exact functional \( F_i \), then the next iteration \( x_{k+1} \) should be the same \( (x_{k+1} = x_k) \). This means that the functional \( \hat{F}_i \) should have the following property: suppose the curve \( x = \mathbf{c} \) is a minimum of \( F_i(\mathbf{x}) \), then this curve \( \mathbf{c} \) should also be a minimum of the data dependent approximation \( \hat{F}_i(\mathbf{x}; \mathbf{c}) \).

For the case \( i = 1 \) we can prove that this condition is met using the Euler Lagrange equations. The Euler Lagrange equations (2.22) for \( F_1 \) and \( \hat{F}_1 \) are given by

\[
\begin{align*}
\frac{d}{dt} \frac{\partial f_1}{\partial \dot{x}_i} &= \frac{\ddot{x}_i (\ddot{x} \cdot \dot{x})^{1/2} - \ddot{x}_i (\ddot{x} \cdot \dot{x})}{(\ddot{x} \cdot \dot{x})^{3/2}} = 0, \\
\frac{d}{dt} \frac{\partial \hat{f}_1}{\partial \dot{x}_i} &= \frac{2 \ddot{x}_i (\ddot{x} \cdot \dot{x}) - \ddot{x}_i (\ddot{x} \cdot \dot{x})}{(\ddot{c} \cdot \ddot{c})^{3/2}} = 0.
\end{align*}
\]

Now suppose that the curve \( x = \mathbf{c} \) is a minimum of the exact functional \( F_1(\mathbf{x}) \). This means that the curve \( \mathbf{c} \) must satisfy (2.23). By substitution in (2.24), it follows directly that \( \mathbf{c} \)
also satisfies (2.24). Therefore we can conclude that a minimum \( \mathbf{x} = \mathbf{c} \) of \( F_1(\mathbf{x}) \) is also a minimum of \( \hat{F}_1(\mathbf{x}; \mathbf{c}) \).

For the case \( i = 2 \) we will follow the same approach. For \( F_2 \) and \( \hat{F}_2 \) we can derive

\[
\begin{align*}
\frac{\partial f_2}{\partial x_i} &= \dot{x}_i \left( -3(\dot{x} \cdot \ddot{x}) (\dddot{x} \cdot \ddot{x}) + 5(\dddot{x} \cdot \ddot{x})^2 \right) + \ddot{x}_i \left( -2(\dot{x} \cdot \dddot{x}) \right), \\
\frac{\partial^2 f_2}{\partial x_i^2} &= \dddot{x}_i \left( -2(\dot{x} \cdot \dddot{x}) \right) + \ddot{x}_i \left( \frac{2}{(\dot{x} \cdot \dddot{x})^3/2} \right), \\
\frac{\partial f_2}{\partial \dot{x}_i} &= \dot{x}_i \left( -2(\dot{x} \cdot \dddot{x}) \right) + \ddot{x}_i \left( -2(\dot{x} \cdot \dddot{x}) \right), \\
\frac{\partial^2 f_2}{\partial \dot{x}_i^2} &= \dddot{x}_i \left( -2(\dot{x} \cdot \dddot{x}) \right) + \ddot{x}_i \left( \frac{2}{(\dot{x} \cdot \dddot{x})^3/2} \right).
\end{align*}
\]

The condition that a minimum \( \mathbf{x} = \mathbf{c} \) of \( F_2(\mathbf{x}) \) is also a minimum of \( \hat{F}_2(\mathbf{x}; \mathbf{c}) \) is met if the following expression is zero:

\[
\begin{align*}
\frac{\partial F_2}{\partial x_i} - \frac{d}{dt} \frac{\partial F_2}{\partial \dot{x}_i} + \frac{d^2}{dt^2} \frac{\partial^2 F_2}{\partial \dot{x}_i^2} \bigg|_{\mathbf{x} = \mathbf{c}} - \frac{\partial \hat{F}_2}{\partial x_i} - \frac{d}{dt} \frac{\partial \hat{F}_2}{\partial \dot{x}_i} + \frac{d^2}{dt^2} \frac{\partial^2 \hat{F}_2}{\partial \dot{x}_i^2} \bigg|_{\mathbf{x} = \mathbf{c}} &= 0 - \frac{d}{dt} \left( \dot{\mathbf{c}} \right) \frac{\partial^2}{\partial \dot{x}_i^2} \left( \frac{\dot{\mathbf{c}} \cdot \dddot{\mathbf{c}} + 3(\dot{\mathbf{c}} \cdot \dddot{\mathbf{c}})^2}{(\dot{\mathbf{c}} \cdot \dddot{\mathbf{c}})^3/2} \right) + \frac{d^2}{dt^2} \left( \frac{\dot{\mathbf{c}} \cdot \dddot{\mathbf{c}}}{(\dot{\mathbf{c}} \cdot \dddot{\mathbf{c}})^3/2} \right) = 0.
\end{align*}
\]

However, it is not obvious that the right hand side of this equation is equal to zero for a minimum \( \mathbf{c} \) of \( F_2(\mathbf{x}) \). This indicates that the above mentioned condition probably doesn’t hold for the case \( i = 2 \). As a result, the iterative method doesn’t converge for \( i = 2 \). The experiments in section 3.6 also point into this direction.

### 2.4.8 Smoothness integrals for surfaces

The goal of variational surface modeling is to find smooth surfaces. Analogous to curves (section 2.4.5), the area of a surface

\[ F_1(\mathbf{F}) = \int_\Omega \, d\omega_\mathbf{F}, \quad (2.25) \]

(with \( d\omega_\mathbf{F} = \|\mathbf{F}_u \times \mathbf{F}_v\| \, du \, dv \) the infinitesimal surface element) and the integral of the squared principal curvatures

\[ F_2(\mathbf{F}) = \int_\Omega \kappa_1^2 + \kappa_2^2 \, d\omega_\mathbf{F} \quad (2.26) \]

are two integrals that are often used to measure the smoothness of a surface \( \mathbf{F} \), defined on the region \( \Omega \) (see also section 4.3). The integrals \( F_i(\mathbf{F}), i = 1, 2 \), are both geometric properties of the surface \( \mathbf{F} \), i.e. they are invariant under a parameter transformation \( \varphi \):

\[ F_i(\mathbf{F} \circ \varphi) = F_i(\mathbf{F}), \quad i = 1, 2. \]
2.4 DIFFERENTIAL GEOMETRY

2.4.9 Approximations of smoothness integrals

Unfortunately, the smoothness integrals (2.25) and (2.26) are too complicated for most practical applications. Just like for curves, we will define data dependent approximations to these integrals.

Let $M$ be the set of tangent plane continuous surfaces in $\mathbb{R}^3$ with piecewise continuous curvature, defined on the region $\Omega$. Let $\mathbf{F}$ and $\mathbf{G}$ be two surfaces in $M$. Then we define the data dependent approximation $\hat{F}_1(\mathbf{F}; \mathbf{G})$ of $F_1(\mathbf{F})$ as

$$\hat{F}_1(\mathbf{F}; \mathbf{G}) = \int_{\Omega} \sum_{i=1}^{3} \text{grad}_{\mathbf{G}}(F_i) \cdot \text{grad}_{\mathbf{G}}(F_i) d\omega_{\mathbf{G}}$$

and the data dependent approximation $\hat{F}_2(\mathbf{F}; \mathbf{G})$ of $F_2(\mathbf{F})$ as

$$\hat{F}_2(\mathbf{F}; \mathbf{G}) = \int_{\Omega} \sum_{i=1}^{3} \text{trace} \left( \text{Hess}_{\mathbf{G}}(F_i)^2 \right) d\omega_{\mathbf{G}}.$$

The approximations $\hat{F}_1$ and $\hat{F}_2$ have the following properties:

- They are exact for $\mathbf{F} = \mathbf{G}$, i.e. $\hat{F}_i(\mathbf{G}; \mathbf{G}) = F_i(\mathbf{G})$, $i = 1, 2$.
- They are invariant under a parameter transformation $\varphi$ of $\mathbf{F}$ and $\mathbf{G}$:

$$\hat{F}_i(\mathbf{F} \circ \varphi; \mathbf{G} \circ \varphi) = \hat{F}_i(\mathbf{F}; \mathbf{G}), \quad i = 1, 2.$$

The first property follows directly from the properties (2.15) of the gradient and the Hessian. The second property is a consequence of the fact that the gradient and the Hessian are independent of the parametrization of a surface, since they were defined as tensors.

The approximations $\hat{F}_i$ for surfaces haven’t been investigated theoretically any further. Presumably they behave the same as the corresponding approximations for curves. At least the experimental results from section (4.6) point in this direction. It should be possible to generalize the norm $\| \cdot \|_c$ for curves (2.19) to a norm $\| \cdot \|_{\mathbf{G}}$ for surfaces, such that an error estimation similar to the one in proposition 5 can be made. The iterative method for curves described in section 2.4.7 can be straightforwardly generalized to surfaces. Some experiments with this method are found in section 4.6.
2.5 Numerical methods

In this section the numerical methods are discussed that are used to solve the energy minimization problems that occur in variational modeling. First linear systems are treated (section 2.5.1), then the minimization of nonlinear functions (section 2.5.2 and 2.5.3), with quadratic functions as an important special case (section 2.5.4). Finally, in section 2.5.5 the numerical integration of one and two dimensional integrals is discussed.

2.5.1 Linear systems

In this section we consider the linear system

$$Ax = b,$$  \hspace{1cm} (2.27)

where $A$ is a large sparse regular $n \times n$ matrix and $b$ an $n$-vector.

There are two basic approaches to solve such a system: direct and iterative. A direct method usually involves a matrix factorization such as

$$PAQ = LU,$$

where $P$ and $Q$ are permutation matrices and $L$ and $U$ are triangular matrices. The solution $x^* = A^{-1}b$ is then retrieved by solving

$$Ly = Pb, \hspace{1cm} Ux = y, \hspace{1cm} x^* \leftarrow Qx.$$

An iterative method generates a sequence of approximations $\{x_k\}_{k=0}^{\infty}$ that converges to the solution $x^* = A^{-1}b$. In practice, the iteration is stopped when a sufficiently accurate approximation is obtained. Iterative methods usually require less storage than direct methods. This is because iterative methods access the coefficient matrix $A$ of the linear system only via the matrix-vector product $y = Ax$ (and perhaps $z = A^T x$).

Direct methods

An $LU$-decomposition of a nonsingular matrix $A$ is a factorization of the form $A = LU$, with $L$ a lower and $U$ an upper triangular matrix. The Cholesky decomposition of a symmetric positive definite matrix $A$ is the unique factorization of the form $A = LL^T$, with $L$ a lower triangular matrix. For both factorizations stable algorithms exist (see [GL90]). These algorithms are expensive: the number of operations (multiplication and addition) involved is $O(n^3)$. Direct methods should therefore only be used if the dimension $n$ of the problem is small (say less than 500).

If the matrix $A$ is banded, this band structure can be exploited. A matrix $A = [a_{ij}]$ has upper bandwidth $q$ if $a_{ij} = 0$ whenever $j > i + q$ and lower bandwidth $p$ if $a_{ij} = 0$ whenever $i > j + p$. If $p$ and $q$ are equal, the term bandwidth is used. Now suppose $A$ is symmetric and positive definite and has bandwidth $p$. Then the Cholesky decomposition requires only $\frac{2n^2}{3} + O(p^3 + pn)$ operations (see [GL90]). A direct approach is therefore very attractive in the special case that $A$ is symmetric positive definite, and has a small bandwidth $p$. 


2.5 NUMERICAL METHODS

Iterative methods

Many different iterative methods exist for solving large sparse systems of linear equations. Each method is appropriate for a specific class of matrices, depending on properties like (positive) definiteness and symmetry. In [BBC+94] an overview of the most important iterative methods is given. In this section a short description will be given of three conjugate gradient type methods that are used in the sections (3.6) and (4.6) to solve the energy minimization problems.

For symmetric problems, the convergence rate of iterative methods mainly depends on spectral properties of the coefficient matrix $A$, i.e. on the distribution of the eigenvalues. An often applied technique to improve the spectral properties of the coefficient matrix is the following. If the matrix $M$ approximates the coefficient matrix $A$ in some way, the transformed system

$$M^{-1}Ax = M^{-1}b$$

has the same solution as the original system (2.27), but the spectral properties of its coefficient matrix $M^{-1}A$ may be more favorable (see [BBC+94]). The matrix $M$ is called a preconditioner. There is always a trade-off between the cost of constructing and applying the preconditioner, and the gain in convergence speed.

The conjugate gradient algorithm (CG) is a well-known iterative method for solving (2.27), with $A$ a positive definite matrix. The iterates are updated according to the relation $x_{k+1} = x_k + d_k$, in such a way that the vectors $d_k$ are conjugate with respect to $A$, i.e. $d_k^TAd_j = 0$ if $i \neq j$ and $d_i^TAd_j \neq 0$ if $i = j$. The conjugate gradient algorithm has the following minimization property in terms of the norm $\|x\|_A = \sqrt{x^TAx}$:

$$\|x_k - x^*\|_A = \min_{x \in x_0 + K_k(r_0; A)} \|x - x^*\|_A, \quad x_k - x_0 \in K_k(r_0; A),$$

(2.28)

where $K_k(r_0; A) = \text{span} \{r_0, Ar_0, \ldots, A^{k-1}r_0\}$ is the $k$-th Krylov subspace of $\mathbb{R}^n$ generated by $r_0 = Ax_0 - b$, and $x^*$ is the solution of (2.27). The conjugate gradient algorithm will therefore find the solution $x^*$ in at most $n$ iterations (where we have to assume that computations are performed exactly). In general a much smaller number of iterations is needed to obtain a good approximation to $x^*$.

For a symmetric indefinite matrix $A$, the conjugate gradient algorithm cannot be used. A possible solution is to solve the equivalent system $A^T\bar{x} = A\bar{b}$, where now $A^T$ is positive definite. The main drawback of this approach is that the new equations are much more ill-conditioned. In [Sto83], two conjugate gradient like methods are described to solve (2.27) for indefinite symmetric matrices: the conjugate residual algorithm (CR), and stabilized orthogonal directions (STOD). For both methods a characterization similar to (2.28) can be given.

All three methods (CG, CR and STOD) have minimal memory requirements and have proven to be stable.
2.5.2 Unconstrained minimization

In this section we consider (unconstrained) optimization problems of the form

\[
\minimize_{x \in \mathbb{R}^n} f(x)
\]

(2.29)

where \( f : \mathbb{R}^n \to \mathbb{R} \) is continuous and possesses continuous second partial derivatives. In a solution point of (2.29) certain conditions must hold (see [Lue89], [Fle86]). These conditions are simply extensions to \( \mathbb{R}^n \) of the well-known derivative conditions for a function of a single variable that hold at a maximum or a minimum point.

The first-order necessary conditions state that

**Proposition 6** Let \( f \in C^1 \) be a function on \( \mathbb{R}^n \). If \( x^* \) is a relative minimum point of \( f \), then for any \( d \in \mathbb{R}^n \) we have \( \nabla f(x^*)d \geq 0 \).

The second-order necessary conditions state that

**Proposition 7** Let \( f \in C^2 \) be a function on \( \mathbb{R}^n \). If \( x^* \) is a relative minimum point of \( f \), then for any \( d \in \mathbb{R}^n \) we have

\[
\text{(i)} \quad \nabla f(x^*)d \geq 0 \\
\text{(ii)} \quad \text{if } \nabla f(x^*)d = 0 \text{ then } d^T \nabla^2 f(x^*)d \geq 0.
\]

Iterative methods

If the function \( f \) is nonlinear, the problem (2.29) is usually solved using an *iterative method*. An iterative method generates a sequence of approximations \( \{x_k\}_{k=0}^{\infty} \) that converges to a local minimum \( x^* \) of the function \( f \). Iterative methods are usually based on a descent principle. This means that the following must hold: if the point \( x_k \) is not a solution to (2.29) then \( f(x_{k+1}) < f(x_k) \) and if the point \( x_k \) is a solution to (2.29) then \( f(x_{k+1}) = f(x_k) \).

Iterative methods can be described with the relation

\[
x_{k+1} = x_k + t_k d_k.
\]

In the point \( x_k \) a descending *search direction* \( d_k \) is determined from (estimates of) the first and second derivatives of \( f \) at \( x_k \). The nonnegative value \( t_k \) is then determined by performing a *line search* in the direction \( d_k \), i.e. by minimizing the univariate function \( g(t) = f(x_k + td_k) \).

One of the oldest and most widely known methods for minimizing a function of several variables is the method of *steepest descent*. For this method the search direction is the negative gradient, i.e. \( d_k = -\nabla f(x_k)^T \). As a consequence of this two successive search directions are always orthogonal, which is considered as an important drawback of the method. In general, the method of steepest descent converges linearly. Another well-known method is *Newton’s method*. For this method the search direction is chosen as follows:
2.5 NUMERICAL METHODS

\[ d_k = -F(x_k)^{-1}\nabla f(x_k)T \], with \( F(x_k) \) the Hessian of \( f \) at \( x_k \). In general, the order of convergence of Newton’s method is two.

For a lot of problems, the convergence associated with steepest descent is too slow, and the requirements of Newton’s method (evaluation, storage, and inversion of the Hessian) are too demanding. In the implementation of variational modeling (see section 3.6 and 4.6) the conjugate gradient method is used, which can be regarded as being somewhat intermediate between these two methods. In the conjugate gradient method, the search direction \( d_k \) is computed from the gradients \( \nabla f(x_0), \ldots, \nabla f(x_k) \). This method is only slightly more complicated than steepest descent, and it does not suffer from the problem that successive search directions are orthogonal. A clear description of the conjugate gradient method can be found in [She94].

It can be shown that the conjugate gradient for the linear system \( Ax = b \), given in section 2.5.1, is in fact the (nonlinear) conjugate gradient method applied to the function \( f(x) = x^T Ax - 2b^T x \).

2.5.3 Nonlinear programming

In this section we consider the general nonlinear programming problem with equality constraints of the form

\[
\begin{align*}
\text{minimize} & \quad f(x), \\
\text{subject to} & \quad h(x) = 0
\end{align*}
\]  

where the functions \( f : \mathbb{R}^n \to \mathbb{R} \) and \( h : \mathbb{R}^n \to \mathbb{R}^m \), \( m \leq n \), are continuous, and possess continuous second partial derivatives. A point \( x \) that satisfies the constraints \( h(x) = 0 \) is called a feasible point. A feasible point \( x \) is called regular if the gradient vectors \( \nabla h_1(x), \nabla h_2(x), \ldots, \nabla h_m(x) \) are linearly independent.

For a regular point \( x \) that satisfies the constraints \( h(x) = 0 \), necessary and sufficient conditions can be derived for \( x \) to be a local minimum of (2.31), see [Luc89]. The first-order necessary conditions are given by

**Theorem 8** Suppose that \( x^* \) is a local extremum point of \( f \) subject to \( h(x) = 0 \) and that \( x^* \) is a regular point of these constraints. Then there is a \( \lambda \in \mathbb{R}^m \) such that

\[
\nabla f(x^*) + \lambda^T \nabla h(x^*) = 0,
\]  

where \( \nabla h(x^*) \) is the \( m \times n \) matrix defined by

\[
\nabla h(x) = \left[ \frac{\partial h_i(x)}{\partial x_j} \right].
\]

The vector \( \lambda \) is called the vector of Lagrange multipliers. The first-order necessary equations (2.32), together with the constraints \( h(x^*) = 0 \), give a total of \( n + m \) equations in the \( n + m \) variables comprising \( x^*, \lambda \). These equations are called the Lagrange necessary conditions for the problem (2.31).
CHAPTER 2 MATHEMATICAL BACKGROUND

Linear constraints

An important subclass of the nonlinear programming problems are the problems with linear constraints:

\[
\begin{align*}
\text{minimize} & \quad f(x), \\
\text{subject to} & \quad Ax = b,
\end{align*}
\]

with \( A \) an \( m \times n \) matrix (\( m < n \)) with full rank \( m \).

A straightforward way of solving (2.33) is to use the constraints to eliminate variables. In [Col84] and [Fle86] a so called \textit{generalized elimination method} is described in which essentially a linear transformation of variables is made. Let the matrix \( A \) be partitioned into \( [A_1 : A_2] \), where \( A_1 \) is invertible. For this, the columns of \( A_2 \) (and the corresponding components of \( b \) and \( x \)) possibly have to be reordered. We define the \( n \times (n - m) \) matrix \( C \) and the \( (n - m) \)-vector \( d \) as

\[
C = \begin{bmatrix}
-A_1^{-1}A_2 \\
I_{n-m}
\end{bmatrix}, \quad d = \begin{bmatrix}
A_1^{-1}b \\
0
\end{bmatrix}.
\]

The \textit{null column space} of the matrix \( A \) is the linear space

\[
\{ x \mid Ax = 0 \},
\]

which has dimension \( n - m \). The columns of \( C \) form a basis of this null space, which can be verified by noting that \( AC = 0 \). Any vector \( x \) that satisfies the constraints \( Ax = b \) can be uniquely written as

\[
x = Cy + d.
\]

Thus (2.34) provides a way of eliminating the constraints \( Ax = b \) in terms of the vector \( y \). The vector \( y \) is called the vector of \textit{reduced variables}. After substitution of (2.34) into (2.33), the following unconstrained minimization problem remains:

\[
\text{minimize} \quad g(y) = f(Cy + d),
\]

(2.35)

The gradient \( \nabla g \) and the Hessian \( G \) of the function \( g \) can be expressed into the gradient \( \nabla f \) and Hessian \( F \) of \( f \) as follows:

\[
\nabla g(y) = \nabla f(Cy + d)C, \\
G(y) = C^TF(Cy + d)C.
\]

2.5.4 Quadratic programming

The general \textit{quadratic programming problem} with equality constraints only can be expressed as
2.5 NUMERICAL METHODS

\[
\text{minimize } f(x) = x^T A x + b^T x + c \\
\text{such that } \quad D x = e,
\]

for some \( A \in \mathbb{R}^{m \times n} \), \( b \in \mathbb{R}^n \), \( c \in \mathbb{R} \), \( D \in \mathbb{R}^{m \times n} \) and \( e \in \mathbb{R}^m \), \( m \leq n \). We will assume that the matrix \( D \) has full rank, that the matrix \( A \) is positive semidefinite and that \( A \) is positive definite on the null column space of \( D \), such that the problem has a unique solution. Furthermore we assume that the matrices \( A \) and \( D \) are sparse.

This problem can be solved in a variety of ways (see [Col84], [Fel86] and [Lue89]). Two of them will be mentioned, a null space method and the Lagrangian method.

**Nullspace method** The quadratic programming problem (2.36) is a special case of the nonlinear programming problem with linear constraints, that was discussed in the previous section. Again, a matrix \( F \) and a vector \( g \) can be computed such that any vector \( x \) that satisfies the constraints \( D x = e \) can be uniquely written as \( x = F y + g \). After substitution of this equation into (2.36), the following unconstrained minimization problem is found, that is the equivalent of (2.35)

\[
\text{minimize } g(y) = y^T P y + q^T y + r, \\
\quad y \in \mathbb{R}^{n-m},
\]

with \( P = F^T A F \), \( q = (A^T g + b^T F) \) and \( r = g^T A g + b^T g \). Since the matrix \( A \) is positive semidefinite and \( F \) has full rank, the matrix \( P \) is positive semidefinite too. Now if \( y^* \) is the minimum of \( g \), then \( x^* = F y^* + g \) is the solution of (2.36).

From the first-order necessary conditions of this unconstrained problem (2.37) it can be derived that the solution \( y^* \) must satisfy the following relation

\[
(P + P^T)y = -q. \tag{2.38}
\]

This linear system (2.38) can be solved using the techniques discussed in section 2.5.1. We can make the following remarks regarding the procedure to compute \( F \) and \( g \).

1. For iterative methods it is important that the condition number of the matrix \( P = F^T A F \) is not too high. By choosing a matrix \( F \) with orthonormal columns, one can assure that the condition numbers of \( A \) and \( P \) are the same.

2. Especially if the number of constraints \( (m) \) is large, the procedure to compute the matrix \( F \) and the vector \( g \) should exploit the sparseness of \( D \). Note that for an iterative method the matrix \( P \) doesn’t have to be computed explicitly, since most iterative methods require only the computation of the matrix vector product \( (P + P^T)x \).

3. If the matrix \( A \) is banded, one can try to choose the matrix \( F \) such that \( P = F^T A F \) is also banded. If a procedure like in section 2.5.3 is used, where the variables \( x \) are reordered, at least this reordering should be undone. However, to guarantee that the bandwidth of the matrix \( P \) is not much bigger than the bandwidth of the matrix \( A \), the matrix \( D \) should have a particularly simple structure.
Lagrangian method  An alternative way of deriving the solution \( x^* \) to the problem (2.36) is by the method of Lagrange multipliers. The Lagrange necessary conditions for this problem are linear in the unknowns \( x \) and the Lagrange multipliers \( \lambda \):

\[
\begin{pmatrix}
A + A^T & D^T \\
D & 0
\end{pmatrix}
\begin{pmatrix}
x \\
\lambda
\end{pmatrix} =
\begin{pmatrix}
-b \\
e
\end{pmatrix},
\]

(2.39)

The coefficient matrix is referred to as the Lagrangian matrix and is symmetric and indefinite. If the matrix \( A \) is positive definite, (2.39) is called a saddle point problem (see [FGS3]). The linear system (2.39) can be solved using a suitable factorization method or, if the dimension \( n + m \) is too high, by one of the iterative methods described in section 2.5.1.

2.5.5 Numerical integration

For the energy minimization of curves and surfaces certain one- and two-dimensional integrals have to be computed. It is sometimes impractical (or even impossible) to evaluate these integrals analytically. In these cases numerical integration formulas have to be used. In the one-dimensional case the well known Gauss-Legendre formulas have been used, and in the two-dimensional case the formulas listed in [Str71].

The one-dimensional case

The Gauss-Legendre integration formula is given by

\[
\int_{-1}^{1} f(x) \, dx \simeq \sum_{i=1}^{n} A_i f(x_i)
\]

(2.40)

where \( x_1, \ldots, x_n \) are the \( n \) zeroes of the Legendre polynomial \( P_n(x) \) and \( A_1, \ldots, A_n \) are suitably chosen constants (see [Str71]). The formula (2.40) is a degree \( 2n - 1 \) formula. This means that the approximation is exact for polynomials of degree not exceeding \( 2n - 1 \). A numerical integration formula for an arbitrary interval \([a, b]\) can be obtained by applying a coordinate transformation to the integral in (2.40).

The two-dimensional case

A general form of a numerical integration formula in two dimensions is as follows:

\[
\iint_{R} f(x, y) \, dx \, dy \simeq \sum_{i=1}^{N} w_i f(x_i, y_i),
\]

(2.41)

with \( R \) a given region in \( \mathbb{R}^2 \). The \( x_i = (x_i, y_i) \) are points in \( \mathbb{R}^2 \) and are called the points (or nodes) of the formula; the \( w_i \) are constants which do not depend on the function \( f \) and are called the coefficients of the formula. The formula (2.41) is said to have degree \( d \) if it is
2.5 NUMERICAL METHODS

exact for all polynomials in $x, y$ of degree $\leq d$ and there is at least one polynomial of degree $d + 1$ for which it is not exact.

In [Str71] a large number of tables containing weights and coefficients for numerical integration formulas are listed. We will use the formulas for standard regions like the unit 2-cube $C_2$ and the 2-simplex $T_2$. As in the one-dimensional case, the formulas for an arbitrary rectangle and an arbitrary triangle can be obtained by applying an (affine) coordinate transformation to (2.41).
2.6 Miscellaneous

This section contains a few miscellaneous mathematical topics that will be used in the following chapters.

2.6.1 Distance between point and line

Let \( l \) be the 3D line given by \( \{ \mathbf{x} \in \mathbb{R}^3 \mid \mathbf{x} = \mathbf{r} + t \mathbf{s}, t \in \mathbb{R} \} \). The distance \( d(\mathbf{x}, l) \) between the point \( \mathbf{x} \) and the line \( l \) is given by

\[
d(\mathbf{x}, l) = \|A\mathbf{x} + \mathbf{b}\|, \tag{2.42}
\]

where \( A = I_3 - ss^T/(s \cdot s) \) and \( \mathbf{b} = (\mathbf{r} \cdot s)s/(s \cdot s) - \mathbf{r} \).

2.6.2 Distance between point and plane

Let \( V \) be the 3D plane given by \( \{ \mathbf{x} \in \mathbb{R}^3 \mid \mathbf{n} \cdot \mathbf{x} = d \} \). The distance \( d(\mathbf{x}, V) \) between the point \( \mathbf{x} \) and the plane \( V \) is given by

\[
d(\mathbf{x}, V) = |\mathbf{n} \cdot \mathbf{x} - d|/\|\mathbf{n}\|. \tag{2.43}
\]

2.6.3 Parallel projection

Let \( V \) be the 3D plane given by \( \{ \mathbf{x} \in \mathbb{R}^3 \mid \mathbf{n} \cdot \mathbf{x} = d \} \) and let \( \mathbf{r} \) be a given direction. We define the parallel projection \( PP(\mathbf{x}) \) of a point \( \mathbf{x} \) into the plane \( V \) along the direction \( \mathbf{r} \) as the intersection of the line through \( \mathbf{x} \) parallel to \( \mathbf{r} \) and the plane \( V \). It is given by

\[
PP(\mathbf{x}) = A\mathbf{x} + \mathbf{b}, \tag{2.44}
\]

where \( A = (I_3 - \mathbf{r}\mathbf{r}^T)/(\mathbf{n} \cdot \mathbf{r}) \) and \( \mathbf{b} = d\mathbf{r}/(\mathbf{n} \cdot \mathbf{r}) \).

2.6.4 Quadratic functions

Minimizing the energy of a variational curve often comes down to minimizing a quadratic function. In this section some properties of quadratic functions are given.

Let \( f : \mathbb{R}^n \to \mathbb{R} \) be a quadratic function defined by

\[
f(\mathbf{x}) = \mathbf{x}^TA\mathbf{x} + \mathbf{b}^T\mathbf{x} + c,
\]

with \( A \in \mathbb{R}^{n \times n} \), \( \mathbf{b} \in \mathbb{R}^n \) and \( c \in \mathbb{R} \). The gradient of \( f \) is given by

\[
\nabla f(\mathbf{x}) = \mathbf{x}(A + A^T) + \mathbf{b}^T
\]

and the Hessian of \( f \) is given by

\[
F(\mathbf{x}) = A + A^T.
\]
2.6 MISCELLANEOUS

If the Hessian is a positive definite definite matrix, the function $f$ has a unique minimum. This minimum can be found by setting the gradient to 0. This is equivalent to solving the linear system
\[(A + A^T)x = -b.\]

Now let $g$ be the reciprocal of $f$:
\[g(x) = \frac{1}{f(x)}.\]

Then the gradient and the Hessian of $g$ can be expressed in the gradient and the Hessian of $f$:
\[\nabla g(x) = \frac{-\nabla f(x)}{f^2(x)}\]
and
\[G(x) = \frac{-f(x)F(x) + 2f(x)^T \nabla f(x)}{f^3(x)}\]

Now let $g$ be given by
\[g(x) = f(Dx + e)\]
then
\[\nabla g(x) = \nabla f(Dx + e)D\]
and
\[G(x) = D^TF(Dx + e)D\]
Chapter 3
Curves

3.1 Introduction

The modeling of smooth curves is very important in areas like industrial design, desk top publishing and styling. Applications are found in the design of automobiles, ships and aircraft. A problem that occurs very often is the curve-fitting problem:

Given an ordered set of points $P_i$ in the Euclidean plane

$$P_i = (x_i, y_i), \quad i = 1, \ldots, n,$$

construct a smooth curve passing through or near these points that captures the ‘design intent’.

To solve the problem, it must be specified what is meant by a smooth curve and what is meant by the design intent. It should not only be smooth in a mathematical sense, but also in an aesthetic sense.

3.1.1 Fairness

In the past, the curve-fitting problem problem was solved manually by a draftsman or boardman. In [BAFS94] a number of criteria are listed that a boardman normally uses to determine if a curve is smooth (or fair):

- S-shaped segments (inflection points) should be avoided
- flat spots should be avoided
- buckles or bumps should be avoided
- the curve should be ‘visually pleasing’

The first two criteria are illustrated in figure (3.1).

For a boardman these criteria will do. Based on his experience, he can decide if corrections to a curve are needed. For a mathematician the given criteria are too vague. They can
be made more precise using the following observation: most features of a curve can be conveniently expressed in the curvature $\kappa$. Table 3.1 summarizes a number of them (see also [BAFS94]).

<table>
<thead>
<tr>
<th>feature</th>
<th>mathematical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>inflection point</td>
<td>point where $\kappa$ changes sign</td>
</tr>
<tr>
<td>convex curve</td>
<td>$\kappa \geq 0$</td>
</tr>
<tr>
<td>concave curve</td>
<td>$\kappa \leq 0$</td>
</tr>
<tr>
<td>flat spot</td>
<td>minimum of $</td>
</tr>
<tr>
<td>buckle, bump, corner or break</td>
<td>maximum of $</td>
</tr>
<tr>
<td>curve is accelerating/decelerating</td>
<td>$</td>
</tr>
</tbody>
</table>

Table 3.1: Visual features of a curve expressed in the curvature $\kappa$

The mathematical descriptions of curve features help us to determine whether or not a curve is fair. But what we really want is a procedure that enables us to automatically compute a fair curve. For that, a fairness functional is needed. This is a functional that expresses the fairness of a curve in a number. Often this number is referred to as the energy of a curve. Once a fairness functional is given, the curve of minimal energy that satisfies the constraints can be computed using an automatized procedure.

### 3.1.2 Splines

The idea of using fairness functionals is not new. For well over forty years they have been applied in the subject of spline functions. The name spline comes from a very old technique in drafting in which a long thin strip of wood, called a draftsman’s spline, is used to pass a smooth curve through a set of points. The spline is secured at the points of interpolation by means of lead weights. A spline can be characterized mathematically by the fact that it
3.1 INTRODUCTION

assumes a shape which minimizes its strain energy. Daniel Bernoulli (1742) first suggested that the strain energy is proportional to the integral of the square of the curvature \( \kappa \) (with respect to arc length \( s \)) taken along the curve \( \mathbf{x} \):

\[
E_{\text{strain}}(\mathbf{x}) = \int \kappa^2 \, ds.
\]

(3.1)

This observation has led to the theory of spline functions. A curve that minimizes the strain energy (3.1) is referred to as a nonlinear spline (see below) or a minimal-energy spline. Usually, the curve with minimal energy is sought in the following set of admissible curves:

i) the curve interpolates the ordered set of points \( \{ \mathbf{P}_i \}_{i=1...n} \)

ii) the tangent direction is continuous everywhere

iii) the curvature is piecewise continuous, with discontinuities permitted on a finite set of points

Sometimes the set of admissible curves is even further restricted:

iv) the tangent directions at the end-points are prescribed

v) the segment of the curve connecting the points \( \mathbf{P}_i \) and \( \mathbf{P}_{i+1} \) has prescribed length \( l_i \), \( i = 1 \ldots n - 1 \)

The problem of finding an admissible curve that minimizes the strain energy is a problem from variational calculus (see [For60]). For this reason, modeling with fairness functionals is often referred to as variational modeling. With techniques from variational calculus, a differential equation can be derived (see [LF73] or [Mal77]) that must be satisfied by curves for which the strain energy attains a minimum. This differential equation is nonlinear, which is the reason why the splines are also called nonlinear.

Many research has been done for nonlinear splines, theoretical as well as practical. Theoretical research is mainly focused on existence and uniqueness of solutions. Some results are known, but these problems are not completely solved yet. There are sets of points and tangents for which no solution exists, or more than one solution, with different numbers of loops between adjacent points (see [LF73]). In [JH90] conditions are derived for which a solution exists in the case of the one-segment problem with prescribed tangents at the end-points and prescribed length. In the absence of condition v) an absolute minimum to (3.1) does not exist, except in the trivial case of a straight line (see [BdB65]); this is because one can construct large loops joining given end-points with given end-slopes, of length \( 2\pi r \) and curvature \( \kappa = O(1/r) \), for arbitrarily large \( r \), hence with strain energy less than any preassigned positive number. The strain energy (3.1) is defined in terms of the geometric invariants ar length and curvature. For this reason, the strain energy is invariant under parameter transformations and under rigid rotations of the curve.

Of a very practical nature is the following observation, made by [Hor83]: the fact that a curve has near minimal energy does not imply that this curve will necessarily lie very close to the curve of minimal energy!
CHAPTER 3 CURVES

Practical research on nonlinear splines is mainly focused on finding methods for approximating minimal-energy splines. In [Mal77] an overview of these methods is given. This research is still in progress, see [BK94].

The fact that the nonlinear differential equation is difficult to solve has inspired people to use a linearized version. If \( x_1 < x_2 < \ldots < x_n \), the curve can be represented as the graph of a function \( f \), so that \( y = f(x) \). Under the assumption that the first derivative of \( f \) is sufficiently small (which is not always very realistic), the strain energy may be approximated with

\[
\hat{E}_{\text{strain}}(x) = \int \left( \ddot{f}(x) \right)^2 dx,
\]

see [HS91].

Curves that minimize (3.2), under the conditions i), ii) and iii) are referred to as linear splines, or natural cubic spline functions. They are the solution to a fourth order linear differential equation. This differential equation always has a unique solution, which is a piecewise cubic polynomial. A disadvantage of linear splines is that they are parametrization dependent. Two curves with the same shape but with a different parametrization may therefore have different values of the energy (3.2).

3.1.3 Other fairness functionals

In the literature, several other fairness functionals can be found. Here we will only consider the ones that are independent of the parametrization of the curve. They are all defined in terms of the geometric invariants arc length \( s \), curvature \( \kappa \) and torsion \( \tau \) (see section 2.4). The independence of the parametrization is a desirable property of fairness functionals, since changing the parametrization does not change the shape of the curve. However, in practice it is hard to work with quantities like \( s, \kappa, \) and \( \tau \). The corresponding fairness functionals are very complicated, and the computation of a minimal energy curve may be very elaborate. For practical applications, it is therefore common practice to use parametrization dependent approximations of these fairness functionals.

In [HS91], an overview of fairness functionals is given. An important one is the extension of the strain energy with a weighted arc length part:

\[
E_{\text{tension}}(x) = \int \kappa^2 ds + \sigma^2 \int 1ds
\]

This addition of an arc length part has two advantages. In the first place, the problem that the energy can be made arbitrarily small by introducing large loops is overcome. In the second place, unwanted inflection points can be removed by increasing the tension \( \sigma^2 \). The extremals of (3.3) are called nonlinear splines in tension.

In [MS92], the variation of the curvature is used as a fairness functional:

\[
E_{\text{moredon}}(x) = \int \left( \frac{d\kappa}{ds} \right)^2 ds.
\]
3.1 INTRODUCTION

Furthermore, there exist physically meaningful fairness functionals in terms of even higher order derivatives. The functional

\[ E_{jerk}(\mathbf{x}) = \int \left( \frac{d^3 \kappa}{ds^3} \right)^2 ds \]

is defined in terms of the physical quantity called jerk and the functional

\[ E_{load}(\mathbf{x}) = \int \left( \frac{d^4 \kappa}{ds^4} \right)^2 ds \]

is defined in the physical quantity called load (see [HS91]).

In [RRP91] the following two fairness functionals for three dimensional curves are defined:

\[ E_{roulier,1}(\mathbf{x}) = \int \sqrt{\frac{\tau^2}{\kappa^2} + \left( \frac{dp}{ds} \right)^2} ds \quad (3.4) \]

and

\[ E_{roulier,2}(\mathbf{x}) = \int \sqrt{(\tau \kappa)^2 + \left( \frac{d\kappa}{ds} \right)^2} ds, \quad (3.5) \]

where \( \rho = 1/\kappa \) is the radius of curvature.

The first fairness functional (3.4) penalizes large values of \( \tau \) and large oscillations of \( \rho \). Thus, the curve will tend to avoid pulling too far out of a plane, and its radius of curvature will not oscillate too much. A problem is that (3.4) may have singularities at inflection points (\( \kappa = 0 \)). The second fairness functional (3.5) can be interpreted in a similar way.

3.1.4 Modeling with minimal energy curves

Modeling with minimal energy curves is a powerful way of modeling. It gives the designer the possibility to smooth a curve automatically, and it frees the designer of curve representation details like control points. There are two basic ways of interaction in the design process of a minimal energy curve. In the first place, one of the earlier described fairness functionals can be chosen. With the choice of the fairness functional the designer has a certain kind of global control over the shape of the curve. In the second place, constraints can be imposed upon the curve. The most common constraints are point and tangent interpolation. They can be used to sketch a rough outline of a curve.

Although this way of modeling is powerful, there are a couple of serious problems. The first problem is that the parameterization independent fairness functionals are too complicated to use for interactive modeling. In practice often parameterization dependent approximations are used. For instance, we already encountered linear splines, which are obtained by approximating the strain energy (3.1) with (3.2). The approximated fairness functionals often lead to simple quadratic functions for the energy of a curve. Quadratic functions can be efficiently minimized (see section 2.5), so they are very suitable for interactive modeling. A problem
with the current quadratic approximations is that the results can be rather poor (see section 3.3). The second problem is that the two ways of interaction with a curve are sometimes not flexible enough. If the designer wants to deform a minimal energy curve locally, interpolation constraints are not always the obvious means to do this. The first problem can be solved by using ‘data dependent’ fairness functionals, and the second problem by using external energy operators that are introduced in this dissertation (see below).

3.1.5 Data dependent fairness functionals

For interactive modeling, the parametrization independent fairness functionals mentioned above are too difficult. On the other hand, their (quadratic) approximations are simple, but they can be very poor (see also section 3.3).

Recently, Greiner has introduced the concept of data dependent fairness functionals (see [Adl93] or [Gre94]): if a smooth curve is available that is already ‘close’ to the solution, this curve can be used to obtain a much better quadratic approximation of a fairness functional than the usual ones. This technique is very promising: at a reasonable cost (data dependence and somewhat increased complexity of fairness functionals), minimal energy curves can be found that are much closer (in the sense of lower energy) to the real solutions than the usual approximations. The mathematical properties of data dependent fairness functionals are discussed in section 2.4.6, and their usage in variational modeling is discussed in section 3.3.

3.1.6 External energy operators

When a minimal energy curve is not satisfactory, a designer may decide to deform the curve locally. This could be done by adding some new constraints, or by editing the old ones. But some effects are difficult to achieve with constraints. If a part of a curve is too flat or too round, interpolation constraints are not the obvious means to change this. To solve this problem, the designer could be allowed to manipulate the curve directly (for instance, by moving its control points, see section 3.1.7). However, this can be a very tedious job. Moreover, it can only be done as a post processing step, since after minimizing the energy the effect would be undone.

A better solution is to provide the designer with special operators to deform a curve. These operators can be seen as forces that are applied to certain parts of the curve. In the literature two examples of such ‘forces’ are encountered. In [CG91] spring forces are used to attract certain points on the curve towards other points in the plane. And in [KWT88] certain image forces are used to attract a (discrete) curve towards features in a digitized image. The latter idea has led to a large amount of publications of so called active contours. In section (3.4) a whole set of ‘force-based’ operators is introduced. Like the spring and the image forces these operators are all defined in terms of energies. They can be added to the fairness functional of the curve, which means that they fit into the minimal energy approach.

The fairness functional depends only on internal properties of the curve, and will be referred to as the internal energy $E_{int}$ of the curve. The energies of the design operators are combined into the external energy $E_{ext}$. Together these energies form the total energy $E_{tot}$
3.1 INTRODUCTION

of the curve, which has to be minimized:

\[ E_{\text{tot}}(x) = E_{\text{int}}(x) + E_{\text{ext}}(x). \]  

(3.6)

3.1.7 Curve representation

For variational modeling the choice of the curve representation is very important. This representation has a large influence on the complexity of the energy minimization. Two different approaches can be followed.

The first approach is to model with curve schemes that guarantee fair curves. For example, in [Sch66] Cornu spirals (or clothoïdes) are used. A Cornu spiral is a curve with linear curvature \( \kappa \) (with respect to the arc length parameter \( s \)). An example where this principle is applied is the Dutch railways. The tracks are constructed from three different elements:

- straight line segments \( (\kappa(s) = 0) \)
- circular arcs \( (\kappa(s) = r) \)
- Cornu spirals \( (\kappa(s) = as + b) \)

At every transition from a straight line into a circular arc, a part of a Cornu spiral is put in between. In this way, the curvature of the railway becomes a piecewise linear function, which is considered as fair. Since the centrifugal force is proportional to \( \kappa \), the tracks should be chosen such that \( \kappa \) is a continuous function.

The second approach is to use existing curve schemes that are regularly used in CAGD (computer aided geometric design), like Bézier curves or B-spline curves. In this dissertation only the second approach will be followed. The shape of these curves usually depends on a number of parameters. These parameters will be called the shape parameters of the curve. To illustrate the difference of shape parameters with a parametrization of a curve as defined in section 2.3, we consider the straight line \( x(t) = tA + B \), with \( A \) and \( B \) points in \( \mathbb{R}^3 \). The parametrization of the curve is determined by the parameter \( t \), and the shape of the curve is determined by the shape parameters \( A \) and \( B \).

The majority of the curve representations in CAGD can be written in the following canonical representation:

\[ x(t) = \sum_{i=0}^{n-1} P_i N_i(t), \quad t \in [a, b], \]  

(3.7)

where \( \{P_i\}_{i=0}^{n-1} \) is a set of points in \( \mathbb{R}^3 \) (often called the control points of the curve) and \( \{N_i\}_{i=0}^{n-1} \) is a set of basis functions (most often piecewise polynomials or rationals). The shape parameters of this curve are the control points \( P_i \). The functions \( N_i \) may depend on certain parameters, but we assume that these are given fixed values. Certainly there are other possible representations (for instance implicit curves), but these will not be considered.

Suppose that the \( i \)-th control point is represented as \( P_i = (x_i, y_i, z_i)^T \). Then we define the concatenation vector \( P \) of the control points \( \{P_i\} \) as

\[ P = (x_0, y_0, z_0, \ldots, x_{n-1}, y_{n-1}, z_{n-1})^T. \]  

(3.8)
CHAPTER 3 CURVES

The shape of the curve (3.7) is completely specified by this vector.

Apart from the reason that control point based curves are often used for modeling, there is another reason to choose them for variational modeling. The most important and most difficult aspect of variational modeling is the energy minimization problem. The level of difficulty of this problem depends strongly on the curve representation. Curves of the form (3.7) score rather well regarding this difficulty level. In the sections 3.2, 3.3 and 3.4, we will see that the most often used constraints and internal and external energy functionals are particularly simple for control point based curves.

In [BGMS94], [Far93] and [Pfe95] a number of criteria is listed that have proven to be valuable for shape modeling with control point based curves of the form (3.7). These criteria will be repeated here. Some of the criteria give a geometric meaning to the control points $P_i$. This is desirable when the control points are used directly as design handles. However, in variational modeling the control points are computed automatically, so the geometric meaning of the control points is less important.

- **Local support** Local support means that every basis function $N_i$ is non-zero only over a small parameter interval, and therefore influences only a small part of the corresponding curve. When a curve is represented in such a basis, only a small number of control points needs to be altered in order to deform a curve locally. This property is not essential for variational modeling, since the control points are computed automatically. However, if the minimal energy curve is computed iteratively, the local support will probably have a positive effect on the computation of the new solution in case of small deformations. On the other hand, for large deformations the absence of basis functions with global support may cause slower convergence (see [GC95]).

- **Affine Invariance** Affine invariance means that the shape of a curve is invariant under an affine transformation $T$ of the control points:

$$T(x(t)) = T \left( \sum_{i=0}^{n} P_i N_i(t) \right) = \sum_{i=0}^{n} T(P_i) N_i(t).$$

That is, transforming the entire curve is equivalent to transforming the control points $P_i$. Clearly, if $\sum_{i=0}^{n} N_i(t) = 1$ (*summation to one*), then this property is true. If in addition the basis functions are positive, the curve lies in the convex hull of the control points (the *convex hull property*). Affine invariance is a desirable property for curve representations in general, since it makes it very easy to translate or rotate a curve. However, for variational modeling it is not an essential property, since then the control points are automatically computed.

- **Variation Diminishing property** The variation diminishing property means that the curve has no more intersections with any plane than the *control polygon* has (i.e. the polygon formed by the control points $P_i$). If this property holds in addition with the convex hull property, the curve will ‘mimic’ the control polygon. The variation diminishing property gives a geometric meaning to the control points, and is therefore not essential for variational modeling.
3.1 INTRODUCTION

- **Efficient Algorithms** Evaluating a curve at a parameter \( t \) requires the evaluation of the basis functions \( N_i \). For curve schemes like Bézier and B-spline curves, basis functions do not need to be explicitly evaluated in order to evaluate the curve itself. Rather, efficient algorithms exist for evaluating segment polynomials directly from the control point values. This is a useful property for curve representations in general.

- **Refinement** During the design process it may become clear that finer control over the shape of a curve is desired in some region. This can often be obtained by dividing polynomial segments into two or more pieces, and reformulating the curve in a new, finer basis. In the case of piecewise polynomials, the coarser basis is a subspace of the finer space. The new coefficients in the finer space can always be found by solving the appropriate system of equations. However, if this operation occurs frequently, then it is advantageous for the spline scheme to provide support in the form of efficient algorithms for calculating the new coefficient values.

For variational modeling refinement is sometimes necessary to satisfy (additional) constraints or to allow for a more accurate approximation of the minimal energy curve. It is not essential that the refinement leaves the curve unchanged, since after an energy minimization new values for the control points will be computed anyhow.

- **Automatic Continuity** If the basis functions \( N_i \) all possess \( C^k \) continuity over the whole interval \([a, b]\), then so will the curves formed from that basis, and thus we are guaranteed parametric smoothness. Such a curve representation will be called **automatic \( C^k \) continuous**. A curve representation with this property frees a shape designer from the burden of enforcing continuity by constraints on the control points. In geometric modeling often composite curves are used. A composite curve is a curve that is obtained by joining a number of separate curve segments. In variational modeling it is important that adjacent segments of composite curves are joined in a continuous way. The most often used definitions of continuity are parametric (\( C^k \)) and geometric (\( G^k \)) continuity (see also [Vel92]). For variational modeling, geometric continuity is the preferred form, since it depends on the shape and not on the parametrization of the curve. If a composite curve has no automatic continuity of the desired degree, continuity constraints have to be imposed on the curve. If these constraints take a too difficult form, the minimization problem may become computationally too expensive to solve. We will now give characterizations for geometric and parametric continuity of a curve \( \mathbf{x} \). Let \( + \) and \( - \) as subscripts denote right and left limits, respectively. The right end-point of a segment is therefore denoted as \( \mathbf{x}_+ \) and the left end-point of the adjacent segment as \( \mathbf{x}_- \). (We will only consider continuous curves, so that \( \mathbf{x}_- = \mathbf{x}_+ \) always.) A curve is **tangent continuous** if the tangent vectors \( \dot{\mathbf{x}}_- \) and \( \dot{\mathbf{x}}_+ \) are parallel. A curve is **curvature and osculating plane continuous** if in addition the planes spanned by the vectors \( \{\dot{\mathbf{x}}_-, \ddot{\mathbf{x}}_-\} \) and \( \{\dot{\mathbf{x}}_+, \ddot{\mathbf{x}}_+\} \) coincide and a curve is **curvature continuous** if the curvatures \( \kappa_- \) and \( \kappa_+ \) are the same. Finally, a curve is said to be \( G^k \), or \( k \)th order **geometrically continuous**, if a regular re-parametrization exists after which it is \( C^k \). Table 3.2 gives an overview of several types of continuity.
<table>
<thead>
<tr>
<th>type</th>
<th>arbitrary parametrization</th>
<th>arc length parametrization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C^1$-continuity</td>
<td>$\hat{x}<em>-(t) = \hat{x}</em>+(t)$</td>
<td>-</td>
</tr>
<tr>
<td>$C^2$-continuity</td>
<td>$\ddot{x}<em>-(t) = \ddot{x}</em>+(t)$</td>
<td>-</td>
</tr>
<tr>
<td>$C^3$-continuity</td>
<td>$\dot{x}<em>-(t) \times \dot{x}</em>+(t) = 0$</td>
<td>$x'<em>-(s) = x'</em>+(s)$</td>
</tr>
<tr>
<td>osc. plane continuity</td>
<td>$\ddot{x}<em>-(t) \cdot (\dot{x}</em>+(t) \times \dot{x}_+(t)) = 0$</td>
<td>$x''<em>-(s) \times x''</em>+(s) = 0$</td>
</tr>
<tr>
<td>$G^2$-continuity</td>
<td>$\kappa_-(t) = \kappa_+(t)$</td>
<td>$x''<em>-(s) = x''</em>+(s)$</td>
</tr>
</tbody>
</table>

Table 3.2: Continuity relations at a point that connects two segments

In section 3.6 some popular curve representation schemes are discussed in more detail.

### 3.1.8 Minimizing the energy

One of the main problems in variational modeling is how the minimal energy curve can be computed efficiently. If a highly nonlinear fairness functional is used, like in [MS92], a brute force method has to be used to compute the minimal energy curve. If a quadratic approximation is used as a fairness functional, and if the constraints imposed on the curve are all linear, the minimization problem becomes a quadratic programming problem (see [Lue89]), which can be solved efficiently. In section 2.5 a number of numerical methods are discussed that can be used to solve the energy minimization problem.

To experiment with variational modeling, an interactive program has been built with uniform B-spline curves as the curve representation (see section 3.6). In this section also several aspects regarding the computation of minimal energy curves are discussed.
3.2 CONSTRAINTS

3.2 Constraints

In variational curve modeling, constraints are an important tool. They can be used to specify certain geometric features of a curve exactly. It is necessary to apply at least some kind of constraints to the end points of a curve, since the energy of an unconstrained curve can be made zero by collapsing the curve to a single point (see also section 3.3). These endpoint constraints may be replaced by some of the design operators that will be introduced in section 3.4. Due to the constraints, the energy minimization of a curve becomes a constrained minimization problem. To keep this problem solvable within a reasonable amount of time, linear constraints are preferable.

Most constraints are of a geometric nature. Often they involve the location of points (positional constraints), or the direction of tangents (directional constraints). The complexity of the constraints depends greatly on the curve representation scheme. For instance, a constraint that is linear in the shape parameters of a parametric curve may very well be highly nonlinear in the shape parameters of an implicit curve. We will use the canonical representation (3.7) for parametric curves that was defined in section 3.1:

\[ x(t) = \sum_{i=0}^{n-1} N_i(t) P_i, \quad t \in [a, b]. \]

Recall that the shape parameters of this curve are the control points \( \{P_i\}_{i=0}^{n-1} \).

3.2.1 Positional constraints

Positional constraints are constraints that put restrictions on the locations of points on a curve. They are expressed in the x, y and z-coordinates of a curve. Three examples will be given, the point to point, the point to plane and the point to line constraint.

**Point to point constraint**

The point to point constraint is the most frequently used constraint. This constraint makes the curve interpolate a given point \( P \), for some parameter value \( t_0 \in [a, b] \):

\[ x(t_0) = P. \quad (3.9) \]

![Figure 3.2: Minimal energy curve with four point to point constraints.](image)
See figure (3.2) for an illustration of this constraint. Since the exact value of the parameter $t_0$ has no direct geometric relevance, this parameter could be taken as a variable in the minimization process. However, the equation (3.9) is normally nonlinear in $t_0$, which is unattractive for the energy minimization. Therefore the parameter $t_0$ is often fixed. For a fixed value $t_0$, the constraint is linear in the control points of the canonical curve representation (3.7):

$$\sum_{i=0}^{n} a_i \mathbf{P}_i = \mathbf{P},$$

with $a_i = N_i(t_0)$. Note that this equation can be split in three independent equations for the $x$, $y$ and $z$-coordinates of the control points $\mathbf{P}_i$. This may be useful in the special case that the minimization problem can be solved for $x$, $y$ and $z$ independently, since the dimension of the problem is lowered with a factor three.

**Point to plane constraint**

A point to plane constraint makes a point of a curve interpolate an arbitrary point of a given plane $V = \{ \mathbf{x} \in \mathbb{R}^3 | \mathbf{n} \cdot \mathbf{x} = d \}$, for some parameter value $t_0 \in [a, b]$:

$$\mathbf{x}(t_0) \in V \Rightarrow \mathbf{n} \cdot \mathbf{x}(t_0) = d \quad (3.10)$$

![Figure 3.3: Minimal energy curve with a point to plane constraint and two point to point constraints.](image)

Substitution of the canonical curve representation (3.7) yields the equation

$$\sum_{i=0}^{n-1} a_i \cdot \mathbf{P}_i = d$$

with $a_i = N_i(t_0)\mathbf{n}$. This equation is linear in the *components* of the control points $\mathbf{P}_i$. Since this constraint is a relation in the $x$, $y$ and $z$-coordinates, the energy minimization problem cannot be solved independently for the $x$, $y$ and $z$-coordinates of the control points. Figure (3.3) demonstrates the point to plane constraint. For the same reason as with the point to point constraint, the parameter $t_0$ is usually fixed.
3.2 CONSTRAINTS

Point to line constraint

A point to line constraint makes a point of a curve interpolate an arbitrary point of a given line \( l = \{ \mathbf{x} \in \mathbb{R}^2 | \mathbf{x} = \mathbf{r} + t \mathbf{s}, t \in \mathbb{R} \} \), for some parameter value \( t_0 \in [a, b] \):

\[
\mathbf{x}(t_0) \in l.
\]  

Figure 3.4: This spiral was created using nineteen point to line constraints (to four different lines) and two point to point constraints.

Since a line can easily be written as the intersection of two planes, the point to line constraint is in fact a combination of two point to plane constraints. For the same reason, a point to point constraint can be written as a combination of three point to plane constraints. Point to line constraints can be very useful. Figure 3.4 shows how a spiral can be modeled using point to line constraints.

3.2.2 Directional constraints

Directional constraints are constraints that put restrictions on the tangent direction at points on a curve. They are expressed in the derivatives of a curve.

Tangent to vector constraint

A tangent to vector constraint prescribes the direction \( \mathbf{t} \) of the tangent vector in a point of a curve with some parameter value \( t_0 \in [a, b] \). In this point, the cross product of the tangent vector with \( \mathbf{t} \) is zero:

\[
\dot{\mathbf{x}}(t_0) \times \mathbf{t} = \mathbf{0}.
\]

This constraint is linear in the components of the control points of the canonical curve representation (3.7):

\[
\sum_{i=0}^{n-1} \mathbf{a}_i \times \mathbf{P}_i = \mathbf{0},
\]

with \( \mathbf{a}_i = \dot{\mathbf{N}}_i(t_0) \mathbf{t} \). An illustration of the tangent to vector constraint is given in figure 3.5. The tangent to vector constraint is often combined with a point to point constraint, to gain control over the exact location where the tangent direction is constrained.
Tangent to plane constraint

The tangent to plane constraint is a relaxed version of the tangent to line constraint. In this case the direction of the tangent vector should be in a given plane, which means that it should be perpendicular to the normal \( \mathbf{n} \) of this plane:

\[
\mathbf{x}(t_0) \cdot \mathbf{n} = 0.
\]

The tangent to plane constraint can be used to force the curve to be tangent to a given plane in a specific point. Like the previous ones, this constraint is linear in the components of the control points of the canonical curve representation (3.7):

\[
\sum_{i=0}^{n-1} a_i \cdot \mathbf{P}_i = 0,
\]

with \( a_i = \mathbf{N}_i(t_0) \mathbf{n} \).

3.2.3 Continuity constraints

When curve modeling is done with composite curves, the resulting curve may fail to be tangent continuous at the junction points of successive segments, depending on the type of curve. To guarantee that the curve has overall smoothness of sufficiently high order, continuity constraints have to be applied. Figure 3.6 shows a minimal energy curve with and without a \( C^1 \) continuity constraint.

We will now give characterizations of the five types of continuity relations listed in table 3.2, applied to a curve with the canonical curve representation (3.7). Let again \( - \) and \( + \) denote left and right limits with respect to the parameter \( t \). Let \( t_0 \) be the parameter of a junction point between two successive segments of a curve. The \( C^1 \) and \( C^2 \) continuity constraints at \( t_0 \) are linear relations in the control points \( \mathbf{P}_i \):

\[
\sum_{i=0}^{n} a_i \mathbf{P}_i = 0,
\]
3.2 CONSTRAINTS

Figure 3.6: The figure shows a curve consisting of two segments, which are joined at the point \( x(t_0) \). The left curve was generated without continuity constraints. The right curve was generated with a \( C^1 \) continuity constraint at the point with parameter \( t_0 \).

with \( a_i = \hat{N}_{i+}(t_0) - \hat{N}_{i-}(t_0) \) and \( a_i = \hat{N}_{i+}(t_0) - \hat{N}_{i-}(t_0) \) respectively. In general, the \( G^1 \), \( G^2 \) and osculation plane continuity constraint lead to quadratic or higher order equations in the components of the control points \( P_i \).

For variational modeling, the last three types of continuity are preferable, since they are of a geometric nature. The \( C^1 \) and \( C^2 \) constraints are in fact too strong. They do not only affect the shape of the curve, but also the parametrization, which costs extra degrees of freedom. This can be compensated for by adding extra curve segments. Because of their linearity, \( C^1 \) and \( C^2 \) continuity constraints are more often used than the other three.

Whenever possible, continuity constraints should be avoided. They typically lead to a large set of constraints on the curve, which is unattractive for the minimization problem. Therefore (basis-)functions \( \hat{N}_i \) are preferred that already have the desired order of continuity.

3.2.4 Other constraints

The constraints that were mentioned so far are among the most practical ones. Of course there exist many others. Here we will list a few of them.

**curvature and torsion interpolation**

The curvature \( \kappa \) or the torsion \( \tau \) may be prescribed in a point of the curve. They both lead to very complicated equations in the control points of the canonical curve representation (3.7).

**quadratic curve interpolation**

A point of the curve may be forced to interpolate a quadratic curve \( f\): 

\[
 f(x, y, z) = \sum_{|i+j+k| \leq 2} a_{ijk} x^i y^j z^k = 0.
\]

An example is the point to sphere constraint 

\[
 \| x(t_0) - m \|^2 = r^2,
\]
where the point with parameter \( t_0 \) must interpolate a sphere with center \( m \) and radius \( r \). These constraints are quadratic in the components of the control points of the canonical curve representation (3.7).

**constraints on a segment of a curve**

A constraint doesn’t have to be restricted to one point of the curve; it can also be applied to a whole segment of the curve. As an example, a part of the curve may be forced to be planar. The feasibility of this type of constraints depends strongly on the curve representation. For instance, the functions \( N_i \) of the canonical curve representation (3.7) should at least have local support and occupy the same function space as the curve to be interpolated.

**inequality constraints**

Inequality constraints are difficult to maintain during the energy minimization, but they can be of great practical value. For instance, a point or a part of a surface may be restricted to lie on one side of a plane, or inside a sphere.
3.3 Internal energy

The internal energy of a curve is the part of the total energy (3.6) that depends only on properties of the curve itself, like the curvature. It determines the global shape characteristics of the curve. In this section one particular choice for the internal energy is treated, namely a linear combination of the bend energy (3.1) and the stretch energy. This functional (which is also known as the nonlinear splines under tension model (3.3), see section 3.1) is often used, see ([CG91], [WW92]) etc. The internal energy $E_{\text{int}}$ of a curve $x$ can thus be written as

$$E_{\text{int}}(x) = \alpha E_{\text{stretch}}(x) + (1 - \alpha)E_{\text{bend}}(x),$$

(3.12)

with $\alpha$ a value in the interval [0,1]. There is no special reason for taking a convex combination of the bend and stretch energy in this formula, except that it is slightly more general than formula for nonlinear splines under tension (3.3). In general, decreasing $\alpha$ causes the length of the curve to shrink, and increasing $\alpha$ causes a more balanced spreading of the curvature over the curve. This is illustrated in figure 3.7.

![Figure 3.7: Increasing the factor $\alpha$ reduces the length of the curve. Decreasing the factor $\alpha$ reduces the bending of the curve.](image)

In the next two sections, the bend and stretch energy and some approximations to them will be discussed. The sections 3.3.3 and 3.3.4 deal with some theoretical properties. The general form of unconstrained curves with minimal bend or stretch energy will be derived, and we will investigate how the bend and stretch energy behave under scaling and parameter transformations. The latter is important for the choice of the weight factor $\alpha$ in (3.12).

The bend and stretch energy will be expressed in the control points of the canonical curve representation (3.7)

$$x(t) = \sum_{i=0}^{n-1} P_i N_i(t),$$

(3.13)

that was introduced in section (3.1).

3.3.1 Bend energy

The bend (or strain) energy of a curve $x$ is the integral of the squared curvature $\kappa^2$ (see also section 3.1.2):

$$E_{\text{bend-strain}}(x) = \int_a^b \kappa^2(t) \|\dot{x}(t)\| dt.$$

(3.14)
It measures how much the curve is bent. A problem with this integral is that it is a very complicated expression in the control points $P_i$ of the canonical curve representation (3.7). For most practical situations an approximation is needed. Two of these approximations will be given. A simple one, and a more accurate data dependent one.

**Simple approximation**

A simple approximation for the bend energy (3.14) can be derived as follows. Substitution of the expression for the curvature (2.9) gives

$$E_{\text{bend, exact}}(x) = \int_a^b \kappa^2(t) \|\vec{x}(t)\| dt = \int_a^b \left(\frac{\vec{x}(t) \cdot \vec{x}(t)}{\|\vec{x}(t)\|^3} - \frac{(\dot{x}(t) \cdot \ddot{x}(t))^2}{\|\vec{x}(t)\|^5}\right) dt.$$  

If we assume that the curve has a nearly arc length parametrization ($\ddot{x} \cdot \dot{x} \approx 1 \Rightarrow \dot{x} \cdot \ddot{x} \approx 0$) the following approximation is found:

$$E_{\text{bend, simple}}(x) = \int_a^b \|\ddot{x}(t)\|^2 dt.  \quad (3.15)$$

This approximation is often used (see for instance [FG92], [GC95]). However, this approximation can be very poor, which is illustrated in the figures 3.8 and 3.9. The extreme differences between the two functions are caused by the fact that the curve is not even nearly arc length parametrized (see the second plot in figure 3.9). Another problem is that the integral (3.15) depends on the parametrization.

![Figure 3.8: A uniform B-spline curve $x$ with the property that the squared second derivative $\|\ddot{x}(t)\|^2$ is a poor approximation of the squared curvature $\kappa^2(t)$. The small circles are the control points.](image)

The great benefit of the approximation (3.15) is that it is quadratic in the control points $P_i$ of the canonical curve representation (3.7):

$$E_{\text{bend, simple}}(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} P_i \cdot P_j,$$
3.3 Internal Energy

![Graphs showing the graphs of $\kappa^2(t)$ and $\|\tilde{x}(t)\|^2$ (dashed) for the curve shown in the previous figure. Note the extreme differences between the two! The right figure shows the graph of $\|\tilde{x}(t)\|^2$.]

with

$$a_{ij} = \int_a^b \tilde{N}_i(t) \tilde{N}_j(t) dt.$$  

This is a very desirable form, since quadratic functions can be efficiently minimized (see section 2.5 and 3.6). The matrix $A_{\text{bend simple}} = [a_{ij}]$ has the following properties:

- $A_{\text{bend simple}}$ is symmetric.
- $A_{\text{bend simple}}$ is positive semidefinite. This follows from the fact that the bend energy $E_{\text{bend simple}}(\mathbf{x})$ is always positive.
- If the basis functions $N_i$ have local support, then the matrix $A_{\text{bend simple}}$ is banded.

**Data dependent approximation**

There is a big gap between the exact expression for the bend energy (3.14) (which is complicated) and the approximation (3.15) (which is simple). In [Adl93] an approximation is proposed that is less easy to compute, but still quadratic in the control points of the canonical curve (3.7). This approximation uses a reference curve $\mathbf{c}$, which is a curve that is somehow close to the curve $\mathbf{x}$. It is given by

$$E_{\text{bend data}}(\mathbf{x}; \mathbf{c}) = \int_a^b \sum_{i=1}^3 \text{Hess}_c(x_i)^2 \|\dot{\mathbf{c}}\| dt$$

$$= \int_a^b \Delta_c(\mathbf{x}) \cdot \Delta_c(\mathbf{x}) \|\dot{\mathbf{c}}\| dt,$$

(3.16)

where the Laplace operator $\Delta$ is given by $\Delta_c(\mathbf{x}) = \frac{\ddot{\dot{\mathbf{x}}}}{\dot{\mathbf{c}} \cdot \dot{\mathbf{c}}} - \frac{(\dot{\mathbf{c}} \cdot \ddot{\dot{\mathbf{x}}})}{(\dot{\mathbf{c}} \cdot \dot{\mathbf{c}})^2}$. In section (2.4) it is shown that (3.16) is a good approximation of (3.14) provided that the distance between $\mathbf{c}$ and $\mathbf{x}$ is small. This distance is measured by the norm $\|\|_e$, that was defined in (2.19). It is
worth noting that the simple approximation (3.15) is a special case of the data dependent approximation (3.16). For an arbitrary arc length parametrized curve \( \mathbf{c} \) (for example the curve \( \mathbf{c}(t) = (t, 0, 0), \ t \in [a, b] \)), the expression (3.16) can be simplified to (3.15).

If the canonical curve representation (3.7) is substituted into (3.16), a quadratic expression in the control points \( \mathbf{P}_i \) is found:

\[
E_{\text{bend, data}}(\mathbf{x}; \mathbf{c}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j,
\]

with

\[
a_{ij} = \int_a^b \text{Hess}_c(N_i) \text{Hess}_c(N_j) \| \dot{\mathbf{c}} \| \, dt.
\]

The coefficients \( a_{ij} \) normally cannot be computed analytically. They have to be approximated using the numerical integration routines given in section (2.5.5). The matrix \( A_{\text{bend, data}} = [a_{ij}] \) has the same properties as the matrix \( A_{\text{bend, simple}} \).

### 3.3.2 Stretch energy

The stretch energy measures the length of a curve. It is given by

\[
E_{\text{stretch}}(\mathbf{x}) = \int_a^b \| \dot{\mathbf{x}}(t) \| \, dt.
\]

(3.17)

This expression is the square root of a quadratic expression in the control points \( \mathbf{P}_i \) of the canonical curve representation (3.7). Although this is considerably easier than the corresponding expression for the bend energy, it is still too complicated for most practical applications. Again a simple approximation and a more difficult one using a reference curve will be given.

The following approximation is often used:

\[
E_{\text{stretch, simple}}(\mathbf{x}) = \int_a^b \| \dot{\mathbf{x}}(t) \|^2 \, dt.
\]

(3.18)

If we substitute the canonical curve representation (3.7), a quadratic expression in the control points \( \mathbf{P}_i \) is found:

\[
E_{\text{stretch, simple}}(\mathbf{x}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j,
\]

with \( a_{ij} = \int_a^b \dot{N}_i(t) \dot{N}_j(t) \, dt \). Like the simple approximation for the bend energy (3.15), this approximation can be rather poor. A better approximation is the following, defined in terms of a reference curve \( \mathbf{c} \):

\[
E_{\text{stretch, data}}(\mathbf{x}; \mathbf{c}) = \int_a^b \sum_{i=1}^3 \text{grad}_c(x_i) \cdot \text{grad}_c(x_i) \, dt
\]

\[
= \int_a^b \frac{\ddot{\mathbf{x}} \cdot \ddot{\mathbf{x}}}{\| \mathbf{c} \|} \| \mathbf{c} \| \, dt.
\]

(3.19)
3.3 Internal Energy

In section (2.4) it is shown that this functional is a good approximation of (3.17) if the norm \( \| \|_c \) of the difference between \( x \) and \( c \) is small. If we substitute the canonical curve representation (3.7), a quadratic expression in the control points \( P_i \) is found:

\[
E_{\text{stretch, data}}(x; c) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} P_i \cdot P_j,
\]

with \( a_{ij} = \int_0^L \frac{\hat{N}_i \cdot \hat{N}_j}{\| \hat{c} \|} dt \). Again, the coefficients \( a_{ij} \) have to be approximated using numerical integration. Note that for an arc length parametrized curve (\( \| \dot{x}(t) \| \equiv 1 \)), the approximation is exact.

3.3.3 Curves with minimal bend or stretch energy

In this section unconstrained curves with minimal bend and stretch energy are discussed. Although unconstrained curves are not practically applicable, we study them to gain a better understanding of constrained curves with minimal energy.

Curves with minimal stretch energy

The curves with minimal stretch energy are easy to characterize: the stretch energy (3.17) and the approximations (3.18) and (3.19) of a curve \( x \) are zero (and have thus minimal energy) if and only if \( \ddot{x} \equiv 0 \). This is the case when the curve is a single point.

Curves with minimal bend energy

For curves with minimal bend energy the situation is somewhat more difficult. The bend energy (3.14) of a curve \( x \) is zero if and only if \( \kappa(t) \equiv 0 \). This is equivalent with saying that \( x \) is a part of a straight line. The approximation (3.15) is zero if and only if \( \ddot{x} = \mathbf{0} \).

From this it follows that \( x \) is part of a straight line. Note that the reverse is not true: the fact that a curve \( x \) is part of a straight line does not guarantee that \( \ddot{x} = \mathbf{0} \). Finally, the approximation (3.16) is zero if and only if \( \Lambda_c(x) \equiv \mathbf{0} \). This is equivalent with

\[
\ddot{x} - \left( \frac{\dot{c} \cdot \dot{c}}{\dot{c} \cdot \dot{c}} \right) \dot{x} = \mathbf{0}.
\]  

The general solution to this differential equation is given by

\[
x(t) = C_0 \int_0^t (\dot{c} \cdot \dot{c})^{1/2} d\tau + C_1,
\]

with \( C_0 \in \mathbb{R} \) and \( C_1 \in \mathbb{R}^3 \) two arbitrary constants. The equation (3.20) implies that the vectors \( \ddot{x} \) and \( \dot{x} \) are parallel, and thus the solution is part of a straight line. Again, the reverse is not true: the fact that a curve \( x \) lies on a straight line does not imply that \( \Lambda_c(x) = \mathbf{0} \).
3.3.4 The effect of scaling and parameter transformations

In the sections (3.3.1) and (3.3.2) the bend and stretch energy with their approximations were treated. Now we will investigate how these quantities behave under scaling of a curve and under a linear parameter transformation.

Let \( \mathbf{x} \) be a curve defined on the interval \([a, b]\) and let \( \mathbf{c} \) be a reference curve for \( \mathbf{x} \) defined on the same interval. When \( \mathbf{x} \) is compared with the scaled curve \( K \mathbf{x} \), the following relations are found:

\[
\begin{align*}
E_{\text{stretch\_exact}}(K \mathbf{x}) &= KE_{\text{stretch\_exact}}(\mathbf{x}) \\
E_{\text{stretch\_simple}}(K \mathbf{x}) &= K^2 E_{\text{stretch\_simple}}(\mathbf{x}) \\
E_{\text{stretch\_data}}(K \mathbf{x}; \mathbf{c}) &= KE_{\text{stretch\_data}}(\mathbf{x}; \mathbf{c}) \\
E_{\text{bend\_exact}}(K \mathbf{x}) &= (1/K) E_{\text{bend\_exact}}(\mathbf{x}) \\
E_{\text{bend\_simple}}(K \mathbf{x}) &= K^2 E_{\text{bend\_simple}}(\mathbf{x}) \\
E_{\text{bend\_data}}(K \mathbf{x}; \mathbf{c}) &= (1/K) E_{\text{bend\_data}}(\mathbf{x}; \mathbf{c})
\end{align*}
\]

Scaling the curve with a factor \( K \) will result in a curve for which the stretch energy \( E_{\text{stretch\_exact}} \) is multiplied with \( K \) and for which the bend energy \( E_{\text{bend\_exact}} \) is multiplied with \( 1/K \). Therefore, first scaling a curve and then minimizing its energy (3.12) will result in a curve with a different shape than first minimizing its energy and then scaling it. To prevent this, the weight factor \( (1 - \alpha) \) for the bend energy should be multiplied with \( K^2 \).

The effect of scaling on the approximations is exactly the same. On the contrary, for the simple approximations \( E_{\text{bend\_simple}} \) and \( E_{\text{stretch\_simple}} \) scaling doesn’t have much effect. In this case the stretch and bend energy are multiplied with the same factor \( K^2 \).

When the curve \( \mathbf{x} \) is compared with the curve \( \mathbf{x} \circ \phi \), which is obtained from \( \mathbf{x} \) by applying the linear parameter transformation \( \phi : t \mapsto \gamma t + \delta \), the following relations are found:

\[
\begin{align*}
E_{\text{stretch\_exact}}(\mathbf{x} \circ \phi) &= E_{\text{stretch\_exact}}(\mathbf{x}) \\
E_{\text{stretch\_simple}}(\mathbf{x} \circ \phi) &= \frac{1}{\gamma} E_{\text{stretch\_simple}}(\mathbf{x}) \\
E_{\text{stretch\_data}}(\mathbf{x} \circ \phi; \mathbf{c} \circ \phi) &= E_{\text{stretch\_data}}(\mathbf{x}; \mathbf{c}) \\
E_{\text{bend\_exact}}(\mathbf{x} \circ \phi) &= E_{\text{bend\_exact}}(\mathbf{x}) \\
E_{\text{bend\_simple}}(\mathbf{x} \circ \phi) &= \frac{1}{\gamma} E_{\text{bend\_simple}}(\mathbf{x}) \\
E_{\text{bend\_data}}(\mathbf{x} \circ \phi; \mathbf{c} \circ \phi) &= E_{\text{bend\_data}}(\mathbf{x}; \mathbf{c})
\end{align*}
\]

For the exact expressions \( E_{\text{bend\_exact}} \) and \( E_{\text{stretch\_exact}} \) and for the approximations \( E_{\text{bend\_data}} \) and \( E_{\text{stretch\_data}} \) a parameter transformation of \( \mathbf{x} \) doesn’t have any effect. This is not surprising, since these energies are defined in terms of geometric quantities. As can be seen from (3.21), the simple approximations \( E_{\text{bend\_simple}} \) and \( E_{\text{stretch\_simple}} \) are sensitive for a (linear) parameter transformation. This is clearly an unwanted effect, since a parameter transformation doesn’t change the shape of a curve. The following rule can be used to eliminate the
3.3 **INTERNAL ENERGY**

effect of a linear parameter transformation: always replace the weight factor $\alpha$ for the bend energy with $\alpha(b-a)$ and the weight factor for the stretch energy $(1-\alpha)$ with $(1-\alpha)(b-a)^2$, where $[a, b]$ is the interval over which the curve is defined. This rule is applied in the implementation of the modeling program that is described in section (3.6).
3.4 External energy

The external energy of a curve consists of a number of separate energy terms. Each of these terms has a certain deformation effect on the curve. If the addition of such an energy term has a predictable effect on a minimal energy curve, it can be offered as a design tool to a curve modeler. In this section a number of these design operators or external energy operators are introduced, together with their intended effect. The goal of these operators is to give the designer more control over the shape of the minimal energy curve, which will make it easier to create a curve that meets the design intent.

The external energy $E_{ext}$ is a weighted sum of the contributions from the external energy operators. It can therefore be written as

$$E_{ext}(x) = \sum_i w_i E_{external,i}(x).$$

Increasing a weight factor $w_i$ will make the effect of the corresponding operator stronger. In a modeling application these weights should therefore be under control of the user. In section (3.3.4) it was pointed out that the weight factors for the bend and stretch energy should be adapted when a curve or its parametrization is scaled. The same holds true for the weights $w_i$ of the external energy operators. This topic is further addressed in section (3.4.4).

Most of the operators have some kind of attracting behavior (for example attraction towards a given point in space). For all these ‘attractors’ a repelling counterpart can be defined that has the opposite effect. Like the constraints, the design operators will be subdivided in positional operators (operators that act on the location of points of the curve) and directional operators (operators that act on tangent directions of the curve). Another possible way of subdividing the operators is to distinguish between operators that act on a single point of the curve, and operators that act on a specific segment of the curve.

As was done in the previous sections, the external energy operators will be expressed in the control points of the canonical curve representation

$$x(t) = \sum_{i=0}^{n-1} N_i(t)P_i.$$  

3.4.1 Positional attractors

Positional attractors are design operators that exert some kind of attracting force on the location of points on a curve. First some positional attractors will be defined that are closely related to the positional constraints. Then a more sophisticated attractor is defined that is particularly useful for 3D curves (the profiler).

Point to point attractor

The point to point attractor is a design operator that pulls a curve towards a given point in space, as illustrated in figure 3.10. The grey dot is the point that attracts the curve; the point on the curve to which the attractor applies is graphically represented by a circle.
3.4 EXTERNAL ENERGY

Figure 3.10: Left: the point $x(t_0)$ on the dashed curve is attracted towards the point $p$ using a point to point attractor. Right: if the weight of the attractor is increased, the effect of the operator becomes stronger.

The energy term that corresponds with the point to point attractor is the squared distance between the point $p$ and a point $x(t_0)$ on the curve:

$$E_{\text{point to point}}(x) = d(x(t_0), p)^2 = ||x(t_0) - p||^2.$$  \hspace{1cm} (3.22)

Ideally, the parameter $t_0$ of the point of the curve should be free, so that the true distance $d(x, p)$ between the curve $x$ and the point $p$ is taken into account. However, for practical implementations this is not feasible. The minimal value of the expression (3.22) is of course obtained when $p$ and a $x(t_0)$ coincide. Other energy terms may prevent the point to point attractor term from reaching this minimum.

The point to point attractor is encountered more often in the literature. In ([FG92]) the term (3.22) is called a spring force, for its resemblance with the potential energy of a spring. The energy term for the point to point attractor is quadratic in the control points of the canonical curve representation (3.7):

$$E_{\text{point to point}}(x) = \sum_{i,j} a_{ij} P_i P_j + \sum_i b_i P_i \cdot p + c,$$

with

$$a_{ij} = N_i(t_0) N_j(t_0), \quad b_i = -2N_i(t_0), \quad c = p \cdot p.$$

**Point to line attractor**

The *point to line attractor* is a design operator that pulls a curve towards a given line $l$ in space. The energy term that corresponds with the point to line attractor is the squared distance between the line $l$ and a point $x(t_0)$ on the curve:

$$E_{\text{point to line}}(x) = d(x(t_0), l)^2 = ||Ax(t_0) + b||^2,$$

where $A \in \mathbb{R}^{3 \times 3}$ and $b$ are given by (2.42). This expression is quadratic in the components of the control points of the canonical curve representation (3.7):

$$E_{\text{point to line}}(x) = \sum_{i,j} P_i^T M_{ij} P_j + \sum_i P_i \cdot B_i + c,$$
with

\[ M_{ij} \in \mathbb{R}^{3 \times 3} = N_i(t_0)N_j(t_0)A^TA, \quad B_i = 2N_i(t_0)A^Tb, \quad c = b \cdot b. \]

**Point to plane attractor**

The *point to plane attractor* is a design operator that pulls a point of a curve towards a given plane \( V = \{ \mathbf{x} \in \mathbb{R}^3 | \mathbf{n} \cdot \mathbf{x} = d \} \). The energy term that corresponds with the point to plane attractor is the squared distance between the plane \( V \) and a point \( \mathbf{x}(t_0) \) on the curve:

\[ E_{\text{point to plane}}(\mathbf{x}) = d(\mathbf{x}(t_0), V)^2 = \frac{\left| \mathbf{n} \cdot \mathbf{x}(t_0) - d \right|^2}{\mathbf{n} \cdot \mathbf{n}}. \]

This expression is also quadratic in the components of the control points of the canonical curve representation (3.7):

\[ E_{\text{point to plane}}(\mathbf{x}) = \sum_{i,j} \mathbf{P}_i^T A_{ij} \mathbf{P}_j + \sum_i \mathbf{P}_i B_i + c, \]

with

\[ A_{ij} \in \mathbb{R}^{3 \times 3} = N_i(t_0)N_j(t_0)\mathbf{n}\mathbf{n}^T/(\mathbf{n} \cdot \mathbf{n}) \quad B_i = -2dN_i(t_0)\mathbf{n}/(\mathbf{n} \cdot \mathbf{n}) \quad c = d^2/(\mathbf{n} \cdot \mathbf{n}). \]

**Segment to point attractor**

The *segment to point attractor* is the same as a point to point attractor, except that it operates on a segment of the curve with a fixed parameter interval \([t_{\text{min}}, t_{\text{max}}] \). The energy term that corresponds with the segment to point attractor is the integral over the squared distance between the curve \( \mathbf{x} \) and the point \( \mathbf{p} \):

\[ E_{\text{segment to point}}(\mathbf{x}) = \int_{t_{\text{min}}}^{t_{\text{max}}} \| \mathbf{x}(t) - \mathbf{p} \|^2 \, dt. \]

This expression is quadratic in the control points of the canonical curve representation (3.7):

\[ E_{\text{segment to point}}(\mathbf{x}) = \sum_{i,j} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j + \sum_i b_i \mathbf{P}_i \cdot \mathbf{p} + c, \]

with

\[ a_{ij} = \int_{t_{\text{min}}}^{t_{\text{max}}} N_i(t)N_j(t) \, dt, \quad b_i = -2 \int_{t_{\text{min}}}^{t_{\text{max}}} N_i(t) \, dt, \quad c = \mathbf{p} \cdot \mathbf{p}. \]

**Segment to line attractor**

The *segment to line attractor* attracts a segment of the curve with a fixed parameter interval \([t_{\text{min}}, t_{\text{max}}] \) towards a given line \( l \) in space, as illustrated in figure 3.11.

Note that the distance between the curve segment and the line can be reduced by increasing the weight factor of the operator. The energy term that corresponds with the segment to line attractor is the integral over the squared distance between the line \( l \) and the curve \( \mathbf{x} \):

\[ E_{\text{segment to line}}(\mathbf{x}) = \int_{t_{\text{min}}}^{t_{\text{max}}} \| A\mathbf{x}(t) + \mathbf{b} \|^2 \, dt, \]
3.4 EXTERNAL ENERGY

Figure 3.11: Left: the segment $[t_0, t_1]$ of the curve is attracted towards the line $c$ using a segment to line attractor. Middle and right: the result of the operator is shown from two different views.

where $A \in \mathbb{R}^{3 \times 3}$ and $b$ are given by (2.42). The energy term for the segment to line attractor is quadratic in the components of the control points of the canonical curve representation (3.7):

$$E_{\text{segment to line}}(x) = \sum_{i,j} P^T_i M_{ij} P_j + \sum_i P_i B_i + c$$

with

$$M_{ij} \in \mathbb{R}^{3 \times 3} = \int_{t_{\text{start}}}^{t_{\text{end}}} N_i(t)N_j(t)dt A^T A, \quad B_i = 2 \int_{t_{\text{start}}}^{t_{\text{end}}} N_i(t)dt A^T b, \quad c = b \cdot b.$$

Segment to plane attractor

The segment to plane attractor attracts a segment of the curve with a fixed parameter interval $[t_{\text{min}}, t_{\text{max}}]$ towards a given plane $V = \{x \in \mathbb{R}^3 | n \cdot x = d\}$. The energy term that corresponds with the segment to plane attractor is the integral over the squared distance between the plane $V$ and the curve $x$:

$$E_{\text{segment to plane}}(x) = \int_{t_{\text{min}}}^{t_{\text{max}}} \frac{n \cdot x(t) - d^2}{n \cdot n} dt.$$

This expression is also quadratic in the components of the control points of the canonical curve representation (3.7):

$$E_{\text{segment to plane}}(x) = \sum_{i,j} P^T_i A_{ij} P_j + \sum_i P_i B_i + c$$

with

$$A_{ij} \in \mathbb{R}^{3 \times 3} = \int_{t_{\text{start}}}^{t_{\text{end}}} N_i(t)N_j(t)dt nn^T/(n \cdot n), \quad B_i = -2d \int_{t_{\text{start}}}^{t_{\text{end}}} N_i(t)dt n/(n \cdot n), \quad c = d^2/(n \cdot n).$$
CHAPTER 3 CURVES

Segment to curve attractor

The segment to curve attractor is a design operator that attracts a segment of the curve towards another curve. The part of the curve $\mathbf{x}$ in the interval $[t_{\min}, t_{\max}]$ can be attracted to the curve $\mathbf{c}$ by an energy term of the following form:

$$E_{\text{segment\to\text{curve}}} (\mathbf{x}) = \int_{t_{\min}}^{t_{\max}} \| \mathbf{x}(t) - \mathbf{c}(t)\|^2 dt.$$  

Suppose that $\mathbf{x}$ and $\mathbf{c}$ both have the canonical representation (3.7), where $\mathbf{x}$ has control points $\{\mathbf{P}_i\}$ and $\mathbf{c}$ has control points $\{\mathbf{Q}_i\}$. Then

$$E_{\text{segment\to\text{curve}}} (\mathbf{x}) = \sum_{i,j} a_{ij} \mathbf{P}_i \mathbf{P}_j + \sum_i \mathbf{P}_i \mathbf{B}_i + c,$$

with

$$a_{ij} = \int_{t_{\min}}^{t_{\max}} N_i(t) N_j(t) dt, \quad \mathbf{B}_i = -2 \sum_j \int_{t_{\min}}^{t_{\max}} N_i(t) N_j(t) dt \mathbf{Q}_j,$$

$$c = \sum_{i,j} \int_{t_{\min}}^{t_{\max}} N_i(t) N_j(t) dt \mathbf{Q}_i \mathbf{Q}_j.$$

Profiler

When modeling a three dimensional curve, a designer is usually confronted with a two dimensional view of this curve, which is in fact a projection of the curve in a plane. A typical modeling operation is to modify this projection of the curve. The profiler is a modeling operator that supports this. It attracts the parallel projection of a part of the curve in a given plane $V$ towards another curve (a ‘profile’) in this plane. This is illustrated in figure 3.12. The profile is shown in grey, the part of the curve to which the operator applies is between the circles (which coincide with interpolation points). The profiler attracts the parallel projection of a segment of the curve with a fixed parameter interval $[t_{\min}, t_{\max}]$ into a given plane $V = \{ \mathbf{x} \in \mathbb{R}^3 | \mathbf{n} \cdot \mathbf{x} = d \}$ towards a given curve $\mathbf{c}$ in $V$. Its energy term is given by

$$E_{\text{profiler}} (\mathbf{x}) = \int_{t_{\min}}^{t_{\max}} \| \text{Proj}(\mathbf{x}(t)) - \mathbf{c}(t)\|^2 dt,$$

where $\text{Proj}(\mathbf{x})$ is the parallel projection of $\mathbf{x}$ in $V$ (see section 2.6.3).

Suppose that $\mathbf{x}$ and $\mathbf{c}$ both have the canonical representation (3.7), where $\mathbf{x}$ has control points $\{\mathbf{P}_i\}$ and $\mathbf{c}$ has control points $\{\mathbf{Q}_i\}$. The energy term of the profiler is quadratic in the components of the control points $\{\mathbf{P}_i\}$:

$$E_{\text{profiler}} (\mathbf{x}) = \sum_{i,j} \mathbf{P}_i^T \mathbf{M}_{ij} \mathbf{P}_j + \sum_i \mathbf{P}_i \mathbf{B}_i + c$$

with

$$\mathbf{M}_{ij} \in \mathbb{R}^{3 \times 3} = \int_{t_{\min}}^{t_{\max}} N_i(t) N_j(t) dt A^T A,$$
3.4 EXTERNAL ENERGY

Figure 3.12: The profiler is an operator that attracts the parallel projection of a curve in a
given plane towards a curve (drawn in gray) in this plane.

\[
\begin{align*}
B_i &= \int_{t_{\text{min}}}^{t_{\text{max}}} N_i(t)dt Ab - \sum_j \int_{t_{\text{min}}}^{t_{\text{max}}} N_i(t)N_j(t)dt A^Tb, \\
c &= b \cdot b - \sum_i \int_{t_{\text{min}}}^{t_{\text{max}}} N_i(t)dt b Q_i + \sum_{i,j} \int_{t_{\text{min}}}^{t_{\text{max}}} N_i(t)N_j(t)dt Q_i Q_j,
\end{align*}
\]

where \( A \in \mathbb{R}^{3 \times 3} \) and \( b \) are the parameters of the mapping \( \text{Proj} \), given by (2.44).

3.4.2 Directional attractors

Directional attractors are design operators that exert some kind of attracting force on the
tangent direction of points on a curve. The examples below are closely related to the direc-
tional constraints from section 3.2.

Tangent to line attractor

The tangent to line attractor attracts the tangent vector in a point of the curve with a
fixed parameter \( t_0 \) towards a given direction \( d \). The quantity \( \|\hat{x}(t_0) \times d\|^2 \) is a measure for
the orthogonality between the tangent vector \( \hat{x}(t_0) \) and the vector \( d \). Therefore, a suitable
energy term that tries to direct the tangent of the curve at the point with parameter \( t_0 \) into
the specified direction is

\[
E_{\text{tangent to line}}(x) = \|\hat{x}(t_0) \times d\|^2.
\]
CHAPTER 3 CURVES

This expression is quadratic in the control points of the canonical curve representation (3.7):

$$ E_{tangent_{to\ line}}(x) = \sum_{i,j} P_i^T A_{ij} P_j $$

with

$$ A_{ij} \in \mathbb{R}^{3 \times 3} = \hat{N}_i(t_0)\hat{N}_j(t_0) \left( d \cdot d_3 - dd^T \right). $$

Tangent to plane attractor

The tangent to plane attractor attracts the tangent vector in a point of the curve with a fixed parameter $t_0$ towards a given plane through the origin $V = \{ x \in \mathbb{R}^3 | n \cdot x = 0 \}$. The quantity $(\dot{x}(t_0) \cdot n)^2$ is a measure for the deviation of the tangent vector $\dot{x}(t_0)$ from the plane $V$. Therefore, a suitable energy term that tries to direct the tangent of the curve at the point with parameter $t_0$ towards the given plane $V$ is

$$ E_{tangent_{to\ plane}}(x) = (\dot{x}(t_0) \cdot n)^2. $$

This expression is quadratic in the components of the control points of the canonical curve representation (3.7):

$$ E_{tangent_{to\ plane}}(x) = \sum_{i,j} P_i^T A_{ij} P_j $$

with

$$ A_{ij} \in \mathbb{R}^{3 \times 3} = \hat{N}_i(t_0)\hat{N}_j(t_0)nn^T. $$

Tangent segment to line attractor

The tangent segment to line attractor attracts the tangent direction in a segment on the curve with a fixed parameter interval $[t_{min}, t_{max}]$ towards a given direction $d$. Like in the point case, the quantity $\|\dot{x}(t) \times d\|^2$ is a measure for the orthogonality between the tangent vector at the point on the curve with parameter $t$ and the vector $d$. Therefore, a suitable energy term that tries to direct the tangent of the curve in the parameter interval $[t_{min}, t_{max}]$ into the specified direction is

$$ E_{tangent_{segment\ to\ vector}}(x) = \int_{t_{min}}^{t_{max}} \|\dot{x}(t) \times d\|^2 \, dt. $$

Tangent segment to plane attractor

The tangent segment to plane attractor attracts the normal in a segment on the curve with a fixed parameter interval $[t_{min}, t_{max}]$ towards a given plane through the origin $V = \{ x \in \mathbb{R}^3 | n \cdot x = 0 \}$. Analogous to the point case, the following energy term has the desired effect:

$$ E_{tangent_{segment\ to\ plane}}(x) = \int_{t_{min}}^{t_{max}} (\dot{x}(t) \cdot n)^2 \, dt. $$
3.4 EXTERNAL ENERGY

3.4.3 Repellors

The external energy operators that we encountered so far all have some attracting behavior. For each of these operators a repelling counterpart exists with the opposite effect.

How repelling behavior can be achieved will be illustrated for the case of a point repellor. A point repellor is a modeling operator that can be compared with an electrical point charge in an electric field. It is a point in space that exerts a repelling force on a specific point of the curve. The force is parallel to the line between the two points and its magnitude decreases when the distance between the two points increases. One way to simulate the repelling behavior of the point repellor is to take a point attractor with a negative weight factor. However, doing this would introduce negative energy terms. If the weight factor is made too negative, the energy can be made arbitrarily low by moving the curve towards infinity. Another way to simulate repelling behavior is to attract the point towards the opposite direction. Although this may give satisfactory results, it doesn’t model the earlier described behavior. The best way of modeling repelling behavior is to take the reciprocal of the attractor energy term. In the case of the point repellor this becomes

\[ E_{\text{point,repellor}}(\mathbf{x}) = \frac{1}{d(\mathbf{x}(t_0), \mathbf{p})^2}. \]  

(3.23)

From a comparison with the point to point attractor (3.22) it follows that this expression is the reciprocal of a quadratic function in the control points of the canonical curve representation (3.7). This is a disadvantage, since if the energy of a curve contains non-quadratic terms, the minimization becomes significantly more laborious. The point repellor expression (3.23) has one more problem. The point \( \mathbf{x}(t_0) \) can move away from the point \( \mathbf{p} \) not only along the current direction \( \mathbf{x}(t_0) - \mathbf{p} \), but also by sliding along the curve. This phenomenon is illustrated in the left and middle curve of figure 3.13. It will reduce the repellor energy, but the shape of the curve may hardly change.

![Diagram showing point repellors](image)

Figure 3.13: A problem with point repellors is that they may reduce their energy by changing the parametrization of the curve (left and middle). In this case the problem can be solved by combining the repellors with a point to line constraint (right).

This problem is caused by taking the distance \( d(\mathbf{x}(t_0), \mathbf{p}) \) between the point \( \mathbf{p} \) and a point on the curve with a specific parameter. Ideally this term should be replaced by the real distance \( d(\mathbf{x}, \mathbf{p}) \) between the curve \( \mathbf{x} \) and the point \( \mathbf{p} \), but this is not feasible in practical
applications. A solution to this problem is to combine the point repeller with a constraint that keeps $\mathbf{x}(t_0)$ on the line through $\mathbf{p}$ and the current value of $\mathbf{x}(t_0)$. This will give the repeller a more predictable behavior, as illustrated by the right curve in figure 3.13.

As already mentioned, repellers exist for all the attractors given before. Some of them are listed below.

$$E_{\text{point from point}}(\mathbf{x}) = \frac{1}{d(\mathbf{x}(t_0), \mathbf{p})^2}$$

$$E_{\text{point from line}}(\mathbf{x}) = \frac{1}{d(\mathbf{x}(t_0), l)^2}$$

$$E_{\text{point from plane}}(\mathbf{x}) = \frac{1}{d(\mathbf{x}(t_0), V)^2}$$

$$E_{\text{segment from point}}(\mathbf{x}) = \frac{1}{\int_{t_{\text{min}}}^{t_{\text{max}}} d(\mathbf{x}(t), \mathbf{p})^2 dt}$$

$$E_{\text{segment from line}}(\mathbf{x}) = \frac{1}{\int_{t_{\text{min}}}^{t_{\text{max}}} d(\mathbf{x}(t), l)^2 dt}$$

$$E_{\text{segment from plane}}(\mathbf{x}) = \frac{1}{\int_{t_{\text{min}}}^{t_{\text{max}}} d(\mathbf{x}(t), V)^2 dt}$$

with $\mathbf{p}$, $l$ and $V$ an arbitrary point, line and plane in $\mathbb{R}^3$.

### 3.4.4 The effect of scaling a curve and its parametrization

In section 3.3.4 the effect of scaling a curve or its parametrization on the internal energy was investigated. Now we will investigate the effect of these transformations on the external energy.

When the curve $\mathbf{x}$ is compared with the scaled curve $K\mathbf{x}$, where of course the geometric entities of the operators are also scaled with the factor $K$, the following relations are found:

$$E_{\text{positional attractor}}(K\mathbf{x}) = K^2 E_{\text{positional attractor}}(\mathbf{x})$$

$$E_{\text{directional attractor}}(K\mathbf{x}) = K^2 E_{\text{directional attractor}}(\mathbf{x})$$

$$E_{\text{positional repeller}}(K\mathbf{x}) = (1/K^2) E_{\text{positional repeller}}(\mathbf{x})$$

$$E_{\text{directional repeller}}(K\mathbf{x}) = (1/K^2) E_{\text{directional repeller}}(\mathbf{x}),$$

where $E_{\text{positional attractor}}$ is the energy term of one of the positional attractors, $E_{\text{directional attractor}}$ is the energy term of one of the directional attractors, etc. Because of the completely different behavior of the attractor and repeller weights, it seems a good idea to adapt the weight factors after a scaling of the curve.

When the curve $\mathbf{x}$ is compared with the curve $\mathbf{x} \circ \phi$, with $\phi(t) = \gamma t + \delta$ a linear function, the following relations are found:

$$E_{\text{positional attractor}}(\mathbf{x} \circ \phi) = (1/\gamma) E_{\text{positional attractor}}(\mathbf{x})$$

$$E_{\text{directional attractor}}(\mathbf{x} \circ \phi) = \gamma E_{\text{directional attractor}}(\mathbf{x}),$$
3.4 EXTERNAL ENERGY

\[ E_{\text{positional\_repellor}}(\mathbf{x} \circ \phi) = \frac{1}{\gamma} E_{\text{positional\_repellor}}(\mathbf{x}), \]
\[ E_{\text{directional\_repellor}}(\mathbf{x} \circ \phi) = \frac{1}{\gamma^3} E_{\text{directional\_repellor}}(\mathbf{x}). \]

To cancel the effect of scaling the parametrization, in the implementation described in section 3.6 the parameter domain \([a, b]\) is consequently transformed to \([0, 1]\). According to the above relations this is equivalent to multiplying the weights of the four types of operators with \((b - a), 1/(b - a), (b - a),\) and \((b - a)^3\) respectively.
3.5 Curve representations

The choice of the curve representation is very important in variational modeling. It has a large influence on the solvability of the energy minimization problem. In this section an overview is given of two curve representations that are often used in CAGD: Bézier curves and B-spline curves. Their suitability for variational modeling is investigated using the list of desirable properties for curve representations given in section 3.1.7. An excellent reference for curve (and surface) representations is [Far93].

3.5.1 Bézier curves

A Bézier curve is a polynomial curve. It is defined in terms of the univariate Bernstein polynomials

\[ B^n_i(t) = \binom{n}{i} t^i(1-t)^{n-i}, \quad t \in [0, 1], \]

(3.24)

where the binomial coefficients are given by

\[ \binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leq i \leq n \\ 0 & \text{otherwise.} \end{cases} \]

A degree \( n \) Bézier curve \( \mathbf{x} \) is defined as

\[ \mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{P}_i B^n_i(t), \quad t \in [0, 1], \]

(3.25)

where \( \{\mathbf{P}_i\} \) is a set of control points in \( \mathbb{R}^3 \). Note that this curve can be rewritten into the canonical curve representation (3.7), that was used in previous sections. A Bézier curve can represent any polynomial curve on the interval \([0, 1]\). A Bézier curve has the following properties (most of them are explained in section 3.1.7)

- **Affine Invariance** This is a consequence of the fact that the Bernstein polynomials are positive and sum to one.

- **Coefficient-Based Algorithms** The de Casteljau algorithm is a coefficient-based algorithm that can be used for the computation of points and derivatives of a Bézier curve (see [Far93]).

- **Refinement** There are two common ways to refine a (composite) Bézier curve. With **subdivision** a Bézier curve on the interval \([0, 1]\) is split into two successive Bézier curves defined on the intervals \([0, c]\) and \([c, 1]\), \(0 < c < 1\). With **degree elevation**, the degree of the Bézier curve is raised. Formulas for the new control points of the Bézier curve after subdivision or degree elevation can be found in [Far93].
3.5 Curve Representations

- **Endpoint interpolation** A Bézier curve interpolates the first and last control point, i.e. \( x(0) = P_0 \) and \( x(1) = P_n \).

One single Bézier curve is not very suitable for curve modeling. Although polynomial curves can approximate smooth curves with arbitrary precision, the degree of these approximations quickly becomes unacceptably high. For this reason virtually always composite Bézier curves are used.

**Composite Bézier curves**

A *composite Bézier curve* is a chain of Bézier curves. It is formed by joining the end-point of a Bézier curve with the begin-point of the next one. Because of the end-point interpolation property of Bézier curves, this means that two successive segments have to share a control point. A composite Bézier curve on the \( m \) segments \([t_k, t_{k+1}]\), \( k = 0, \ldots, m-1 \) can be defined as follows:

\[
x(t) = \sum_{i=0}^{n} P_{i+k} B^m_i(\lambda(t)), \quad \text{such that } t \in [t_k, t_{k+1}],
\]

where \( \lambda(t) = \frac{t - t_k}{t_{k+1} - t_k} \) is the local parameter on the segment \([t_k, t_{k+1}]\). The curve is defined by \( mn + 1 \) control points \( P_i \). A composite Bézier curve can be written in the canonical curve representation (3.7). The basis functions \( N_i \) in this representation have the form \( B^m_i \circ \lambda_k \), except for the basis functions corresponding to a shared control point. These basis functions are composed of two functions \( B^m_i \circ \lambda_k \). Besides the properties already mentioned for a single Bézier curve, a composite Bézier curve also has local support and a restricted form of automatic continuity:

- **Local support** When a composite Bézier curve is written in the canonical representation (3.7), the basis functions \( N_i \) are nonzero on only one segment, or on two successive segments if \( i \) corresponds to a shared control point \( P_i \).

- **Automatic Continuity** Composite Bézier curves only have automatic \( C^0 \) continuity. This means that continuity constraints have to be applied to enforce continuity of a higher order between successive segments.

We can conclude that composite Bézier curves have a number of properties that are attractive for variational modeling. They have a simple (control point) representation, there is an efficient evaluation algorithm, and they can be easily refined. However, the lack of automatic \( C^1 \) or \( C^2 \) continuity can be a problem. If the number of segments is large, the number of continuity constraints is also large. This has a negative effect on the minimization problem.
3.5.2 B-spline curves

A B-spline curve is a piecewise polynomial curve. A **B-spline curve** of degree \(d\) with **knot sequence** \(\{t_0, \ldots, t_{m+2d-2}\}\) and control points \(\{P_0, \ldots, P_{m+d-1}\}\) is defined as (see [Far93]):

\[
x(t) = \sum_{i=0}^{m+d-1} P_i N_i^d(t),
\]

(3.26)

where the **B-spline basis functions** \(N_i^d\) are piecewise polynomials functions, defined by the de Boor recursion relations:

\[
N_i^0(u) = \begin{cases} 1 & \text{if } t_{i-1} \leq t < t_i, \\ 0 & \text{otherwise.} \end{cases}
\]

\[
N_i^d(u) = \frac{t - t_{i-1}}{t_{i+d} - t_{i-1}} N_i^{d-1}(t) + \frac{t_{i+d} - t}{t_{i+d} - t_i} N_{i+1}^{d-1}(t).
\]

(3.27)

(the indeterminate expression \(0/0\) is treated as 0).

The curve is defined over the domain \([t_{d-1}, \ldots, t_{d+m-1}]\). It can be divided into \(m\) polynomial **segments**, defined on the intervals \([t_{i-1}, t_i]\), \(d \leq i < d + m\). A B-spline curve is called **uniform** if the spacing between the knots is constant \((t_i - t_{i-1} = h\) for all \(i\)). **Basis functions** of uniform B-spline curves have the following property:

\[
N_i^d(t) = N_{i-k}^d(t - kh), \quad k \in \mathbb{N}, \quad k \leq i
\]

(3.28)

Some important properties of B-spline curves are (see also section 3.1.7)

- **Local support** The basis function \(N_i^d\) is nonzero on the interval \([t_{i-1}, t_{i+d}]\).
- **Affine Invariance** This is a consequence of the fact that the B-spline basis functions are positive and sum to one.
- **Coefficient-Based Algorithms** The **de Boor algorithm** is a coefficient-based algorithm that can be used for the computation of points and derivatives of a B-spline curve (see [Far93]).
- **Refinement** A B-spline curve can be refined by **knot insertion** (see [Far93]).
- **Automatic Continuity** A B-spline curve is \(C^{d-1}\) continuous if the knots \(t_i\) are **simple** (do not coincide). It is possible to lower the order of continuity at a joint \(t_i\) of two segments by letting some of the knots coincide \((t_i = t_{i+1} = \ldots)\). There is one small problem related to the continuity of B-spline curves: the fact that they are \(C^{d-1}\) continuous doesn’t mean that they are always regular! Consider the uniform B-spline curve \(x\) with control points \((0,0,0), (1,0,0), (1,1,0)\) and \((0,5,0)\), and knot vector \(\{0,1,2,3,4,5\}\) (see figure 3.14). This curve has an irregular point (with zero derivative) at the parameter 2.5: \(x(2.5) = 0\). Although the curve is \(C^2\) continuous, it isn’t \(C^1\) continuous in the point with parameter 2.5! In this point the tangent vector changes direction, which causes a sharp peak (sometimes called: **cusp**).
3.5 CURVE REPRESENTATIONS

Figure 3.14: Example of a uniform B-spline curve with a cusp. The sharp peak is caused by an irregular point, where the derivative of the curve is zero and changes from direction.

B-spline curves are very well suited for variational modeling. Their (control point) representation is simple, there exists an efficient evaluation algorithm and they can be easily refined. Compared to composite Bézier curves they have the advantage of automatic $C^{d-1}$ continuity, which cancels the need for continuity constraints.
3.6 Implementation and results

A program has been built for the modeling of variational curves with the external operators introduced in this dissertation. The goal of this program is to form a test environment for the modeling operators described in the previous sections. First a short description of the functionality of the program will be given. Then it will be explained how the constraint equations and the internal and external energies can be computed and how the minimization problem can be solved. Finally, some examples of modeling with the program will be given.

3.6.1 Functionality of the program

A typical modeling session with the program would be as follows: the design starts with a given curve (for instance a straight line). This curve is modified by applying new constraints and operators to the curve or by editing existing ones. When this is done, the energy of the curve is minimized. The last two steps are repeated until the user is satisfied with the shape of the curve. The energy minimization isn’t performed in real time. This is perhaps feasible for curves with a small number of degrees of freedom, but in normal situations the computation of the minimal energy curve takes too much time (varying from a few seconds to several minutes in extreme cases).

The internal energy of the curve is a linear combination of the bend and stretch energy, as described in section 3.3. The simple approximations of the bend and stretch energy (\(E_{\text{bend\_simple}}\) and \(E_{\text{stretch\_simple}}\)) as well as the data dependent approximations (\(E_{\text{bend\_data}}\) and \(E_{\text{stretch\_data}}\)) have been implemented. As a reference curve always the current curve is used. For the external energy, the operators described in section 3.4 can be chosen. The constraints are limited to the linear constraints described in section 3.2.

The curve can be viewed as a two-dimensional view of the curve displayed in a window. For this, a parallel projection is used (see section 2.6.3). The direction of projection is always chosen perpendicular to the projection plane. Using the mouse, the user can freely rotate, translate and zoom in or out on the scene. Figure 3.15 shows a screen dump of the program.

The constraints and the external energy operators applied to the curve are displayed graphically. For example, the operator on top of the curve in figure (3.15) represents a point to plane attractor. To prevent that the screen becomes too cluttered, the user may decide which of the following items are visible: the curve, the control points of the curve, the constraints, the external energy operators, the axes of the coordinate system and some auxiliary lines that connect the constraints and operators with the curve.

The program supports the following user interaction:

- The number of control points of the curve can be chosen; control points can also be added or deleted.
- The internal energy of the curve can be set to the simple approximation \(\alpha E_{\text{stretch\_simple}} + (1-\alpha) E_{\text{bend\_simple}}\) or the data dependent approximation \(\alpha E_{\text{stretch\_data}} + (1-\alpha) E_{\text{bend\_data}}\). The weight factor \(\alpha\) can be freely chosen.
3.6 IMPLEMENTATION AND RESULTS

Figure 3.15: A screen dump of an interactive program for variational modeling with B-spline curves. The user can graphically apply and edit constraints and operators to the curve.

- The previously mentioned constraints and external energy operators can be entered graphically.
- The control points, constraints and external energy operators can be edited by direct manipulation using the mouse. Their parameters can also be entered numerically. This is sometimes necessary, for instance when two operators are applied to exactly the same point of the curve.
- The minimal energy curve for the current selection of operators and internal energy is computed when requested.

In a real-world application, the details of the curve representation should be hidden completely from the user. Since the main purpose of the program is to test the modeling operators, this goal has only been partially met. For instance, the control points are not made invisible to the user.

The domain of a B-spline curve depends on the number of control points. To hide this aspect to the user, the parameter interval is consequently transformed to [0,1]. In an ideal situation however, the whole concept of parametrization should be hidden to the user.


CHAPTER 3 CURVES

The number of control points is currently under control of the user. It should be possible to shift this control from the user to the program. For this a probably heuristic procedure is needed to decide where control points need to be added or may be deleted to produce an accurate approximation of the minimal energy curve. In this context, ‘accurate’ means that adding more control points to the curve would yield a minimal energy curve which is close to the old one.

For the graphical input of operators it is necessary to select points on the curve by pointing and clicking with the mouse. A point on the screen corresponds to a three-dimensional line parallel to the direction of projection. A procedure is available that computes the point on the curve with minimal distance to this line.

It is possible for a user to apply a set of operators and constraints for which no solution or more than one solution exists. Currently there is only limited support to prevent this.

The factor $\alpha$ in the internal energy and the weight factors from the external energy operators are sensitive to changes in the length of the parameter domain of the curve. Therefore, if the number of control points is changed, these weight factors should be corrected. The corrections proposed in the sections 3.3.4 and 3.4.4 are used to deal with this problem.

3.6.2 Uniform cubic B-spline curves

For the curve representation scheme uniform cubic B-spline curves (3.26) were chosen with the following representation:

$$\mathbf{x}(t) = \sum_{i=0}^{n-1} \mathbf{P}_i N_i^3(t),$$

where $N_i^3$ is the B-spline basis function defined in (3.27), with knots $t_i = i, i = 0 \ldots n + 1$. The domain of the curve is the interval $[2, n - 1]$. In section 3.5 we explained why B-spline curves are suitable for variational modeling. Cubic curves were chosen for the reason that they are $C^2$ continuous. In practice a higher order degree of continuity is seldom necessary. Furthermore, higher order B-splines usually have significantly more control points, which complicates the energy minimization.

A nonuniform spacing of the knots may help to obtain a parametrization of the curve that is more consistent with the arc length parametrization. Although this is certainly advantageous for the accuracy of the approximations to the bend and stretch energy (see also section 3.3), it is not used in the modeling program. The main reason is that assumptions to the final shape of the curve would have to be made, and this shape is not known beforehand. Furthermore, the computation of the minimal energy curve can be done much more efficiently in the case of uniform B-spline curves.

There are two minor problems related to the continuity of uniform B-spline curves:

- Uniform B-spline curves do not support corners (i.e. points where the curve is $C^0$, but not $C^1$). Normally such a corner is enforced by letting some of the knots coincide, but this is currently not supported in the implementation. As a consequence, the modeling of corners can only be done by joining two separate minimal energy curves.
3.6 IMPLEMENTATION AND RESULTS

- B-spline curves are not always regular (see section 3.5). As a consequence of this, it may sometimes happen that a minimal energy curve has cusps.

The shape of a uniform B-spline curve is completely specified by the concatenation vector $\mathbf{P}$ of the control points $\mathbf{P}_i$ (see also section 3.1.7). The constraints and the internal and external energies will now be expressed in this vector $\mathbf{P}$.

3.6.3 Computing the constraints

In this section we explain how the constraint equations can be computed and how they are expressed in terms of the concatenation vector $\mathbf{P} \in \mathbb{R}^{3n}$, with $n$ the number of control points.

From the constraint representations given in section 3.2 it follows that the point to point/line/plane constraint and the tangent to vector/plane constraint can all be written as one or more equations of the form

$$\mathbf{d}_k^T \mathbf{P} = \epsilon_k,$$

for some vector $\mathbf{d}_k \in \mathbb{R}^{3n}$ and some constant $\epsilon_k \in \mathbb{R}$.

A more detailed examination of the constraint representations reveals that the components of $\mathbf{d}_k$ and $\epsilon_k$ can be expressed in the values $N_i^3(t_0)$, $\hat{N}_i^3(t_0)$ and $\hat{\hat{N}}_i^3(t_0)$, $0 \leq i < n$. The value $t_0$ is the parameter value of the point on the curve to which the constraint is applied. The constraints can therefore be combined into one matrix equation

$$D \mathbf{P} = \mathbf{e},$$

with $D \in \mathbb{R}^{m \times 3n}$, for some $m$. Due to the local support of B-splines, each constraint equation depends on a small number of control points. Consequently, the matrix $D$ is sparse.

3.6.4 Computing the internal energy

This section discusses how the internal energy of a uniform cubic B-spline curve can be computed. Special attention will be given to the structure of the matrix $A$ that represents this internal energy. Since this matrix may be very large, this structure should be exploited both for reasons of storage capacity as for the efficient minimization of the energy.

In section 3.3 we showed that the simple approximations $E_{\text{bend\_simple}}$, $E_{\text{stretch\_simple}}$ and the data dependent approximations $E_{\text{bend\_data}}$, $E_{\text{stretch\_data}}$ can all be written as a quadratic expression in the control points:

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j.$$  

The matrices $A = [a_{ij}]$ corresponding to the coefficients $a_{ij}$ are denoted as $A_{\text{bend\_simple}}$, $A_{\text{bend\_data}}$, $A_{\text{stretch\_simple}}$ and $A_{\text{stretch\_data}}$ respectively.
CHAPTER 3 CURVES

Simple approximations

Let \( \mathbf{x}_k \) be the \( k \)-th segment of the curve \( \mathbf{x} \), defined on the interval \([2 + k, 3 + k]\). Due to the local support of B-spline curves, the segment \( \mathbf{x}_k \) can be written as

\[
\mathbf{x}_k(t) = \sum_{i=0}^{3} \mathbf{P}_{i+k} \tilde{N}_{i+k}^3(t).
\]

The simple approximations of the bend and stretch energy of this segment are given by

\[
\begin{align*}
E_{\text{bend simple}}(\mathbf{x}_k) &= \sum_{i=0}^{3} \sum_{j=0}^{3} b_{ij} \mathbf{P}_{i+k} \cdot \mathbf{P}_{j+k}, \\
E_{\text{stretch simple}}(\mathbf{x}_k) &= \sum_{i=0}^{3} \sum_{j=0}^{3} s_{ij} \mathbf{P}_{i+k} \cdot \mathbf{P}_{j+k}
\end{align*}
\]

with \( b_{ij} = \int_{2+k}^{3+k} \tilde{N}_{i+k}^3 \tilde{N}_{j+k}^3 dt \) and \( s_{ij} = \int_{2+k}^{3+k} \tilde{N}_{i+k}^3 \tilde{N}_{j+k}^3 dt \). Due to property (3.28) of uniform B-splines, the matrices \( B = [b_{ij}] \) and \( S = [s_{ij}] \) do not depend on \( k \). They are given by

\[
B = \frac{1}{6} \begin{pmatrix}
2 & -3 & 0 & 1 \\
-3 & 6 & -3 & 0 \\
0 & -3 & 6 & -3 \\
1 & 0 & -3 & 2
\end{pmatrix}, \quad S = \frac{1}{120} \begin{pmatrix}
6 & 7 & -12 & -1 \\
7 & 34 & -29 & -12 \\
-12 & -29 & 34 & 7 \\
-1 & -12 & 7 & 6
\end{pmatrix}.
\]

The matrix \( B \) has eigenvalues 0, 0, 1 and 1.66667 and the matrix \( S \) has eigenvalues 0, 0.0096846, 0.083333 and 0.57365. The null space of the matrix \( B \) (i.e. the eigenvectors with eigenvalue 0) is \( \{ (\lambda, \lambda + \mu, \lambda + 2\mu, \lambda + 3\mu)^T | \lambda, \mu \in \mathbb{R} \} \) and the null space of the matrix \( S \) is \( \{ (\lambda, \lambda, \lambda)^T | \lambda \in \mathbb{R} \} \). From this it can be derived that the bend energy \( E_{\text{bend simple}}(\mathbf{x}_k) \) is zero if the control points \( \mathbf{P}_k \ldots \mathbf{P}_{3+k} \) are on a straight line and have the same distances \( (\mathbf{P}_{j+2} - \mathbf{P}_{j+1} = \mathbf{P}_{j+1} - \mathbf{P}_j, j = k, k+1) \) and the stretch energy \( E_{\text{stretch simple}}(\mathbf{x}_k) \) is zero if the control points \( \mathbf{P}_k \ldots \mathbf{P}_{3+k} \) coincide.

The bend energy of a curve is the sum of the bend energies from its segments. The matrix \( A_{\text{bend simple}} \) can therefore be obtained by starting with the \( n \times n \) zero matrix and then ‘adding’ the matrix \( B \) at the appropriate place for each segment of the curve (see figure 3.16). The matrix \( A_{\text{stretch simple}} \) can be constructed from \( S \) in the same way. The matrices \( A_{\text{bend simple}} \) and \( A_{\text{stretch simple}} \) have the following properties (see also section 3.3)

- They are symmetric.
- They are positive semidefinite.
- They are banded. The upper and lower bandwidth of \( A_{\text{bend simple}} \) and \( A_{\text{stretch simple}} \) are 3.
- \( A_{\text{bend simple}} \) has eigenvalue 0 with multiplicity 2 and \( A_{\text{stretch simple}} \) has eigenvalue 0 with multiplicity 1.
3.6 IMPLEMENTATION AND RESULTS

Figure 3.16: The structure of the bend matrix $A_{\text{bend\_simple}}$ is graphically represented in the above picture. The matrix consists of $m$ copies of the matrix $B$, where $m$ is the number of segments of the curve.

The bend energy $E_{\text{bend\_simple}}(\mathbf{x})$ is zero if the control points $\{\mathbf{P}_i\}$ satisfy the relations $(\mathbf{P}_{j+2} - \mathbf{P}_{j+1} = \mathbf{P}_{j+1} - \mathbf{P}_j, j = 0 \ldots n - 3)$ and the stretch energy $E_{\text{stretch\_simple}}(\mathbf{x})$ is zero if the control points $\{\mathbf{P}_i\}$ all coincide.

Data dependent approximations

The data dependent approximations $E_{\text{bend\_data}}(\mathbf{x}_k)$ and $E_{\text{stretch\_data}}(\mathbf{x}_k)$ to the bend and stretch energy of the $k$-th segment $\mathbf{x}_k$ also have the representation (3.30), with coefficients $b_{ij} = \int_{j2+k}^{j3+k} \text{Hess}_c(N_i) \text{Hess}_c(N_j) \|\mathbf{\dot{c}}\| dt$ and $s_{ij} = \int_{j2+k}^{j3+k} \text{grad}_c(N_i) \cdot \text{grad}_c(N_j) \|\mathbf{\dot{c}}\| dt$. In this case however, the coefficients $b_{ij}$ and $s_{ij}$ do depend on $k$, so they have to be computed for all segments. Furthermore, they cannot be computed analytically, so they have to be approximated using the numerical integration routines from section 2.5.5. The matrices $A_{\text{bend\_data}}$ and $A_{\text{stretch\_data}}$ have the same properties as the matrices $A_{\text{bend\_simple}}$ and $A_{\text{stretch\_simple}}$ given above.

Internal energy expressed in the concatenation vector

The internal energy of a curve $\mathbf{x}$ is a linear combination of the bend and stretch energy: $E_{\text{int}}(\mathbf{x}) = \alpha E_{\text{bend}}(\mathbf{x}) + (1 - \alpha)E_{\text{stretch}}(\mathbf{x})$ and can therefore be represented as

$$E_{\text{int}}(\mathbf{x}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j,$$

where the matrix $A_{\text{int}} = [a_{ij}]$ is given by $A_{\text{int}} = \alpha A_{\text{bend\_simple}} + (1 - \alpha)A_{\text{stretch\_simple}}$ or $A_{\text{int}} = \alpha A_{\text{bend\_data}} + (1 - \alpha)A_{\text{stretch\_data}}$ depending on which approximation is used. If we express the internal energy in the concatenation vector $\mathbf{P}$ of the control points, we obtain

$$E_{\text{int}}(\mathbf{x}) = \mathbf{P}^T \mathbf{A}_{\text{int}} \mathbf{P}, \quad (3.31)$$
with $\mathbf{A}_{\text{int}}$ a $(3n \times 3n)$ matrix, that is obtained from $A_{\text{int}}$ by replacing each component $a_{ij}$ with the $3 \times 3$ matrix

$$
\begin{pmatrix}
a_{ij} & 0 & 0 \\
0 & a_{ij} & 0 \\
0 & 0 & a_{ij}
\end{pmatrix}
$$

The matrix $\mathbf{A}_{\text{int}}$ inherits the following pleasing properties from the matrices $A_{\text{end}}$ and $A_{\text{stretch}}$: $\mathbf{A}_{\text{int}}$ is symmetric, positive semidefinite and banded (with upper and lower bandwidth 9). It is completely specified by the number of control points $n$, the matrices $B$ and $S$, and the factor $\alpha$.

Note that the internal energy can be separated into an $x$-, $y$- and a $z$-part. Let $\mathbf{P}_x = (x_0, \ldots, x_{n-1})^T$, $\mathbf{P}_y = (y_0, \ldots, y_{n-1})^T$ and $\mathbf{P}_z = (z_0, \ldots, z_{n-1})^T$. Then $E_{\text{int}}(\mathbf{x}) = \mathbf{P}_x^T A_{\text{int}} \mathbf{P}_x + \mathbf{P}_y^T A_{\text{int}} \mathbf{P}_y + \mathbf{P}_z^T A_{\text{int}} \mathbf{P}_z$. It is also worth noting that the bandedness of $A$ would have been severely destroyed if the concatenation vector $\mathbf{P}$ was defined as $\left( \mathbf{P}_x^T, \mathbf{P}_y^T, \mathbf{P}_z^T \right)^T$.

### 3.6.5 Computing the external energies

The external energies from the previous section can also be expressed in the concatenation vector $\mathbf{P}$ (3.8). From the external energy representations given in section 3.4 it follows that the point to point/line/plane attractors, the segment to point/line/plane/curve attractors and the tangent to line/plane attractors can all be written in the form

$$
E_{\text{attractor}}(\mathbf{x}) = \mathbf{P}^T A_{\text{attractor}} \mathbf{P} + B_{\text{attractor}}^T \mathbf{P} + C_{\text{attractor}},
$$

(3.32)

for some matrix $A_{\text{attractor}} \in \mathbb{R}^{3n \times 3n}$, some vector $B_{\text{attractor}} \in \mathbb{R}^{3n}$ and some constant $C_{\text{attractor}} \in \mathbb{R}$. The energy contributions of the corresponding repellors have the following form:

$$
E_{\text{repellor}}(\mathbf{x}) = \frac{1}{\mathbf{P}^T A_{\text{attractor}} \mathbf{P} + B_{\text{attractor}}^T \mathbf{P} + C_{\text{attractor}}}.
$$

(3.33)

For operators that apply to a single point with parameter $t_0$, the components of $A_{\text{attractor}}$, $B_{\text{attractor}}$ and $C_{\text{attractor}}$ can be expressed in the values $N_i^3(t_0), \tilde{N}_i^3(t_0)$ and $\tilde{N}_i^3(t_0), 0 \leq i \leq n$.

For operators that apply to an interval $[t_{\min}, t_{\max}]$, the components of $A_{\text{attractor}}$, $B_{\text{attractor}}$ and $C_{\text{attractor}}$ can be expressed in the following integrals

$$
\begin{align*}
&\int_{t_{\min}}^{t_{\max}} N_i^3(t) \, dt, \\
&\int_{t_{\min}}^{t_{\max}} N_i^3(t) \tilde{N}_j^3(t) \, dt, \\
&\int_{t_{\min}}^{t_{\max}} \tilde{N}_i(t) N_j^3(t) \, dt, \\
&\int_{t_{\min}}^{t_{\max}} \tilde{N}_i(t) \tilde{N}_j^3(t) \, dt, \\
&\int_{t_{\min}}^{t_{\max}} t^r N_i^3(t) \, dt, & r = 0 \ldots 3,
\end{align*}
$$

(3.34)
for $0 \leq i, j < n$.

Since the B-spline basis functions $N_i^3$ are piecewise polynomial, these functions have to be integrated for individual segments. However, due to the local support of B-splines and due to property (3.28) of uniform B-spline curves, it suffices to compute the corresponding primitive functions for only one segment (say the first segment $[2,3]$) and for $0 \leq i, j \leq 3$. For example, let $[v, w]$ be the part of the interval $[t_{\min}, t_{\max}]$ on the $k$-th segment of the curve, i.e. $[v, w] = [t_{\min}, t_{\max}] \cap [2 + k, 3 + k]$. Then we can write

$$
\int_v^w N_i^3(t) N_j^3(t) dt = \int_{v+2-k}^{w+2-k} N_{i-2+k}^3(t) N_{j-2+k}^3(t) dt.
$$

The domain of this integral is inside the interval $[2,3]$ and the integral is nonzero if $0 \leq i - 2 + k, j - 2 + k \leq 3$.

The primitive functions of the integrals in (3.31) have to be computed only once. Once they are stored, the computation of the integrals (3.34) is just a substitution of the right parameter values into these polynomials.

### 3.6.6 Solving the minimization problem

The energy minimization of a curve is a nonlinear programming problem (see section 2.5.3). In this section the problem will be formulated in a general form, and the solution methods that have been implemented will be discussed.

The constraints on a curve were expressed as a linear system in the concatenation vector $P$ of the control points in (3.29). The goal function of the minimization problem is the total energy, which is the sum of the internal energy $E_{\text{int}}(x)$ and the external energy $E_{\text{ext}}(x)$. These quantities were expressed in the concatenation vector $P$ in ((3.31), (3.32) and (3.33)). The total energy will be denoted as $f(P)$. Two different cases can be distinguished: if no repellors are applied, the total energy $f(P)$ is a quadratic function in the vector $P$, and if repellors are applied, $f(P)$ contains nonlinear terms that are more difficult to minimize.

**Sparseness**

If the number of the control points $n$ is large, the $(m \times 3n)$ matrix corresponding to the constraints (3.29) and the $(3n \times 3n)$ matrices corresponding to the internal and external energies in (3.31), (3.32) and (3.33) become too large to store them as two dimensional arrays in internal memory. In this case, they have to be stored using sparse representations. In the implementation, for the constraint matrix $D$ in (3.29) only the nonzero elements are stored. The internal energy matrix $\bar{A}_{\text{inf}}$ in (3.31) is stored as a bandmatrix with bandwidth 9. For the external energy matrices $A_{\text{operator}}$ in (3.32) and (3.33) only the nonzero elements are stored. Note that due to the local support of B-splines, the matrices $A_{\text{operator}}$ have a limited bandwidth too.
Minimization without repellers

If the external energy of the curve contains no repellers (and hence terms of the form (3.33) do not appear), the total energy \( f(P) \) of the curve is quadratic in \( P \). Therefore, the minimization problem can then be formulated as a quadratic programming problem with linear constraints:

\[
\begin{align*}
\text{minimize } & \quad f(P) = P^T A P + b^T P + c \\
\text{such that } & \quad DP = e
\end{align*}
\]

(3.35)

for some \( A \in \mathbb{R}^{3n \times 3n}, \ b \in \mathbb{R}^{3n}, \ c \in \mathbb{R}, \ D \in \mathbb{R}^{m \times 3n} \) and \( e \in \mathbb{R}^m \).

The matrix \( A \), which is a weighted sum of the internal energy matrix \( \overline{A}_{int} \), and the external energy matrices \( A_{oper, dir} \) given above, is stored as a bandmatrix. For the matrix \( D \) only the nonzero elements are stored. The system (3.35) is solved using either of the two methods described in section 2.5.4 (the null space method and the Lagrangian method).

The null space method consists of two steps. The first step is the elimination of the constraints \( DP = e \) by rewriting them into the form \( P = FY + g \). This is implemented using Gaussian elimination. However, this procedure doesn’t exploit the sparseness of the matrix \( D \). As a result, this method is only efficient if the number of constraints \( m \) is small. The second step is to solve the linear system that remains after the elimination of the constraints (2.38). This system is solved using LU-decomposition (if the number of control points \( n \) is small) or using conjugate gradients (if \( n \) is large).

The Lagrangian method requires only one step: solving the linear system (2.39). This is done using stabilized orthogonal directions (STOD) and the conjugate residual method (CR), see also section 2.5.1. There was no preconditioner applied.

Minimization with repellers

If the external energy of the curve contains repellers (and hence contains terms that are reciprocals of quadratic functions) the total energy \( f(P) \) of the curve is not quadratic in \( P \). Therefore, the minimization problem can then be formulated as a nonlinear programming problem with linear constraints:

\[
\begin{align*}
\text{minimize } & \quad f(P) \\
\text{such that } & \quad DP = e
\end{align*}
\]

(3.36)

for some \( D \in \mathbb{R}^{m \times 3n} \) and \( e \in \mathbb{R}^m \). This problem is solved using the method described in section 2.5.3. First the constraints are eliminated by rewriting them into the form \( P = FY + g \). The remaining unconstrained minimization problem is of the form (2.35) and is solved using the nonlinear conjugate gradient method (see section 2.5.2).

3.6.7 Examples

A number of experiments with variational curve modeling have been done. A few of them will be described below.
3.6 IMPLEMENTATION AND RESULTS

Iterative application of data dependent fairness functionals

In section 2.4.7 it was suggested that the data dependent fairness functionals can be applied iteratively to compute a minimum of the exact functional. Given an initial curve $x_0$, a new curve $x_1$ is computed by minimizing a data dependent energy functional with $x_0$ as a reference curve. This curve $x_1$ is used as a reference curve for the computation of $x_2$ etc. This yields a sequence of curves $\{x_t\}_{t=0,1,...}$, which may possibly converge to a minimum of the exact functional.

First a uniform B-spline curve $x$ has been iteratively minimized with the data dependent stretch energy functional (3.19) as internal energy and the curve itself as the reference curve ($c = x$). There were three point to point constraints applied to the curve (the case with two point to point constraints is not interesting, since this always yields a straight line in one iteration). The exact solution to this problem (with minimal length) consists of two straight line segments. Figure 3.17 shows the initial curve (left) and the curve after 20 iterations (right). From this picture and the numerical values in table 3.3 it follows that the iterative method converges towards the exact solution.

![Figure 3.17: Left: the initial curve. Right: the curve after 20 iterations with the data dependent stretch energy functional.](image)

For the second experiment a uniform B-spline curve $x$ has been iteratively minimized with the data dependent bend energy functional (3.16) as internal energy and the curve itself as the reference curve ($c = x$). In this case the position and the tangents at the end-points were prescribed. Figure 3.18 shows the initial curve (left) and the curve after 2 iterations (right). The numerical values in table 3.3 show that in this case the iterative method does not converge. In the first few iterations the bend energy decreases, but at a certain moment the value of the bend energy starts increasing again. This behavior seems to be typical for the iterative minimization with the data dependent bend energy functional.

Both experiments are in agreement with the theory presented in section 2.4.7. At an exact minimum $\mathbf{x}$ of the stretch energy the data dependent stretch energy also attains a minimum, which indicates that the iterative method for the stretch energy is stable in the neighborhood of $\mathbf{x}$. For the bend energy this doesn’t hold, which means that the iterative method for the bend energy is unstable in the neighborhood of an exact minimum.
CHAPTER 3 CURVES

<table>
<thead>
<tr>
<th>iteration</th>
<th>stretch energy</th>
<th>iteration</th>
<th>bend energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>119.819</td>
<td>0</td>
<td>0.0514324</td>
</tr>
<tr>
<td>1</td>
<td>117.926</td>
<td>1</td>
<td>0.0269814</td>
</tr>
<tr>
<td>2</td>
<td>117.683</td>
<td>2</td>
<td>0.0269347</td>
</tr>
<tr>
<td>3</td>
<td>117.515</td>
<td>3</td>
<td>0.0269376</td>
</tr>
<tr>
<td>4</td>
<td>117.398</td>
<td>4</td>
<td>0.0269439</td>
</tr>
<tr>
<td>5</td>
<td>117.316</td>
<td>5</td>
<td>0.0269546</td>
</tr>
<tr>
<td>6</td>
<td>117.259</td>
<td>6</td>
<td>0.0269714</td>
</tr>
<tr>
<td>7</td>
<td>117.219</td>
<td>7</td>
<td>0.0269965</td>
</tr>
<tr>
<td>8</td>
<td>117.192</td>
<td>8</td>
<td>0.0270329</td>
</tr>
<tr>
<td>9</td>
<td>117.172</td>
<td>9</td>
<td>0.0270835</td>
</tr>
<tr>
<td>10</td>
<td>117.159</td>
<td>10</td>
<td>0.0271488</td>
</tr>
<tr>
<td>15</td>
<td>117.133</td>
<td>11</td>
<td>0.0272276</td>
</tr>
<tr>
<td>20</td>
<td>117.128</td>
<td>12</td>
<td>0.0273204</td>
</tr>
<tr>
<td>30</td>
<td>117.126</td>
<td>13</td>
<td>0.0274365</td>
</tr>
<tr>
<td>40</td>
<td>117.125</td>
<td>14</td>
<td>0.0276004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0.0278503</td>
</tr>
</tbody>
</table>

Table 3.3: The results of two experiments with the iterative application of the data dependent approximations for the stretch and bend energy.

Data dependent approximations versus simple approximations

Although the data dependent bend energy functional is not stable in the neighborhood of an exact minimum of the bend energy (see the previous experiment), it appears to be a valuable tool in practice. If only one or two iterations are carried out with the data dependent approximation of the bend energy, the bend energy (3.14) of the curve is usually much lower than when the simple approximation is used. When a designer starts modeling, and there is no reference curve available, the following scenario can be followed. First the energy of the curve is minimized using the simple approximation. Then the result of this minimization is used as a reference curve for the data dependent functional. This approach has been followed in a data fitting experiment.

First a number of random points in $\mathbb{R}^2$ is generated. Then a curve is generated that
interpolates these points and that minimizes the simple approximation of the bend energy. After this a curve is generated that interpolates the same points and that minimizes the data dependent approximation of the bend energy, with the former curve as the reference curve. The parameters for the interpolation points are set in two different ways: following a naive approach with equidistant parameters and following a more sophisticated approach where the parameters depend on the distances between successive points. This experiment has been repeated 10 times. The results of the experiments are given in table 3.4.

<table>
<thead>
<tr>
<th>experiment</th>
<th>equidistant parameters</th>
<th>scaled parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>simple appr.</td>
<td>data dep. appr.</td>
</tr>
<tr>
<td>1</td>
<td>21.2081</td>
<td>1.22031</td>
</tr>
<tr>
<td>2</td>
<td>2.00051</td>
<td>1.28658</td>
</tr>
<tr>
<td>3</td>
<td>67.5677</td>
<td>1593.58</td>
</tr>
<tr>
<td>4</td>
<td>3.99578</td>
<td>1.04302</td>
</tr>
<tr>
<td>5</td>
<td>4.28149</td>
<td>0.855798</td>
</tr>
<tr>
<td>6</td>
<td>28.9928</td>
<td>1.56551</td>
</tr>
<tr>
<td>7</td>
<td>1.20997</td>
<td>0.818409</td>
</tr>
<tr>
<td>8</td>
<td>3.7194</td>
<td>0.904392</td>
</tr>
<tr>
<td>9</td>
<td>43.8948</td>
<td>0.63318</td>
</tr>
<tr>
<td>10</td>
<td>63.2519</td>
<td>2.89351</td>
</tr>
</tbody>
</table>

Table 3.4: The results of a data fitting experiment. Each value in the table is the bend energy of a curve that was minimized subject to a number of random point interpolation constraints.

By comparing the first column of table 3.4 with the other three it follows that the sim-
ple bend energy produces extremely poor results when the parameters for the interpolation points are not chosen well. The numbers in the second column indicate that in this case the results can be dramatically improved by afterwards applying the data dependent approximation of the bend energy.

When the parameters are properly scaled, the simple bend energy yields a nearly optimal solution (column three). The results get (slightly) worse when the data dependent approximation of the bend energy is applied afterwards. This is caused by the previously mentioned instability of the data dependent bend energy approximation in the neighborhood of an exact solution. In the figures 3.19 and 3.20 the curves from the second experiment are depicted.

Figure 3.19: Two minimal energy curves that interpolate a number of randomly generated points. The left curve was minimized using the simple approximation of the bend energy, the right one using the data dependent approximation of the bend energy of the left curve. The parameters of the points were chosen equidistant.

Figure 3.20: Two minimal energy curves that interpolate the same points as in the previous figure. This time the parameters of the points were selected using the distances between the interpolation points.
3.6 IMPLEMENTATION AND RESULTS

It is clearly perceptible that the first curve in figure (3.19) (corresponding to the simple bend energy with equidistant parameters) has much sharper bendings than the other three, especially in the top right corner. From this experiment we may conclude that the data dependent energy functional is a useful tool in data fitting for cases where the choice of the parameters is not obvious.

Modeling example

In this section a small example of variational curve modeling is given. The first curve (figure 3.21, left) is obtained by applying three point to point constraints (at the end points and at the top) and by applying two tangent line constraints. The latter is a combination of a point to line constraint and a tangent to vector constraint with the same direction. This constraint ensures that the curve will be tangent to the line (drawn as a dotted line in the figure), without prescribing at which point.

The second curve (figure 3.21, right) is obtained from the first by applying a point to point attractor at two points of the curve. These attractors are depicted as arrows.

The third curve (figure 3.22, left) is obtained from the second by applying a point to point repellor at two points of the curve. The repellor point is depicted as a small circle, and the repellors are depicted as arrows.

Finally, the fourth curve (figure 3.22, right) is obtained from the third by increasing the weight of the point to point attractors. This demonstrates one of the most important advantages of modeling with external energy operators: once they are defined they give the user an easy way of deforming a minimal energy curve.

![Diagram of modeling example](image)

Figure 3.21: A small modeling experiment that demonstrates variational modeling with external energy operators. The second curve is obtained from the first by applying two point attractors.
Figure 3.22: The curve from the previous figure is further deformed by adding two point repellors (left) and by increasing the weight of the point attractors (right).
Chapter 4

Surfaces

4.1 Introduction

The second part of this dissertation is devoted to variational surface modeling. Naturally there is much resemblance with the first part about curves. These resemblances will only sparingly be mentioned. Instead, more attention will be paid to the places where differences occur. The structure of the chapters 3 and 4 is the same. It is therefore advised to always read the curve part of a topic first, before the corresponding surface part is read.

The modeling of smooth surfaces is very important in areas like industrial design and styling. The designer is faced with the following task: find a smooth surface that captures the ‘design intent’. The design intent can be roughly specified by imposing a number of geometric constraints (point interpolation, boundary interpolation, etc.) on the surface. The smoothness (or fairness) of a surface is measured according to some generally accepted criteria:

- wrinkles and undulations should be avoided
- flat spots should be avoided
- buckles or bumps should be avoided
- the surface should be ‘visually pleasing’

A problem is that some of these criteria are rather vague and cannot be defined exactly.

The shape of a surface usually depends on a number of parameters. These parameters will be called the shape parameters of the surface. For example, the shape parameters of a spline surface are the control points. To illustrate the difference between these shape parameters and the parametrization of a surface as defined in section 2.3, consider the linear surface $F(u, v) = uA + vB + C$, with $A$, $B$ and $C$ points in $\mathbb{R}^2$. The parametrization of the surface $F$ is determined by the parameters $u$ and $v$, and the shape of the surface is determined by the shape parameters $A$, $B$ and $C$.

Especially for surfaces, the number of shape parameters can be overwhelming. It is therefore not a good idea to put these shape parameters under direct control of the designer.
Tools are needed that automatically produce fair surfaces that satisfy a number of user specified geometric constraints. Variational modeling is such a tool. In variational modeling a certain fairness functional is defined that measures the fairness (or quality) of the surface. Then the surface is computed that minimizes this functional, while satisfying the constraints. The fairness functional is ideally based on physical energies or on geometric properties of the surface. Consequently, such a fairness functional shouldn’t depend on the parametrization of the surface. This is a desirable property of fairness functionals, since for a modeling task only the shape of the surface matters.

### 4.1.1 Thin plate energy

The most often used fairness functional is the thin plate energy. It is an expression that is proportional to the strain energy of a thin elastic plate. This fairness functional makes a surface resist bending, and for this reason we will call it the bend energy of a surface. For a parametric surface $\mathbf{F} : \Omega \rightarrow \mathbb{R}^3$ the bend energy is defined as

$$E_{\text{bend}}(\mathbf{F}) = \int_\Omega \kappa_1^2 + \kappa_2^2 \, d\omega_\mathbf{F},$$

where $d\omega_\mathbf{F} = \|\mathbf{F}_u \times \mathbf{F}_v\| \, dudv$ is the (infinitesimal) surface element and $\kappa_1$ and $\kappa_2$ are the principal curvatures. One can say that this functional measures the total curvature of $\mathbf{F}$. Since the expression (4.1) is far too complex for most applications, it is usually approximated. The following simplified version of the bend energy is often used (see [CG91] and [WW92])

$$E_{\text{bend},\text{simple}}(\mathbf{F}) = \int_\Omega \mathbf{F}_{uu} : \mathbf{F}_{uu} + 2\mathbf{F}_{uv} : \mathbf{F}_{uv} + \mathbf{F}_{vv} : \mathbf{F}_{vv} \, dudv.$$  

Since this expression is quadratic in the control points of a spline surface (see also section 4.3), it can be much easier minimized than the highly nonlinear functional $E_{\text{bend}}$. This is the reason for its frequent use in variational modeling, even in situations where $E_{\text{bend}}$ and $E_{\text{bend},\text{simple}}$ differ drastically.

### 4.1.2 Other fairness functionals

In the literature several other fairness functionals for surfaces are encountered. In this section we will mainly concentrate on the ones that are independent of the parametrization of the surface. These functionals are defined in terms of the geometric invariants like the principal curvatures $\kappa_1$ and $\kappa_2$, and the unit normal $\mathbf{N}$.

1. Analogous to curves (3.3), the bend energy can be combined with a stretching term (the surface area $\int d\omega_\mathbf{F}$)

$$E_{\text{thin,plate,under,tension}}(\mathbf{F}) = \int_\Omega \kappa_1^2 + \kappa_2^2 \, d\omega_\mathbf{F} + \sigma^2 \int \omega_\mathbf{F}.$$

This is the so called thin plate under tension model. See for instance [CG91], [WW92] and [Kal93], where quadratic approximations to this functional are used. If the tension
4.1 INTRODUCTION

parameter $\sigma^2$ is increased, the surface will resist stretching more, which generally yields surfaces with smaller surface area.

2. Rado and Roulier ([RRP91] and [RR91]) follow an elegant approach to obtain fairness functionals that have a flattening, a rounding or a rolling effect. From a surface $\mathbf{F} : \Omega \to \mathbb{R}^3$ another surface $\mathbf{c} : \Omega \to \mathbb{R}^3$ is derived that is merely defined by means of geometrical properties of $\mathbf{F}$, such as the Gaussian curvature $K = \kappa_1 \kappa_2$, the mean curvature $H = \frac{1}{2}(\kappa_1 + \kappa_2)$ and the unit normal $\mathbf{N}$. The surface area of this derived surface is used as a fairness functional. Their approach can be best illustrated with an example. Let us consider a fairness functional that has a rounding effect. Since a sphere is perfectly round, its fairness functional should be zero. This can be achieved by defining the derived surface $\mathbf{c}$ as the set of all centers of curvature of the surface $\mathbf{F}$. For a sphere this derived surface is a single point, so its fairness functional (the surface area of this point) is zero. Formally, this fairness functional is defined as follows:

- The rounding metric is the surface area of the derived surface

$$\mathbf{c}(u, v) = \mathbf{F}(u, v) + \frac{H(u, v)}{K(u, v)} \mathbf{N}(u, v).$$

This fairness functional has a tendency towards spherical shapes.

Four other fairness functionals, obtained by the same kind of reasoning, are given below.

- The flattening metric is the surface area of the derived surface

$$\mathbf{c}(u, v) = K(u, v) \mathbf{N}(u, v).$$

This metric has a tendency towards developable surfaces.

- The rolling metric is the surface area of the derived surface

$$\mathbf{c}(u, v) = \left( K(u, v) + H(u, v)^2 \right) \mathbf{N}(u, v).$$

This metric has a tendency towards cylindrical shapes or conical shapes.

- Another version of the rolling metric is the surface area of the derived surface

$$\mathbf{c}(u, v) = H(u, v) \mathbf{N}(u, v).$$

This metric has a tendency towards planar, cylindrical or conical shapes.

- Finally, a third version of the rolling metric is the surface area of the derived surface

$$\mathbf{c}(u, v) = \frac{1}{2(2H(u, v)^2 - K(u, v))} \mathbf{N}(u, v).$$

This metric has a tendency to roll the surface, but it will not flatten the surface.
3. Instead of the total curvature $\kappa_1^2 + \kappa_2^2$, one can also use the (absolute) Gaussian curvature $|K| = |\kappa_1 \kappa_2|$ or the squared mean curvature $4H^2 = (\kappa_1 + \kappa_2)^2$ in a fairness functional. The Gaussian curvature is not very suitable: a cylindrical shape has zero Gaussian curvature, but it may contain ripples or flat spots. Under normal circumstances, the squared mean curvature is also unsuitable, since it is sensitive to the relative magnitudes and signs of the principal curvatures (see [LP88]). However, if the boundary data (including the derivatives) are fixed, the squared mean curvature produces exactly the same results as the total curvature. In [Gre94] Greiner gives a quadratic data dependent approximation to the functional

$$\int_{\Omega} (\kappa_1 + \kappa_2)^2 \, d\omega_{\mathbf{F}}$$

and applies it successfully to surfaces with fixed boundaries.

4. Moreton and Sequin use fairness functionals in which the variation of the principal curvatures is taken into account (see [MS92] and [MS94])

$$\int_{\Omega} \frac{d\kappa_i}{d\mathbf{e}_j} \cdot \left( \frac{d\kappa_1}{d\mathbf{e}_1} + \frac{d\kappa_2}{d\mathbf{e}_2} \right) \, d\omega_{\mathbf{F}},$$

with $\frac{d\kappa_i}{d\mathbf{e}_j}$ the directional derivative of the $i$-th principal curvature with respect to the $j$-th principal direction. They have used the exact formulation of the functional, which yields very fair surfaces. However, the corresponding minimization problems are computationally expensive to solve.

5. Another possibility is to use a functional that measures the difference between a surface $\mathbf{F}$ and a given surface $\mathbf{F}_0$.

$$\int_{\Omega} (\mathbf{F} - \mathbf{F}_0)^2 \, d\omega_{\mathbf{F}}$$

Kallay uses this functional in combination with some of the previous ones in [Kal93]. The goal of this functional is to keep the distance between $\mathbf{F}$ and $\mathbf{F}_0$ small. This functional depends on the parametrization of $\mathbf{F}$ and $\mathbf{F}_0$. Ideally, this difference would be measured as a volume bounded by both surfaces, but this is far too difficult to do.

### 4.1.3 Modeling with minimal energy surfaces

Modeling with minimal energy surfaces goes exactly the same as for curves (see section 3.1.4). There are two basic ways of interaction in the design process of a minimal energy surface. In the first place, the designer can choose a suitable fairness functional. This gives a certain kind of global control over the shape of the surface. In the second place, the designer can apply geometric constraints to the surface (see section 4.2). A typical modeling session would be as follows. We assume that the designer starts with an initial surface. Then iteratively the following three steps are performed, until the designer is satisfied with the result:
4.1 INTRODUCTION

1. apply or edit constraints to the surface

2. minimize the energy

3. evaluate the result

Ideally the energy minimization would be performed in real time, but this is currently not feasible for most applications. The evaluation of a surface is an important aspect of surface modeling. Often a visual inspection, by rotating and zooming in or out on a 2D representation of the surface, is not sufficient. Interrogation tools are needed to help the designer with identifying unwanted curvature regions (see [BFHS6] and [HHS*92]).

Although the modeling with minimal energy surfaces is very powerful, there are also some problems related to it. The parametrization independent fairness functionals are too complicated to use for interactive modeling, and the interaction using constraints is not always flexible enough. These are serious problems for curves (see section 3.1.4), but they are even more serious for surfaces! However, the solutions given for curves (data dependent fairness functionals and external energy operators) are also applicable for surfaces.

4.1.4 Data dependent fairness functionals

For interactive modeling, the parametrization independent fairness functionals mentioned above are far too difficult. On the other hand, their (quadratic) approximations are simple, but they can be very poor (see also section 4.3).

Analogous to curves (section 3.1.5), Greiner has introduced the concept of data dependent fairness functionals for surfaces in [Gre94] and [GLW96]. The idea is as follows: often the designer has at least a rough idea of what the final shape of the surface should look like. Now suppose an initial surface is available that is close to the final result. This surface will be called the reference surface. The reference surface can be used to define a quadratic fairness functional that is a good approximation to the exact functional for surfaces in the neighborhood of the reference surface. The fairness functional obtained in this way is data dependent, since it depends on the reference surface. Experiments show that this data dependent fairness functional can produce much better results than the simple approximation $E_{int, simple}$ (see section 4.6). The mathematical properties of data dependent fairness functionals are discussed in section 2.4.9, and their usage in variational modeling is discussed in section 4.3.

4.1.5 External energy operators

Analogous to curves, constraints are sometimes not the right tool to deform a minimal energy surface (see section 3.1.6). To solve this problem, also for surfaces a number of special design operators are introduced in section 4.4. These operators (called external energy operators) can be interpreted as external forces on the surface. They are defined as energy terms, which can be added to the fairness functional. The total energy of the surface $E_{tot}$ is thus the sum of the fairness functional $E_{int}$ (the internal energy) and the contributions of the external energy
operators $E_{\text{ext}}$ (the external energy). The corresponding energy minimization problem can be formulated as

$$\min_{F \in \mathcal{F}} E_{\text{tot}}(F) = E_{\text{int}}(F) + E_{\text{ext}}(F),$$  \hspace{1cm} (4.3)

where $\mathcal{F}$ is a suitable function space. The idea behind this approach is that each term of the external energy $E_{\text{ext}}$ will have an intuitive deformation effect on the surface $F$, while the internal energy $E_{\text{int}}$ will take care of a pleasant shape.

### 4.1.6 Surface representations

The choice of the surface representation is an important issue in variational modeling. It has a large influence on the complexity of the energy minimization problem (4.3).

In this thesis only surfaces of the following canonical form will be treated:

$$F(u,v) = \sum_{i=1}^{n} N_i(u,v) P_i, \quad (u,v) \in \Omega,$$  \hspace{1cm} (4.4)

where $\{P_i\}_{i=1}^{n}$ is a set of control points in $\mathbb{R}^3$ and $\{N_i\}_{i=1}^{n}$ a set of basis functions defined on the region $\Omega$ (see also section 3.1.7). Most popular surface representations in CAGD (Bézier surfaces, B-spline surfaces) can be written in this form. Again, we assume that the control points $P_i$ are the shape parameters of the surface. One of the main advantages of control point based surfaces is that the constraints and internal and external energy terms are simple expressions in the control points. Suppose that the $i$-th control point is represented as $P_i = (x_i, y_i, z_i)^T$. Analogous to curves, we define the concatenation vector $P$ of the control points $\{P_i\}$ as

$$P = (x_0, y_0, z_0, \ldots, x_{n-1}, y_{n-1}, z_{n-1})^T.$$  \hspace{1cm} (4.5)

The shape of the surface (4.4) is completely specified by this vector.

In section 3.1.7 a number of criteria is listed that can be used to consider a surface of the form (4.4). These criteria (local support, affine invariance, variation diminishing property, efficient algorithms, refinement and automatic continuity) are also valid for surfaces.

Surfaces in CAGD are often composed of several smaller parts. Just like composite curves are formed by joining segments, composite surfaces are formed by joining patches. Two different types will be considered: rectangular and triangular patches. A rectangular patch is a part of a surface that is the image of a rectangle in the parameter domain and a triangular patch is the image of a triangle in the parameter domain.

A very popular class of surfaces is formed by the tensor product surfaces. The general form of a tensor product surface is

$$F(u,v) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_i(u) B_j(v) P_{ij},$$

with $\{A_i\}_{i=1}^{m}$ and $\{B_j\}_{j=1}^{n}$ sets of arbitrary functions and $P_{ij}$ the control points. Often the functions $A_i$ and $B_j$ are the univariate Bernstein polynomials (3.24) or the univariate B-spline basis functions (3.27). Tensor product surfaces are a very natural transition from curves to
4.1 INTRODUCTION

surfaces. However, there are two disadvantages regarding this representation. In the first place, tensor product surfaces don't support local refinement. For instance, suppose that the functions $A_i$ are B-spline basis functions, and suppose that a knot is inserted between the knots $u_i$ and $u_{i+1}$. The effect of this is that all patches of the form $[u_i, u_{i+1}] \times [v, w]$ for some $v, w$ are refined, instead of only one. In the second place, tensor product surfaces are defined on a rectangular domain, so they are not fitted for surfaces with a complex structure.

For the latter, surface representations based on triangular patches are preferable. Triangular patches are much more flexible and can be locally refined. The latter is important if the level of detail needs to be increased. A disadvantage of triangular patches is that their representation is often more complicated than tensor product surfaces.

A parametric surface $\mathbf{F} = (F_1, F_2, F_3)$ defined on $\Omega \subset \mathbb{R}^2$ is called $C^k$ continuous if its coordinate functions are $C^k$ continuous: $F_i \in C^k(\Omega)$, $i = 1 \ldots 3$. A surface $\mathbf{F}$ is called $C^k$ or $k$-th order geometric continuous if there exists a regular $C^k$ continuous parametrization of $\mathbf{F}$. For a composite surface, sometimes continuity constraints have to be imposed on it, to guarantee that the order of continuity across the patch boundaries is sufficiently high. The reason why is illustrated in figure 4.3.

In section 4.5 some surface representation are treated in more detail.
CHAPTER 4 SURFACES

4.2 Constraints

In variational surface modeling, constraints are an important tool. They can be used to specify certain features of a surface exactly. Most constraints are of a geometric nature: they involve locations of points, distances, directions of tangents etc. It is necessary to apply at least some constraints to the boundary of a surface, since the energy of an unconstrained surface can be made zero by collapsing the surface to a single point (see section 4.3). All of the constraints for curves (see section 3.2) can be applied to surfaces too. The most important new type of constraint is the curve interpolation constraint, which can be used to fix the boundary of a surface.

Just like with curves, we will only consider surfaces having the canonical representation (4.4) that was defined in the introduction:

\[ \mathbf{F}(u, v) = \sum_{i=1}^{n} N_i(u, v) \mathbf{P}_i, \quad (u, v) \in \Omega. \]

The control points \( \mathbf{P}_i \) are the shape parameters that determine the shape of the surface. Since the only difference with the canonical curve representation (3.7) is that the basis functions \( N_i \) depend on two instead of one variable(s), most constraint expressions for curves and surfaces are exactly the same and will not be repeated.

Like constraints on curves, the constraints on surfaces are separated into positional constraints, directional constraints and ‘other’ constraints.

4.2.1 Positional constraints

Positional constraints are constraints that put restrictions on the locations of points on a surface. They are expressed in the \( x, y \) and \( z \)-coordinates of a surface.

The point to point constraint makes the surface interpolate a given point \( \mathbf{P} \), for some parameter value \( (u_0, v_0) \in \Omega \):

\[ \mathbf{F}(u_0, v_0) = \mathbf{P}. \]  

(4.6)

See figure (4.1) for an illustration of this constraint. Analogous to curves, the parameter \( (u_0, v_0) \) is fixed.

The point to line constraint constraint makes a point of a surface interpolate a given line \( l = \{ \mathbf{x} \in \mathbb{R}^3 | \mathbf{x} = \mathbf{r} + t \mathbf{s}, t \in \mathbb{R} \} \), for some parameter value \( (u_0, v_0) \in \Omega \):

\[ \mathbf{F}(u_0, v_0) \in l. \]  

(4.7)

Finally, the point to plane constraint makes a point of a surface interpolate a given plane \( V = \{ \mathbf{x} \in \mathbb{R}^3 | \mathbf{n} \cdot \mathbf{x} = d \} \), for some parameter value \( (u_0, v_0) \in \Omega \). See figure (4.2) for an illustration of this constraint.

\[ \mathbf{F}(u_0, v_0) \in V. \]  

(4.8)

All of these constraints are linear in the components of the control points \( \mathbf{P}_i \), which makes them easy to use for variational modeling.
4.2 Constraints

Figure 4.1: A minimal energy surface that was obtained by interpolating the boundary and by applying two point to point constraints.

Figure 4.2. A minimal energy surface with four point to plane constraints to one and the same plane.

4.2.2 Directional constraints

A normal to vector constraint prescribes the direction $\mathbf{n}$ of the normal vector in a point of a surface with some parameter value $(u_0, v_0) \in \Omega$. In this point, the cross product of the unit normal vector $\mathbf{N}_F$ with $\mathbf{n}$ is zero.

$$\mathbf{N}_F(u_0, v_0) \times \mathbf{n} = \mathbf{0}.$$  

This constraint is often combined with a point to point constraint, to control the location where the normal direction is constrained.

The normal to plane constraint is a relaxed version of the normal to vector constraint. In this case the direction of the normal vector should be perpendicular to a given direction $\mathbf{n}$:

$$\mathbf{N}_F(u_0, v_0) \cdot \mathbf{n} = 0.$$
4.2.3 Continuity constraints

When surface modeling is done with composite surfaces, the resulting surface may fail to be tangent plane continuous at the boundaries of neighboring patches. To guarantee that the surface has overall continuity of sufficiently high degree, continuity constraints have to be applied. Figure 4.3 shows a minimal energy surface with and without a $C^1$ continuity constraint.

Figure 4.3: Left: a minimal energy surface where no continuity constraints are applied to the common boundary of two patches. Right: the same surface, this time minimized with $C^1$ continuity constraints at the patch boundary.

The form that continuity constraints take for a surface depends strongly on its representation. For a surface in the canonical form (4.4) a $C^k$ continuity constraint is often linear in the control points $P_i$, whereas $G^k$ continuity constraints are more complicated. In section 4.5, for some surface representations continuity constraints are treated in more detail.

Just like for curves, continuity constraints should be avoided whenever possible. They typically lead to a large set of constraints on the surface, which is unattractive for the minimization problem.

4.2.4 Curve interpolation constraints

An important constraint for surfaces is the curve interpolation constraint. A curve interpolation constraint makes a curve on the surface, say $F(u(t), v(t))$, $t \in I$, interpolate a given curve $c(t)$, $t \in I$. Curve interpolation constraints can be used to fix the boundary of a surface. Often not only the curve itself is interpolated, but also the normal on the surface along the curve.

The type of curve that can be interpolated depends on the surface representation. In practice, the interpolated curve is often a polynomial curve that coincides with a patch boundary. Just like continuity constraints, curve interpolation constraints depend strongly
4.2 CONSTRAINTS

on the surface representation. In section 4.5, some curve interpolation constraints are treated in more detail.

Apart from the above mentioned constraints, other more complicated ones are possible. Some examples of these are given in section 3.2.4.
4.3 Internal energy

The internal energy of a surface is the part of the total energy (see section 4.1.5) that depends only on properties of the surface itself, like the principal curvatures. It determines the global shape characteristics of the surface. In this section one particular choice for the internal energy is treated, namely a linear combination of the bend energy (4.1) and the stretch energy. This functional (which is also known as the thin plate under tension model, see section 4.1) is often used, see [CG91], [WW92] etc. The internal energy $E_{\text{int}}$ can thus be written as

$$E_{\text{int}}(\mathbf{x}) = \alpha E_{\text{stretch}}(\mathbf{x}) + (1 - \alpha) E_{\text{bend}}(\mathbf{x}),$$  \hspace{1cm} (4.9)

with $\alpha$ a value in the interval $[0,1]$. In general, decreasing $\alpha$ causes the surface area to shrink, and increasing $\alpha$ causes a more balanced spreading of the curvature over the surface.

In the next two sections, the bend and stretch energy and some approximations to them will be treated. The sections 4.3.3 and 4.3.4 deal with some theoretical properties. The general form of unconstrained surfaces with minimal bend or stretch energy will be derived, and we will investigate how the bend and stretch energy behave under scaling and parameter transformations. The latter is important for the choice of the weight factor $\alpha$ in (4.9).

The bend and the stretch energy will be expressed in the control points of the canonical surface representation

$$\mathbf{F}(u,v) = \sum_{i=0}^{n-1} N_i(u,v) \mathbf{P}_i, \quad (u,v) \in \Omega$$

that was introduced in section 4.1.

4.3.1 Bend energy

The bend energy of a surface $\mathbf{F} : \Omega \rightarrow \mathbb{R}^3$ is the integral of the squared principal curvatures

$$E_{\text{bend, \text{external}}} (\mathbf{F}) = \int_{\Omega} \kappa_1^2 + \kappa_2^2 \, d\omega_{\mathbf{F}},$$  \hspace{1cm} (4.10)

with $d\omega_{\mathbf{F}} = \|\mathbf{F}_u \times \mathbf{F}_v\| \, du \, dv$ the infinitesimal surface element (see also section 4.1). For practical applications, this expression is far too complex. Two different approximations will be given, a simple one and a more accurate data dependent one.

For a functionally defined surface $z = f(x,y)$, or $\mathbf{F}(x,y) = (x,y,f(x,y))$, the following simplified version of the thin plate energy can be used:

$$E_{\text{bend, \text{functional}}} (\mathbf{F}) = \int_{\Omega} f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 \, dx \, dy.$$  \hspace{1cm} (4.11)

This is a good approximation of (4.10) if the first derivatives of $f$ are sufficiently small. This can be derived as follows. In section 2.4 it was shown that the square of the principal curvatures $\kappa_1^2 + \kappa_2^2$ is equal to the trace of the matrix $(\mathbf{I}_\mathbf{F}^{-1} \mathbf{I}_\mathbf{F})^2$, where $\mathbf{I}_\mathbf{F}^{-1} \mathbf{I}_\mathbf{F}$ is the matrix
4.3 INTERNAL ENERGY

representation of the Weingarten map with respect to the basis \( \{ \mathbf{F}_x, \mathbf{F}_y \} \) of the tangent space. Substitution of \( \mathbf{F} \) into this matrix yields

\[
I_F^{-1} \mathbf{F} = \begin{pmatrix}
F_{xx} & F_{xy} & F_{x} \\
F_{xy} & F_{yy} & F_{y} \\
F_{x} & F_{y} & F
\end{pmatrix}\begin{pmatrix}
F_x \cdot \mathbf{n}_F & F_y \cdot \mathbf{n}_F \\
F_{xx} \cdot \mathbf{n}_F & F_{xy} \cdot \mathbf{n}_F \\
F_{xy} \cdot \mathbf{n}_F & F_{yy} \cdot \mathbf{n}_F
\end{pmatrix}^{-1}
\]

\[
= \frac{1}{\sqrt{1 + f_x^2 + f_y^2}} \begin{pmatrix}
1 + f_x^2 & f_x f_y \\
1 + f_y^2 & f_y
\end{pmatrix}
\begin{pmatrix}
f_{xx} & f_{xy} \\
0 & 1
\end{pmatrix}
\]

\[
= \begin{pmatrix}
f_{xx} & f_{xy} \\
f_{xy} & f_{yy}
\end{pmatrix}\left(1 + \mathcal{O}(f_x^2 + f_y^2)\right), \quad f_x^2 + f_y^2 \to 0.
\]

Hence, \( \kappa_1^2 + \kappa_2^2 = \text{trace} \left( (I_F^{-1} \mathbf{F})^2 \right) = f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 + \mathcal{O}(f_x^2 + f_y^2) \). If we combine this with

\[
\|\mathbf{F}_x \times \mathbf{F}_y\| = \sqrt{1 + f_x^2 + f_y^2} = 1 + \mathcal{O}(f_x^2 + f_y^2),
\]

the approximation (4.11) is found.

**Simple approximation**

Inspired by the above result, for a general parametric surface \( \mathbf{F} : \Omega \to \mathbb{R}^3 \) often the following simplified version of the bend energy is used (see [CG91] and [WW92])

\[
E_{\text{bend simple}}(\mathbf{F}) = \int_{\Omega} F_{uu} \cdot \mathbf{F}_{uu} + 2F_{uv} \cdot \mathbf{F}_{uv} + F_{vv} \cdot \mathbf{F}_{vv} dudv. \tag{4.12}
\]

For an *isometric surface*, defined by the relation

\[
I_F = \begin{pmatrix}
\mathbf{F}_u \cdot \mathbf{F}_u & \mathbf{F}_u \cdot \mathbf{F}_v \\
\mathbf{F}_u \cdot \mathbf{F}_v & \mathbf{F}_v \cdot \mathbf{F}_v
\end{pmatrix} = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}, \quad \tag{4.13}
\]

the approximation (4.12) is exact, i.e. \( E_{\text{bend simple}}(\mathbf{F}) = E_{\text{bend exact}}(\mathbf{F}) \). This can be derived as follows. By taking partial derivatives with respect to \( u \) and \( v \) of the relation (4.13), it follows that the second derivative vectors \( \mathbf{F}_{uu}, \mathbf{F}_{uv} \) and \( \mathbf{F}_{vv} \) are parallel to the normal vector \( \mathbf{n}_F \). Therefore,

\[
\text{trace} \left( (I_F^{-1} \mathbf{F})^2 \right) = \text{trace} \left( \left( \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} F_{uu} \cdot \mathbf{n}_F & F_{uv} \cdot \mathbf{n}_F \\ F_{uv} \cdot \mathbf{n}_F & F_{vv} \cdot \mathbf{n}_F \end{pmatrix} \right)^2 \right)
\]

\[
= \text{trace} \left( \begin{pmatrix} (F_{uu} \cdot \mathbf{n}_F)^2 & (F_{uv} \cdot \mathbf{n}_F)^2 \\ (F_{uv} \cdot \mathbf{n}_F)^2 & (F_{vv} \cdot \mathbf{n}_F)^2 \end{pmatrix} \right)
\]

\[
= (F_{uu} \cdot \mathbf{n}_F)^2 + (F_{uv} \cdot \mathbf{n}_F)^2 + (F_{vv} \cdot \mathbf{n}_F)^2,
\]

\[
\text{trace} \left( (I_F^{-1} \mathbf{F})^2 \right) = \text{trace} \left( \begin{pmatrix} (F_{uu} \cdot \mathbf{n}_F)^2 & \cdots \\ \cdots & (F_{vv} \cdot \mathbf{n}_F)^2 \end{pmatrix} \right)
\]

\[
= \sum_{i,j} (F_{ij} \cdot \mathbf{n}_F)^2
\]

\[
= \mathbf{n}_F^T \left( \mathbf{F}_{uu} \cdot \mathbf{n}_F \right)^2 + 2\mathbf{n}_F^T \left( \mathbf{F}_{uv} \cdot \mathbf{n}_F \right)^2 + \mathbf{n}_F^T \left( \mathbf{F}_{vv} \cdot \mathbf{n}_F \right)^2.
\]
\[\begin{align*}
    &= \text{trace}\left( (F_{uu} \cdot F_{uu} + (F_{uv} \cdot F_{uv}) + \cdots \right) \\
    &= F_{uu} \cdot F_{uu} + 2F_{uv} \cdot F_{uv} + F_{vv} \cdot F_{vv}.
\end{align*}\]

Together with the fact that an isometric surface \( \mathbf{F} \) satisfies \( d\omega_{\mathbf{F}} = dudv \), this proves that for an isometric surface the approximation (4.12) is exact.

The simple approximation of the bend energy (4.12) suffers from two problems. In the first place, it is parameterization dependent, which means that two surfaces with the same shape can have different values of the (approximated) bend energy. In the second place, the approximation (4.12) can be rather poor. This topic is extensively treated in [GLW96], where also a number of examples are given.

The reason that the approximation (4.12) is quite popular is that it leads to a particularly simple expression for a surface with the canonical representation (4.4). If this representation is substituted into (4.12), a quadratic expression in the control points \( \mathbf{P}_i \) is found:

\[E_{\text{bend, simple}}(\mathbf{F}) = \sum_{i,j} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j,\]

with

\[a_{ij} = \int_{\Omega} N_{i uu} N_{j uu} + 2N_{i uv} N_{j uv} + N_{i vv} N_{j vv} dudv.\]  \hfill (4.14)

Quadratic functions are desirable, since they can be efficiently minimized (see also section 4.6).

The matrix \( A_{\text{bend, simple}} = [a_{ij}] \) has the following properties:

- \( A_{\text{bend, simple}} \) is symmetric.

- \( A_{\text{bend, simple}} \) is positive semidefinite. This follows from the fact that the bend energy \( E_{\text{bend, simple}}(\mathbf{F}) \) is always positive.

**Data dependent approximation**

There is a big gap between the exact expression for the bend energy \( E_{\text{bend, exact}}(\mathbf{F}) \) (which is far too complicated to be used for interactive modeling) and the approximation \( E_{\text{bend, simple}}(\mathbf{F}) \) (which is well suited for interactive modeling, but may lead to poor results). In [GLW96] a reasonable compromise between these two functionals is proposed. The designer has to select a reference surface \( \mathbf{G} \), which is a surface that is close to the final result. This surface is used to define a quadratic approximation \( E_{\text{bend, data}} \) that is a good approximation to \( E_{\text{bend, exact}} \) for surfaces in the neighborhood of the chosen reference surface \( \mathbf{G} \). The data dependent functional \( E_{\text{bend, data}} \) is given by

\[E_{\text{bend, data}}(\mathbf{F}; \mathbf{G}) = \int_{\Omega} \sum_{i=1}^{3} \text{trace}\left( (\text{Hess}_{\mathbf{G}}(F_i))^2 \right) d\omega_{\mathbf{G}}. \] \hfill (4.15)

In section 2.4 it is shown that this functional has the following properties:
4.3 INTERNAL ENERGY

- It is exact for $\mathbf{G} = \mathbf{F}$, i.e. $E_{\text{bend data}}(\mathbf{F}; \mathbf{F}) = E_{\text{bend exact}}(\mathbf{F})$
- It is invariant under a parameter transformation $\varphi$ of $\mathbf{F}$ and $\mathbf{G}$:
  $$E_{\text{bend data}}(\mathbf{F} \circ \varphi; \mathbf{G} \circ \varphi) = E_{\text{bend data}}(\mathbf{F}; \mathbf{G}).$$

The first property suggests that the functional $E_{\text{bend data}}(\mathbf{F}; \mathbf{G})$ is locally a good approximation to the exact functional $E_{\text{bend exact}}(\mathbf{F})$, since the functional $E_{\text{bend data}}(\mathbf{F}; \mathbf{G})$ depends continuously on $\mathbf{G}$. It is probably possible to prove this mathematically by defining a norm $\| \cdot \|_G$ similar to the norm for curves (2.19), with the property that

$$|E_{\text{bend data}}(\mathbf{G} + \xi; \mathbf{G}) - E_{\text{bend exact}}(\mathbf{G})| \leq K \| \xi \|_G .$$

The second property guarantees that the approximation $E_{\text{bend data}}(\mathbf{F}; \mathbf{G})$ only depends on the shape of $\mathbf{F}$ and $\mathbf{G}$ and not on the parametrization. However, the expression $E_{\text{bend data}}(\mathbf{F}; \mathbf{G})$ is not completely defined by the shape of $\mathbf{F}$ and $\mathbf{G}$: also a one-to-one correspondence between the points from $\mathbf{F}$ and the points from $\mathbf{G}$ is needed.

It is worth noting that the simple approximation (4.12) is in fact a special case of the data dependent approximation (4.15). If the reference surface $\mathbf{G}$ is an arbitrary isometric surface, (for example $\mathbf{G}(u, v) = (u, v, 0)$, $(u, v) \in \Omega$) the approximations coincide: $E_{\text{bend data}}(\mathbf{F}; \mathbf{G}) = E_{\text{bend simple}}(\mathbf{F})$. This explains why the data dependent approximation yields better results than the simple approximation: in general, one can easily find a reference surface $\mathbf{G}$ that is closer to $\mathbf{F}$ than an arbitrary isometric surface!

If we substitute the canonical surface representation (4.4) into (4.15), a quadratic expression in the control points $\mathbf{P}_i$ is found:

$$E_{\text{bend data}}(\mathbf{F}) = \sum_{i,j} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j$$

with

$$a_{ij} = \int_\Omega \text{trace} \left( \text{Hess}_\mathbf{G}(N_i) \text{Hess}_\mathbf{G}(N_j) \right) d\omega_\mathbf{G} .$$ (4.16)

The coefficients $a_{ij}$ normally cannot be computed analytically. They can be approximated using the numerical integration routines given in section (2.5.5). The matrix $A_{\text{bend data}} = [a_{ij}]$ has the same properties as the matrix $A_{\text{bend simple}}$.

4.3.2 Stretch energy

The stretch energy of a surface $\mathbf{F}: \Omega \to \mathbb{R}^3$ is defined as the surface area

$$E_{\text{stretch exact}}(\mathbf{F}) = \int_\Omega d\omega_\mathbf{F} .$$ (4.17)

Just like the bend energy $E_{\text{bend exact}}$, this expression is too complex for practical applications. Also for the stretch energy two different approximations will be given, a simple one and a more accurate data dependent one.
CHAPTER 4 SURFACES

Simple approximation

The following simplified version of the stretch energy is often used:

\[ E_{\text{stretch\_simple}}(\mathbf{F}) = \int_\Omega \mathbf{F}_u \mathbf{F}_u + \mathbf{F}_v \mathbf{F}_v \, du \, dv \] (4.18)

Analogous to the approximation \( E_{\text{bend\_simple}}(\mathbf{F}) \), the approximation (4.18) is exact (i.e. \( E_{\text{stretch\_simple}}(\mathbf{F}) = E_{\text{stretch\_exact}}(\mathbf{F}) \)) for isometric surfaces. The simple approximation of the stretch energy (4.12) also suffers from the fact that it is parametrization dependent, and that this approximation can be rather poor.

If we substitute the canonical surface representation (4.4), a quadratic expression in the control points \( \mathbf{P}_i \) is found:

\[ E_{\text{stretch\_simple}}(\mathbf{F}) = \sum_{i,j} a_{ij} \mathbf{P}_i \mathbf{P}_j \]

with

\[ a_{ij} = \int_\Omega N_{i_u} N_{j_u} + N_{i_v} N_{j_v} \, du \, dv. \] (4.19)

Data dependent approximation

Similar to the bend energy, a data dependent approximation of the stretch energy can be defined as follows. Let \( \mathbf{G} \) be a reference surface, then

\[ E_{\text{stretch\_data}}(\mathbf{F}; \mathbf{G}) = \frac{1}{2} \sum_{i=1}^{3} \int_\Omega \nabla_\mathbf{G}(F_i) : \nabla_\mathbf{G}(F_i) \, d\omega_\mathbf{G} \]

\{using (2.8)\} \[ = \int_\Omega \text{trace}(I_{\mathbf{G}}^{-1}I_{\mathbf{F}}) \, d\omega_\mathbf{G}. \] (4.20)

In section 2.4 it is shown that this functional has the following properties:

- It is exact for \( \mathbf{G} = \mathbf{F} \), i.e. \( E_{\text{stretch\_data}}(\mathbf{F}; \mathbf{F}) = E_{\text{stretch\_exact}}(\mathbf{F}) \)
- It is invariant under a parameter transformation \( \varphi \) of \( \mathbf{F} \) and \( \mathbf{G} \):

\[ E_{\text{stretch\_data}}(\mathbf{F}; \mathbf{G}) = E_{\text{stretch\_data}}(\mathbf{F}; \mathbf{G}). \]

The expression (4.20) is a good approximation of (4.17) provided that the distance between \( \mathbf{F} \) and \( \mathbf{G} \) is small (see section 2.4). If the reference surface \( \mathbf{G} \) is an arbitrary isometric surface, the simple and the data dependent approximations coincide.

Again, the simple approximation \( E_{\text{stretch\_simple}}(\mathbf{F}) \) is a special case of the data dependent approximation \( E_{\text{stretch\_data}}(\mathbf{F}; \mathbf{G}) \). Let \( \mathbf{G} \) be an arbitrary isometric reference surface, then

\[ E_{\text{stretch\_data}}(\mathbf{F}; \mathbf{G}) = \int_\Omega \text{trace}(I_{\mathbf{G}}^{-1}I_{\mathbf{F}}) \, d\omega_\mathbf{G} \]

\{\mathbf{G} \text{ is isometric}\} \[ = \int_\Omega \text{trace}(I_{\mathbf{F}}) \, du \, dv \]

\[ = E_{\text{stretch\_simple}}(\mathbf{F}). \]
4.3 INTERNAL ENERGY

If we substitute the canonical surface representation (4.4) into (4.20), a quadratic expression in the control points $P_i$ is found:

$$E_{\text{stretch, data}}(F; G) = \sum_{i,j} a_{ij} P_i P_j$$

with

$$a_{ij} = \frac{1}{2} \int_{\Omega} \text{grad}_G(N_i) \cdot \text{grad}_G(N_j) d\omega_G.$$  \hspace{1cm} (4.21)

The coefficients $a_{ij}$ also have to be approximated using the numerical integration routines given in section (2.5.5).

4.3.3 Surfaces with minimal bend or stretch energy

Although in modeling situations always constraints will be applied to a surface, it is important to know about unconstrained surfaces with minimal bend or stretch energy. This will give us more insight into the kind of surfaces these functionals are striving for when they are minimized subject to geometric constraints.

Surfaces with minimal stretch energy

The exact expression for the stretch energy $E_{\text{stretch, exact}}(F)$ of a surface $F$ is zero if the surface area is zero. This is the case if $F$ is a single point or a one-dimensional curve. The simple approximation for the stretch energy $E_{\text{stretch, simple}}(F)$ is zero if $F_u F_u + F_v F_v = 0$. This means that $F$ is a single point. Finally, the data dependent approximation for the stretch energy $E_{\text{stretch, data}}(F; G)$ is zero if $\text{trace}(I_G^{-1} I_F) = 0$. Since $I_G^{-1}$ is positive definite, and $I_F$ is positive semidefinite, this can only be true if $I_F$ is the zero matrix. Hence, also in this case the surface $F$ must be a single point.

Surfaces with minimal bend energy

The exact expression for the bend energy $E_{\text{bend, exact}}(F)$ is zero if $\kappa_1^2 + \kappa_2^2 = 0$. This means that $F$ is a part of a plane. The simple approximation for the bend energy $E_{\text{bend, simple}}(F)$ is zero if $F_{uu} F_{uu} + 2F_{uv} F_{uv} + F_{vv} F_{vv} = 0$. This means that $F$ is a linear function in $u$ and $v$: $F(u, v) = A + uB + vC$ (which implies that $F$ is a part of a plane). Under what conditions the data dependent approximation for the bend energy $E_{\text{bend, data}}(F; G)$ is zero hasn’t been investigated, but it will probably also mean that $F$ is part of a plane. Note that for both approximations the reverse statement is not true: the fact that a surface $F$ is a part of a plane does not imply that the (approximated) bend energy is zero!

4.3.4 The effect of scaling and parameter transformations

The internal energy of a surface (4.9) is a weighted sum of the bend and the stretch energy. Since these quantities are very different, the choice of the weight factor $\alpha$ is not straightforward. In this section we will investigate how the bend and the stretch energy behave under scaling of a surface and under parameter transformations.
CHAPTER 4 SURFACES

Scaling

Let $\mathbf{F}$ be a surface defined on the region $\Omega$ and let $\mathbf{G}$ be a reference surface for $\mathbf{F}$ defined on the same region. When $\mathbf{F}$ is compared with the scaled surface $K\mathbf{F}$, the following relations are found:

\[
E_{\text{stretch}_{\text{exact}}}(K\mathbf{F}) = K^2 E_{\text{stretch}_{\text{exact}}}(\mathbf{F}) \\
E_{\text{stretch}_{\text{simple}}}(K\mathbf{F}) = K^2 E_{\text{stretch}_{\text{simple}}}(\mathbf{F}) \\
E_{\text{stretch}_{\text{data}}}(K\mathbf{F}; K\mathbf{G}) = K^2 E_{\text{stretch}_{\text{data}}}(\mathbf{F}; \mathbf{G}) \\
E_{\text{bend}_{\text{exact}}}(K\mathbf{F}) = E_{\text{bend}_{\text{exact}}}(\mathbf{F}) \\
E_{\text{bend}_{\text{simple}}}(K\mathbf{F}) = K^2 E_{\text{bend}_{\text{simple}}}(\mathbf{F}) \\
E_{\text{bend}_{\text{data}}}(K\mathbf{F}; K\mathbf{G}) = E_{\text{bend}_{\text{data}}}(\mathbf{F}; \mathbf{G})
\]

Note that the bend and stretch energy scale differently for curves and surfaces (see section 3.3.4)! To account for the scaling of a surface with factor $K$, two things can be done: the scaling can be filtered out by multiplying $E_{\text{bend}_{\text{exact}}}$ and $E_{\text{bend}_{\text{data}}}$ with $K^2$, or the simple approximation $E_{\text{bend}_{\text{simple}}}$ can be brought into agreement with the exact functional by dividing it by $K^2$.

Parameter transformations

The exact expressions for the bend and stretch energy are geometric properties of a surface. Therefore they are invariant under any regular parameter transformation $\Phi : \Gamma \subset \mathbb{R}^2 \rightarrow \Omega$ of the surface $\mathbf{F}$:

\[
E_{\text{stretch}_{\text{exact}}}(\mathbf{F} \circ \Phi) = E_{\text{stretch}_{\text{exact}}}(\mathbf{F}), \\
E_{\text{bend}_{\text{exact}}}(\mathbf{F} \circ \Phi) = E_{\text{bend}_{\text{exact}}}(\mathbf{F}).
\]

The same is true for the data dependent approximations to the bend and stretch energy. Because of the geometric nature of the gradient and the Hessian, the expressions $\text{Hess}_\mathbf{G}(\mathbf{F}_i)$ and $\text{grad}_\mathbf{G}(\mathbf{F}_i)$ are independent of the parametrization. Therefore

\[
E_{\text{stretch}_{\text{data}}}(\mathbf{F} \circ \Phi; \mathbf{G} \circ \Phi) = E_{\text{stretch}_{\text{data}}}(\mathbf{F}; \mathbf{G}), \\
E_{\text{bend}_{\text{data}}}(\mathbf{F} \circ \Phi; \mathbf{G} \circ \Phi) = E_{\text{bend}_{\text{data}}}(\mathbf{F}; \mathbf{G}).
\]

However, the simple approximations to the bend and stretch energy are not invariant under parameter transformations. We will consider the following special class of parameter transformations: $\Phi : \mathbf{x} \rightarrow cA\mathbf{x} + \mathbf{b}$, with $A \in \mathbb{R}^{2 \times 2}$ an orthogonal matrix, i.e. $A^T A = A A^T = I_2$. If $c = 1$, $\Phi$ is a rigid movement and if $A = I_2$, $\Phi$ is a scaling with factor $c$.

For convenience, we adopt the following notations:

\[
D \mathbf{F} = \begin{pmatrix} \mathbf{F}_u & \mathbf{F}_v \end{pmatrix}
\]
4.3 **Internal Energy**

\[ D^2 F = \begin{pmatrix} F_{uu} & F_{uv} \\ F_{uv} & F_{vv} \end{pmatrix} \]

If we interpret these matrices as elements of \((\mathbb{R}^3)^{1 \times 2}\) and \((\mathbb{R}^3)^{2 \times 2}\) respectively, the simple approximations can be written as follows:

\[
E_{\text{stretch\_simple}}(F) = \int_{\Omega} D F (D F)^T \, du dv \\
E_{\text{bend\_simple}}(F) = \int_{\Omega} \text{trace} \left( D^2 F \left( D^2 F \right)^T \right) \, du dv.
\]

Now let \( \hat{F} = F \circ \Phi \). If \( \Phi \) is a rigid movement, then we find

\[
D \hat{F} D \hat{F}^T = c^2 D F A A^T (D F)^T \circ \Phi \\
= c^2 D F (D F)^T \circ \Phi
\]

and

\[
\text{trace} \left( D^2 \hat{F} D^2 \hat{F}^T \right) = c^4 \text{trace} \left( A^T (D^2 F) A A^T (D^2 F)^T A \right) \circ \Phi \\
= c^4 \text{trace} \left( A^T (D^2 F) (D^2 F)^T A \right) \circ \Phi \\
= c^4 \text{trace} \left( D^2 F D^2 F^T \right) \circ \Phi.
\]

Since \(|\det(D \Phi)| = c\), we can conclude that for a rigid movement \((c = 1)\)

\[
E_{\text{stretch\_simple}}(F \circ \Phi) = E_{\text{stretch\_simple}}(F) \\
E_{\text{bend\_simple}}(F \circ \Phi) = E_{\text{bend\_simple}}(F).
\]

and for a scaling \((A = I_2)\)

\[
E_{\text{stretch\_simple}}(F \circ \Phi) = c E_{\text{stretch\_simple}}(F) \\
E_{\text{bend\_simple}}(F \circ \Phi) = c^3 E_{\text{bend\_simple}}(F).
\]

A scaling of the parametrization can therefore be compensated for by dividing the bend energy by \(c^2\). Note that when the parametrization is scaled with different factors in the \(u\) and \(v\) direction, no such simple relations exists.
4.4 External energy

A minimal energy surface is not always easy to deform using constraints only. Analogous to curves (see section 3.4), in this section a number of external energy operators is introduced, that are designed to have an intuitive deformation effect on a minimal energy surface. An external energy operator is defined as a functional that can be added to the internal energy of the surface (see section 4.1.5). Each external energy functional has a weight factor that should be under control of the designer. Increasing the weight factor makes the effect of the corresponding operator stronger.

Most of the operators have some kind of attracting behavior (for example attraction towards a given point in space). For all these ‘attractors’ a repelling counterpart can be defined that has the opposite effect. Like the constraints, the external energy operators will be subdivided in positional operators (operators that act on the location of points of the surface) and directional operators (operators that act on normal directions of the surface).

The external energy operators will be expressed in the control points of the canonical surface representation (4.4)

$$\mathbf{F}(u, v) = \sum_{i=0}^{n-1} N_i(u, v) \mathbf{P}_i.$$  

Since a lot of external energy operators for curves and surfaces are the same, the corresponding expressions in terms of the control points $\mathbf{P}_i$ will not always be repeated here.

4.4.1 Positional attractors

Positional attractors are design operators that exert some kind of attracting force on the location of points on a surface.

The point to point attractor is a design operator that pulls a point of a surface towards a given point in space. Analogous to curves, the energy term that corresponds with the point to point attractor is given by:

$$E_{point\rightarrow point}(F) = d(\mathbf{F}(u_0, v_0), \mathbf{p})^2 = \|\mathbf{F}(u_0, v_0) - \mathbf{p}\|^2.$$  

Similarly, a point to line and a point to plane attractor can be defined. A point to line attractor pulls a point of a surface towards a given line $l = \{x \in \mathbb{R}^3 | x = r + ts, t \in \mathbb{R}\}$:

$$E_{point\rightarrow line}(F) = d(\mathbf{F}(u_0, v_0), \mathbf{p})^2 = \|A\mathbf{F}(u_0, v_0) - \mathbf{b}\|^2,$$

where $A \in \mathbb{R}^{3 \times 3}$ and $\mathbf{b}$ are given by (2.42).

A point to plane attractor is a design operator that pulls a point of a surface towards a given plane $V = \{x \in \mathbb{R}^3 | \mathbf{n} \cdot x = d\}$:

$$E_{point\rightarrow plane}(F) = d(\mathbf{F}(u_0, v_0), \mathbf{V})^2 = \frac{|\mathbf{n} \cdot \mathbf{F}(u_0, v_0) - d|^2}{\mathbf{n} \cdot \mathbf{n}}.$$  

These operators are all quadratic in the components of the control points $\mathbf{P}_i$ of the canonical surface representation (4.4). Their expressions in terms of the control points are the same as for curves and can be found in section 3.4.
4.4 EXTERNAL ENERGY

A curve to point attractor is the same as a point to point attractor, except that it operates on a curve on the surface $F(u(t), v(t))$, $t \in I$, instead of on a single point. The energy term that corresponds with the curve to point attractor is the integral over the squared distance between the curve and the point $p$:

$$E_{\text{curve to point}}(F) = \int_I d(F(u(t), v(t)), p)^2 \, dt = \int_I \|F(u(t), v(t)) - p\|^2 \, dt.$$ 

Although attracting a curve towards a point doesn’t seem to be a very useful operation, it is defined here for completeness. More useful variants of the curve to point attractor are the curve to line, the curve to plane and the curve to curve attractor, that can be defined similarly:

$$E_{\text{curve to line}}(F) = \int_I d(F(u(t), v(t)), l)^2 \, dt = \int_I \|A F(u(t), v(t)) + b\|^2 \, dt,$$

$$E_{\text{curve to plane}}(F) = \int_I d(F(u(t), v(t)), V)^2 \, dt = \int_I |\mathbf{n} \cdot F(u(t), v(t)) - d|^2 \, dt,$$

$$E_{\text{curve to curve}}(F) = \int_I d(F(u(t), v(t)), c(t))^2 \, dt = \int_I \|F(u(t), v(t)) - c(t)\|^2 \, dt,$$

where $A \in \mathbb{R}^{3 \times 3}$ and $b$ are given by (2.42), $V = \{x \in \mathbb{R}^3 | \mathbf{n} \cdot x = d\}$ is a given plane and $c$ is a given curve in $\mathbb{R}^3$. In practical applications only a restricted class of curves $(u(t), v(t))$ in the domain of the surface will be allowed (usually consisting of certain line segments, see also section 4.6). The restrictions are chosen such that the corresponding space curve on the surface can be written in the canonical curve representation (3.7):

$$F(u(t), v(t)) = \sum_k M_k(t) P_{ik},$$

where $\{M_k\}$ is a set of piecewise polynomial functions, and $\{P_{ik}\}$ is a subset of the control points of the surface. Under this condition, the energy terms for these operators are exactly the same as the energy terms for the similar operators that were defined for curve segments in section 3.4. Hence they are all quadratic in the components of the control points of the canonical surface representation (4.4).

A region to point attractor is the same as a point to point attractor, except that it operates on a region of the surface with parameter domain $\Gamma \subset \Omega$. The energy term corresponding to it is given by

$$E_{\text{region to point}}(F) = \int_\Gamma d(F(u, v), p)^2 \, du \, dv = \int_\Gamma \|F(u, v) - p\|^2 \, du \, dv.$$ 

Similarly, a region to line, a region to plane and a region to region attractor can be defined. These operators are also quadratic in the control points of the canonical surface representation (4.4). The coefficients of the quadratic expressions are integrals over the domain $\Gamma$. In figure (4.4) an illustration of some of these attractors is given.
4.4.2 Directional attractors

Directional attractors are design operators that exert some kind of attracting force on the tangent direction of points on a surface.

The **normal to vector attractor** attracts the normal vector in a point of the surface with a fixed parameter \((u_0, v_0)\) towards a given direction \(\mathbf{n}\). The quantity \(\|N_{\mathbf{F}}(u_0, v_0) \cdot \mathbf{n}\|^2\) is a measure for the orthogonality between the unit normal vector \(N_{\mathbf{F}}(u_0, v_0)\) and the vector \(\mathbf{n}\). Since this expression is rather complicated in the control points of the canonical surface representation (4.4), for practical applications, the following approximation is used:

\[
E_{\text{normal \_to \_vector}}(\mathbf{F}) = (\mathbf{F}_u(u_0, v_0) \cdot \mathbf{n})^2 + (\mathbf{F}_v(u_0, v_0) \cdot \mathbf{n})^2.
\]

This expression is quadratic in the components of the control points of the canonical surface representation:

\[
E_{\text{normal \_to \_vector}}(\mathbf{F}) = \sum_{i,j} P_i^T A_{ij} P_j,
\]

with

\[
A_{ij} \in \mathbb{R}^{3 \times 3} = (N_{i_u}(u_0, v_0)N_{i_u}(u_0, v_0) + N_{i_v}(u_0, v_0)N_{i_v}(u_0, v_0))\mathbf{nn}^T.
\]

The **normal region to vector attractor** is the same as the normal to vector attractor, except that it operates on a region of the surface with parameter domain \(\Gamma \subset \Omega\). The energy term corresponding to it is given by

\[
E_{\text{normal \_region \_to \_vector}}(\mathbf{F}) = \int_{\Gamma} (\mathbf{F}_u(u, v) \cdot \mathbf{n})^2 + (\mathbf{F}_v(u, v) \cdot \mathbf{n})^2 dudv.
\]

This operator is also quadratic in the components of the control points of the canonical surface representation (4.4):

\[
E_{\text{normal \_region \_to \_vector}}(\mathbf{F}) = \sum_{i,j} P_i^T A_{ij} P_j,
\]

with

\[
A_{ij} \in \mathbb{R}^{3 \times 3} = \int_{\Gamma} (N_{i_u}(u, v)N_{i_u}(u, v) + N_{i_v}(u, v)N_{i_v}(u, v)) dudv \mathbf{nn}^T.
\]

**Repellors**

The external energy operators that we encountered so far all have some attracting behavior. For each of these operators a repelling counterpart exists with the opposite effect.

Analogous to curves (see section 3.4.3), the repelling behavior is achieved by taking the reciprocal of each attractor energy term. Since the given attractors are all quadratic in the control points \(P_i\), the repeller terms are non quadratic functions, which makes the energy minimization more difficult. Figure 4.5 shows an example of what can be done with a region to plane repellor.
4.4 EXTERNAL ENERGY

Figure 4.4: Minimal energy surface deformed by a region to line, a region to plane and a normal region to plane attractor. In the left figure the regions of the surface to which the operators are applied are colored dark. In the right figure it can be seen that each operator has a clearly visible effect.

Figure 4.5: A minimal energy surface deformed by a region to plane repellant. The boundary of the surface and the point at the center are fixed using interpolation constraints. For the right surface the weight of the repellant is increased, which makes the effect of the operator stronger. The surfaces are colored according to their curvature (dark regions imply high values of the curvature).

4.4.3 The effect of scaling and parameter transformations

In section 4.3.4 the effect of scaling and parameter transformations on the internal energy was investigated. Now we will investigate the effect of these transformations on the external energy (see also section 3.3.4).

Scaling

When the surface $\mathbf{F}$ is compared with the scaled surface $K\mathbf{F}$, the following relations are found:

\[
E_{\text{positional\_attractor}}(K\mathbf{F}) = K^2 E_{\text{positional\_attractor}}(\mathbf{F})
\]
\[
E_{\text{directional\_attractor}}(K\mathbf{F}) = K^2 E_{\text{directional\_attractor}}(\mathbf{F})
\]
\[
E_{\text{positional\_repellor}}(K\mathbf{F}) = (1/K^2) E_{\text{positional\_repellor}}(\mathbf{F})
\]
\[ E_{\text{directional repellor}}(K \mathbf{F}) = \left(\frac{1}{K^2}\right) E_{\text{directional repellor}}(\mathbf{F}), \]

where \( E_{\text{positional attractor}} \) is the energy term of one of the positional attractors, \( E_{\text{directional attractor}} \) is the energy term of one of the directional attractors, etc. Because of the completely different behavior of the attractors and repellors, it seems a good idea to adapt their weight factors after a scaling of the surface.

**Parameter transformations**

We will consider the same class of parameter transformations as in section 4.3.4. Let \( \varphi \) be the mapping given by \( \varphi : \mathbf{x} \rightarrow cA\mathbf{x} + \mathbf{b} \), with \( A \in \mathbb{R}^{2 \times 2} \) an orthogonal matrix, i.e. \( A^T A = AA^T = I_2 \). If \( c = 1 \), \( \varphi \) is a rigid movement and if \( A = I_2 \), \( \varphi \) is a scaling with factor \( c \).

If this parameter transformation is applied to a surface \( \mathbf{F} \), the following relations are found:

\[
\begin{align*}
E_{\text{positional attractor}}(\mathbf{F} \circ \varphi) &= (1/c) E_{\text{positional attractor}}(\mathbf{F}), \\
E_{\text{directional attractor}}(\mathbf{F} \circ \varphi) &= cE_{\text{directional attractor}}(\mathbf{F}), \\
E_{\text{positional repellor}}(\mathbf{F} \circ \varphi) &= (1/c) E_{\text{positional repellor}}(\mathbf{F}), \\
E_{\text{directional repellor}}(\mathbf{F} \circ \varphi) &= (1/c^3) E_{\text{directional repellor}}(\mathbf{F}).
\end{align*}
\]

To cancel the effect of such a parameter transformation \( \varphi \), the weights corresponding to the operators should be multiplied with \( c, 1/c, c \) and \( c^3 \) respectively.
4.5 Surface representations

The choice of the surface representation is an important issue in variational modeling. It has a large influence on the solvability of the energy minimization problem. In this section an overview is given of three surface representations that are often used in CAGD: tensor product B-spline surfaces, triangular Bézier surfaces and triangular B-spline surfaces. These surface representations can all be written in the canonical representation (4.4). The suitability of these surface representations for variational modeling will be investigated using the properties listed in section (3.1.7).

4.5.1 Tensor product B-spline surfaces

Tensor product B-spline surface are obtained as a straightforward generalization of the B-spline curves defined in section 3.5.2. A degree \((d, d)\) tensor product B-spline surface is a piecewise polynomial surface defined over a rectangular domain. It is defined by two knotsets \(\{u_0, \ldots, u_{m+2d-2}\}\) and \(\{v_0, \ldots, v_{n+2d-2}\}\) and a number of control points \(P_{ij}\) in \(\mathbb{R}^3\) and it has the following representation

\[
F(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} N_{ij}^d(u, v),
\]

where the basis functions \(N_{ij}^d(u, v) = N_i^d(u; \{u_k\})N_j^d(v; \{v_k\})\) are products of univariate B-spline basis functions as defined in (3.27). Note that a tensor product B-spline surface can be easily written into the canonical representation (4.4) that was used in previous sections. The surface is defined over the rectangular domain \([u_{d-1}, u_{m+d-1}] \times [v_{d-1}, v_{n+d-1}]\) and it can be divided into \(mn\) polynomial patches with domains \([u_{d+i-1}, u_{d+i}] \times [v_{d+j-1}, v_{d+j}]\), \(0 \leq i < m, \ 0 \leq j < n\).

The boundary of a tensor product B-spline surface is formed by the coordinate curves \(u = u_{d-1}, u = u_{m+d-1}, v = v_{d-1}\) and \(v = v_{n+d-1}\). These curves are degree \(d\) B-spline curves. For example, the boundary curve \(v = v_{d-1}\) is a B-spline curve with control points \(Q_i = \sum_{j=0}^{n-1} N_{ij}^d(v_{d-1}, \{v_k\})P_{ij}\).

A tensor product B-spline surface has the following properties, most of them inherited from B-spline curves (see also section 3.1.7)

- **Local support** The basis function \(N_{ij}^d\) is nonzero on the rectangle \([u_{i-1}, u_{i+d}] \times [v_{j-1}, v_{j+d}]\).

- **Affine Invariance** This is a consequence of the fact that the B-spline basis functions are positive and sum to one.

- **Efficient Algorithms** The evaluation of a tensor product surface can be reduced to the evaluation of the underlying curves. Therefore, the *de Boor algorithm* can also be used for tensor product B-spline surfaces.
• **Refinement** A tensor product B-spline curve can be refined by *knot insertion*. However, in contrast to B-spline curves, knot insertion is not a local operation for a tensor product B-spline surface. For instance, if a new knot \( \hat{u} \) is inserted, with \( u_k < \hat{u} < u_{k+1}, \) then all patches \([u_k, u_{k+1}] \times [v_{d+j-1}, v_{d+j}]\), \(0 \leq j \leq n\) are refined.

• **Automatic Continuity** Just like B-spline curves, a tensor product B-spline surface is \(C^{d-1}\) continuous if the knots \( u_i \) and \( v_j \) are *simple* (do not coincide). The problem of the existence of irregular points for certain degenerated configurations of the control points also occurs for surfaces.

Just like B-spline curves, tensor product B-spline surfaces have a number of attractive properties for variational modeling. Their (control point) representation is simple, there exists an efficient algorithm for the evaluation of points and derivatives and they have automatic \(C^{d-1}\) continuity. However, tensor product B-spline surfaces also have some disadvantages: there is no local refinement algorithm and they can only be used if the surface that has to be modeled can be associated with a rectangular domain.

### 4.5.2 Triangular Bézier Surfaces

Triangular Bézier surfaces are a generalization of composite Bézier curves (see section 3.5.1) to surfaces with triangular patches. The patches of triangular Bézier surfaces are called Bézier triangles. For their definition barycentric coordinates are used.

**Barycentric coordinates**

Let \( V = \triangle (v_0, v_1, v_2) \) be a non degenerate triangle in \( \mathbb{R}^2 \). Then any point \( \mathbf{x} \in \mathbb{R}^2 \) can be uniquely written as

\[
\mathbf{x} = \lambda_0 \mathbf{v}_0 + \lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2,
\]

such that \( \lambda_0 + \lambda_1 + \lambda_2 = 1 \). The values \( \lambda_0 \), \( \lambda_1 \) and \( \lambda_2 \) are called the *barycentric coordinates* of \( \mathbf{x} \) with respect to \( V \). They are explicitly given by

\[
\lambda_i = \lambda_i (\mathbf{u} | V) = \frac{d_i (\mathbf{u} | V)}{\det (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)}, \tag{4.22}
\]

where \( \det (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2) \) is the so called *barycentric determinant* of the triple \((\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)\)

\[
\det (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2) = \begin{vmatrix}
1 & 1 & 1 \\
v_{0x} & v_{1x} & v_{2x} \\
v_{0y} & v_{1y} & v_{2y}
\end{vmatrix}
\]

and \( d_i (\mathbf{u} | V) \) is the barycentric determinant of the triple obtained by replacing the \( i \)-th element of \((\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)\) by \( \mathbf{u} \).
4.5 SURFACE REPRESENTATIONS

Bézier triangles

The barycentric coordinates are used to define a degree $n$ polynomial surface patch over the triangle $V = \triangle (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ as follows:

$$\mathbf{F}(\mathbf{u}) = \sum_{i+j+k=n, i,j,k \geq 0} B_{ijk}^n(\lambda_i, \lambda_j, \lambda_k) \mathbf{P}_{ijk},$$  \hspace{1cm} (4.23)

where $\lambda_i = \lambda_i(\mathbf{u}|V)$, $i = 0 \ldots 2$ are the barycentric coordinates of $\mathbf{u}$ with respect to $V$ and where the functions $B_{ijk}^n$ are the degree $n$ bivariate Bernstein polynomials (see [Far93]), given by

$$B_{ijk}^n(u, v, w) = \frac{n!}{i!j!k!} u^i v^j w^k, \quad u, v, w \geq 0, \quad u + v + w = 1.$$  

Such a patch is called a Bézier triangle. A Bézier triangle has $\binom{n+2}{2}$ control points $\mathbf{P}_{ijk}$, which can be arranged in a control polyhedron with a regular structure (see figure 4.6).

![Image of Bézier patch and control polyhedron](image)

Figure 4.6: Left: a Bézier patch and its control polyhedron. Right: the numbering scheme of the control points.

The boundary of a Bézier patch consists of three univariate Bézier curves. For example, the boundary curve corresponding to the domain edge $\mathbf{v}_0\mathbf{v}_1$ can be represented as

$$\mathbf{x}(t) = \mathbf{F}((1-t)\mathbf{v}_0 + t\mathbf{v}_1) = \sum_{i+j+k=n, i,j,k \geq 0} B_{ijk}^n(1-t, t, 0) \mathbf{P}_{ijk} = \sum_{i=0}^{n} B_{i,0}^n(t) \mathbf{P}_{i,(n-i),0},$$

which is a Bézier curve as defined in section 3.5.1. Note that the boundary curves only depend on the boundary control points of the control polyhedron.

A Bézier patch has the following properties (most of them are explained in section 3.1.7)

- **Affine Invariance** This is a consequence of the fact that the Bernstein polynomials are positive and sum to one.
CHAPTER 4 SURFACES

- **Efficient Algorithms** A generalization of the *de Casteljau algorithm* for Bézier curves can be used for the computation of points and derivatives of a Bézier triangle (see [Far93]).

- **Refinement** Just like for Bézier curves, there are two common ways to refine a Bézier triangle. With *subdivision* the Bézier triangle is split into two or more Bézier triangles, each of them defined on a subtriangle of the domain. With *degree elevation*, the degree of the Bézier triangle is raised. Formulas for the new control points of the Bézier triangle after subdivision or degree elevation can be found in [Far93].

**Composite triangular Bézier surfaces**

In a composite triangular Bézier surface, several Bézier patches are linked together to form a piecewise polynomial surface. The domain of a composite Bézier surface is therefore a triangulation in \( \mathbb{R}^2 \). Given the fact that the boundary of a Bézier triangle is determined by the boundary of the corresponding control polygon, two neighboring Bézier patches have to share the control points of the common boundary. Besides the properties already mentioned for a single Bézier patch, a composite Bézier surface has local support:

- **Local support** When a composite Bézier surface is written in the canonical representation (4.4), the basis functions \( N_i \) are nonzero on only one domain triangle, or two or more neighboring domain triangles if the corresponding control point is shared with neighboring Bézier triangles.

Just like composite Bézier curves, composite Bézier surfaces don’t have automatic continuity:

- **Automatic Continuity** Composite Bézier surfaces only have automatic \( C^0 \) continuity, since the basis functions \( N_i \) are only \( C^0 \) continuous across the edges of the domain triangles. This means that continuity constraints have to be applied to enforce a higher order of continuity between neighboring triangles.

We can conclude that composite Bézier surfaces have a number of properties that are attractive for variational modeling. Their (control point) representation is simple, there exists an efficient algorithm for the evaluation of points and derivatives and they can be locally refined. However, the lack of automatic continuity can be a problem. If the number of triangles is large, the number of continuity constraints is also large. This has a negative effect on the minimization problem.

**4.5.3 Triangular B-spline surfaces**

In the previous section it was shown that composite Bézier curves can be generalized to triangular Bézier surfaces by using bivariate Bernstein polynomials. It has taken quite a while before the same kind of generalization was made from B-spline curves to triangular
4.5 SURFACE REPRESENTATIONS

B-spline surfaces. Recently Dahmen, Micchelli and Seidel have defined triangular B-spline surfaces by using bivariate simplex splines as basis functions (see [DMS92] and [Pfe95]).

Simplex splines are piecewise polynomial functions. Like B-spline basis functions they depend on a number of knots. If these knots are ‘in general position’ (a 2D generalization of non coinciding knots), the corresponding triangular B-spline surface is $C^{n-1}$ continuous, where $n$ is the total degree of the polynomial.

Before the definition of triangular B-spline surfaces is given, simplex splines will be defined. Since simplex splines require the notion of ‘half-open intervals’ in $\mathbb{R}^2$, the half-open convex hull has to be introduced first. The evaluation of triangular B-splines is far from trivial. Therefore an efficient algorithm for the evaluation will be given.

**Half-open convex hull**

The support of a bivariate simplex spline is the convex hull of its knots. To attach each point to exactly one triangle, we need the notion of half-open convex hull. A point $(x,y) \in \mathbb{R}^2$ is in the **half-open convex hull** of a triangle $V$ in $\mathbb{R}^2$ if there exist $\delta > 0$, $\varepsilon > 0$ such that the triangle $\Delta ((x,y),(x+\delta,y),(x+\delta,y+\varepsilon))$ is contained in $V$. The half-open convex hull of $V$ is denoted as $[V]$ and the half-open convex hull of a triangle $\Delta (t_0,t_1,t_2)$ is denoted as $[t_0,t_1,t_2)$. Figure 4.7 gives an illustration of the half-open convex hull of some triangles.

![Figure 4.7: A point $(x,y)$ belongs to the half-open convex hull if a triangle with the same orientation as the gray one lies inside the triangle. In the rightmost three triangles the solid edges and the solid vertices belong to the half-open convex hull, whereas the dotted edges and the open vertices do not.](image)

**Simplex splines**

A bivariate **simplex spline** is a piecewise polynomial, defined in terms of a number of parameter points in $\mathbb{R}^2$ called knots. Simplex splines can be defined as projections of simplices (see [DMS92] or [Tra89]), but we will use the recursive definition given in [Tra89] and [Pfe95]. A simplex spline defined on $n+3$ knots is a piecewise polynomial of degree $n$. If the knots are in general position, i.e. no three knots are collinear, the simplex spline is $C^{n-1}$ continuous.

The support of a simplex spline is the (half-open) convex hull of its knots. Figure 4.8 shows an example of a simplex spline defined on 5 knots.
Figure 4.8: Left: a degree two simplex spline. Right: the corresponding domain (which contains 5 knots).

A constant simplex spline \( (n = 0) \) is defined on three knots. The value of the constant simplex spline with knotset \( V = \{v_0, v_1, v_2\} \) in the point \( u \in \mathbb{R}^2 \) is defined as

\[
M(u|V) = \begin{cases} 
\frac{1}{|\text{det}(V)|} & \text{if } u \in [V) \\
0 & \text{otherwise.}
\end{cases}
\] (4.24)

It is essential that the support of the constant simplex spline is the half-open convex hull of the triangle \( V \), otherwise discontinuities will occur for higher order simplex splines.

A degree \( n \) simplex spline \( (n > 0) \) is defined recursively as a weighted sum of three degree \( n - 1 \) simplex splines. Given a knotset \( V = \{v_0, \ldots, v_{n+2}\} \), we arbitrarily select a set of three points \( W = \{w_0, w_1, w_2\} \) from \( V \), such that the points of \( W \) form a non-degenerate triangle. Then

\[
M(u|V) = \sum_{j=0}^{2} \lambda_j(u|W) M(u|V \setminus \{w_j\}),
\] (4.25)

where \( \lambda_j(u|W), j = 0 \ldots 2 \) are the barycentric coordinates of the point \( u \) with respect to the triangle \( W \). This relation is a 2D generalization of the de Boor recursion relations (3.27) for B-spline basis functions. Although it is not immediately clear, the value \( M(u|V) \) is independent of the choice of the triangle \( W \). A proof of this is given in [DMS92].

When the recursion relation (4.25) is fully expanded, a large number of constant simplex splines remains, all weighted by a polynomial in \( u \) that is a product of barycentric coordinates of \( u \) with respect to a number of triangles. This can be denoted as

\[
M(u|V) = \sum_k p_k(u) q_k(u),
\] (4.26)
4.5 SURFACE REPRESENTATIONS

with \( p_k \) a polynomial that is a product of \( n \) barycentric coordinates \( \lambda_j(u|W) \) and \( q_k \) a constant simplex spline. The domain triangles of the constant simplex splines \( q_k \) partition the domain of the simplex spline into a large number of convex polygons. This is illustrated in the right part of figure 4.8. Note that these polygons are not necessarily triangles.

The \textit{directional derivative} of a degree \( n \) simplex spline (\( n > 0 \)) in the parametric direction \( d \) is given by (see [Tra89])

\[
D_d M(u|V) = n \sum_{j=0}^{2} \mu_j(d|W) M(u|V \setminus \{w_j\}),
\]

where the \( \mu_j(d|W) \), \( j = 0 \ldots 2 \) are coefficients satisfying

\[
d = \sum_{j=0}^{2} \mu_j(d|W) w_j, \quad \sum_{j=0}^{2} \mu_j(d|W) = 0.
\]

\textbf{Triangular B-spline Surfaces}

A triangular B-spline surface is a piecewise polynomial surface with simplex splines as basis functions. The domain of a triangular B-spline surface is a triangulation \( \mathcal{T} \) in \( \mathbb{R}^2 \). To each vertex \( \mathbf{v}_i \) of the triangulation an ordered set of \( n + 1 \) knots \( \{\mathbf{v}_{i,0}, \ldots, \mathbf{v}_{i,n}\} \) is attached, where \( n \) is the degree of the surface (see figure 4.9). This knotset is called the with vertex \( \mathbf{v}_i \)

![Figure 4.9: Domain of a degree 3 triangular B-spline surface. To each vertex corresponds a knotcloud containing 4 knots.](image)

associated \textit{knotcloud}. The first element of the knotcloud is always the vertex itself \( (\mathbf{v}_{i,0} = \mathbf{v}_i) \). The basis functions of a triangular B-spline surface are formed by defining simplex splines on the knots of these knotclouds. For any triangle \( I = \triangle(\mathbf{v}_{i_0, \mathbf{v}_i, \mathbf{v}_j}) \) of \( \mathcal{T} \), we define the ordered set

\[
V_{\beta}^I = \{\mathbf{v}_{i_0,0}, \ldots, \mathbf{v}_{i_0,\beta_0}, \mathbf{v}_{i_1,0}, \ldots, \mathbf{v}_{i_1,\beta_1}, \mathbf{v}_{i_2,0}, \ldots, \mathbf{v}_{i_2,\beta_2}\},
\]
where $|\beta| = \beta_0 + \beta_1 + \beta_2 = n$, $\beta_0, \beta_1, \beta_2 \geq 0$, which yields $(n + 1)(n + 2)/2$ simplex splines $M(\cdot|V_{\beta}^I)$. These simplex splines do not sum to one. They can be normalized to one by multiplying them with the factor $\left|\det(\tilde{V}_{\beta}^I)\right|$, where

$$\tilde{V}_{\beta}^I = \{v_{i_0, \beta_0}, v_{i_1, \beta_1}, v_{i_2, \beta_2}\}.$$

This leads to the so called normalized B-splines

$$N_{\beta}^I(u) = \left|\det(\tilde{V}_{\beta}^I)\right| M(u|V_{\beta}^I).$$

A triangular B-spline surface is then formed as a linear combination of the basis functions

$$\mathbf{F}(u) = \sum_{I \in \mathcal{I}} \sum_{|\beta|=n} N_{\beta}^I(u) \mathbf{P}_{\beta}^I,$$

where the coefficients $\mathbf{P}_{\beta}^I$ are the control points.

A patch of a triangular B-spline surface is the image of a domain triangle $I$. The control points $\mathbf{P}_{\beta}^I$, $|\beta| = n$ of the patch can be arranged in the same regular structure as the control points of a triangular Bézier patch (see figure 4.6). A triangular B-spline surface is automatically continuous over the patch boundaries. It is therefore not necessary that control points are shared between neighboring patches, as was the case for composite triangular Bézier surfaces. However, sharing control points has some advantages:

- The number of control points is reduced.
- The evaluation of points and derivatives can be accelerated due to a result presented in [DMS92]. This result states that in the case of shared control points, a degree $n$ triangular B-spline surface can be expressed as a degree $(n-1)$ triangular B-spline surface with ‘virtual’ control points $\tilde{\mathbf{P}}_{\beta}^I$:

$$\sum_{I \in \mathcal{I}} \sum_{|\beta|=n} N_{\beta}^I(u) \mathbf{P}_{\beta}^I = \sum_{I \in \mathcal{I}} \sum_{|\beta|=n-1} N_{\beta}^I(u) \tilde{\mathbf{P}}_{\beta}^I,$$

with

$$\tilde{\mathbf{P}}_{\beta}^I = \lambda_0(u|\tilde{V}_{\beta}^I) \mathbf{P}_{\beta+(1,0,0)}^I + \lambda_1(u|\tilde{V}_{\beta}^I) \mathbf{P}_{\beta+(0,0,1)}^I + \lambda_2(u|\tilde{V}_{\beta}^I) \mathbf{P}_{\beta+(0,0,1)}^I.$$

Triangular B-spline surfaces have the following properties:

- **Affine Invariance** To guarantee affine invariance, certain restrictions have to be put on the knot clouds. These restrictions are given in [Fra95] and [Pfe95]. Under these restrictions, the basis functions $N_{\beta}^I$ are positive and sum to one over the entire domain, so then a triangular B-spline surface also satisfies the convex hull property.

- **Efficient Algorithms** A coefficient-based algorithm for the computation of triangular B-spline surfaces is unknown and probably doesn’t exist (see [Pfe95]). That is, no algorithm is known that can be applied directly to the triangular B-spline coefficients. All algorithms require at least the partial evaluation of the individual basis functions (i.e. the simplex splines).
4.5 SURFACE REPRESENTATIONS

- **Refinement** Triangular B-spline surfaces can be refined using knotcloud insertion. An algorithm for this is given in [SV94].

- **Local support** The basis functions $N_{i,j}^k$ of a triangular B-spline surface are normalized simplex splines. The support of these simplex splines is the convex hull of their knots. Due to the earlier mentioned restrictions on the knot clouds, the support of the functions $N_{i,j}^k$ is local. Note that when the surface has shared control points, the basis function corresponding to a shared control point is composed of two or more functions $N_{i,j}^k$.

- **Automatic Continuity** A triangular B-spline surface is $C^{n-1}$ continuous if the knots of the basis functions $N_{i,j}^k$ are in general position. Like B-spline curves and tensor product B-spline surfaces, a triangular B-spline surface may have irregular points for certain degenerated configurations of the control points.

**Evaluating triangular B-spline surfaces**

A problem with the evaluation of triangular B-spline surfaces is that there is no coefficient based evaluation algorithm (see [Pfl95]). As a consequence, the recursion relation (4.25) has to be at least partially expanded.

A triangular B-spline surface is in fact a complicated expression in barycentric determinants (see section 4.5.2). This can be seen as follows. The evaluation of a constant simplex spline (4.24) is an ‘inside triangle’ test, which is equivalent to a sign test of some barycentric determinants. A degree $n$ simplex spline (4.25), ($n > 0$), is a weighted sum of three lower degree simplex splines, where the weight factors are barycentric determinants, divided by a constant. A triangular B-spline surface, which is a linear combination of simplex splines, can therefore be expressed in barycentric determinants.

Suppose that we want to evaluate the degree $n$ simplex spline $M(\cdot | V_{\beta}^j)$. Then, at level 0 of the recursion there is 1 degree $n$ simplex spline (namely $M(\cdot | V_{\beta}^j)$ itself). At level 1 of the recursion there are 3 degree $n-1$ simplex splines, and, finally, at level $n$ there are $3^n$ degree 0 simplex splines. Depending on the choice of the sets $W$ in the recursion, more or fewer of these simplex splines will coincide. We can represent the recursion in a graph as follows (see figure 4.10).

In an efficient evaluation algorithm, the number of unique nodes in this graph should be as low as possible. When a triangular B-spline surface is evaluated in a point $u$, the simplex spline $M(\cdot | V_{\beta}^j)$ is not only evaluated for one choice of $\beta$, but for all $|\beta| = n$. In [Fra95] a selection procedure for the sets $W$ in the recursion relation (4.25) is given that leads to the minimal number of nodes of the evaluation graphs of the complete set of simplex splines $\{M(\cdot | V_{\beta}^j) | |\beta| = n\}$, for one triangle $I$. In this way, the maximum reuse of partial results is guaranteed.

Now if we combine this selection scheme with a data structure to store the partial results, and if we make sure that the barycentric coordinates are tabulated (to avoid computing them more than once), an efficient algorithm for the evaluation of arbitrary degree triangular B-splines is obtained.
Figure 4.10: The evaluation graph of a degree 2 simplex spline. For clarity, the value $M(u|\{v_i,v_j,v_k\})$ is abbreviated as $M_{ijk}$. Note that the sets $W$ in the recursion relation are chosen such that three of the constant simplex spline nodes (at level 2) are shared.

Due to the similarity between the equations 4.25 and 4.27, derivatives of simplex splines can be simultaneously computed with almost no overhead.
4.6 Implementation and results

The way of variational modeling described in this dissertation (with data dependent fairness functionals and external energy operators) has been implemented for the three surface representations discussed in section 4.5: tensor product B-splines, Bézier triangles and triangular B-splines. This section explains how the constraints and internal and external energy expressions can be computed and how the energy minimization problem is solved for surfaces. Furthermore a short description will be given of an interactive program that has been developed for variational modeling with triangular B-splines. Finally, a number of experiments with variational surface modeling will be described.

4.6.1 Computation

In the implementations, the triangular Bézier surfaces and the tensor product B-spline surfaces are restricted to the cubic case (degree 3). On the other hand, the triangular B-spline surfaces may have arbitrary degree.

Constraints

The implementation is restricted to the linear constraints described in section 4.2. It will be explained how the constraint equations can be computed and how they are expressed in terms of the concatenation vector \( \mathbf{P} \) of the control points, given by (4.5).

From the constraint representations given in section 4.2 it follows that the point to point/line/plane constraint and the normal to vector/plane constraint can all be written as one or more equations of the form

\[
\mathbf{d}_k^T \mathbf{P} = \epsilon_k,
\]

for some vector \( \mathbf{d}_k \in \mathbb{R}^{3n} \) and some constant \( \epsilon_k \in \mathbb{R} \). The components of \( \mathbf{d}_k \) and \( \epsilon_k \) are expressions in the basis functions \( N_i \) and first and second partial derivatives of \( N_i \), evaluated at the parameter value \( (u_0, v_0) \) of the point to which the constraint is applied. These values can be easily computed.

The curve interpolation constraints are limited to some special cases. For tensor product B-spline surfaces only curves of the form \( u = \text{const}, \, v_i \leq v \leq v_j \) and \( v = \text{const}, \, u_k \leq u \leq u_l \) are considered, where \( u_i \) and \( v_j \) are knot values of the surface. These constraints are sufficient to fix the boundary of a tensor product B-spline surface, but also curves in the interior. More information on how to interpolate a curve in the interior of a tensor product B-spline surface can be found in [Kal93]. These curve constraints are linear in the control points \( \mathbf{P}_i \) (see section 4.5.1), hence they can be written in the form (4.29). For triangular Bézier surfaces only patch boundaries are used for curve interpolation. Again, this suffices to interpolate the boundary of the surface as well as interior curves. For interior curves, possibly some patches may have to be split (see [VWvdW93]). Since a boundary curve of a Bézier patch is a Bézier curve with the boundary of the control polyhedron as its control points (see section 4.5.2), such a constraint can be enforced by giving these control points fixed values. Therefore, the curve interpolation constraint for triangular Bézier surfaces is linear in the
control points and can be written in the form (4.29). The interpolation of the boundary of a triangular B-spline surface is a non-trivial task, since the boundary curves don’t have an elegant form like the boundaries of tensor product B-spline surfaces or Bezier surfaces. Due to time constraints no attempts have been made to solve this problem.

For triangular Bezier surfaces a restricted form of $C^1$ continuity constraints has been implemented. Two different cases have to be distinguished: continuity relations along an edge that is shared between two patches and continuity relations around a corner that is shared by two or more patches. The relations along a shared edge are implemented as linear relations with fixed coefficients in the control points $P_0, P_1, P_2$ and $P_3$ (see figure 4.11).

![Figure 4.11: Continuity relations between Bezier triangles.](image)

The relation around a vertex (with corresponding control point $R$) has been implemented as a linear relation in the control points $Q_0, Q_1, \ldots, Q_n$ and $R$, with coefficients depending on the number of patches that are joined at this vertex. Naturally, this gives a restricted form of tangent plane continuity. It would be better to let the values of the coefficients depend on the shape of the surface after energy minimization, but to do the computation, the coefficients be given a value beforehand. Since these constraints are linear, they can also be written into the form (4.29). For tensor product and for triangular B-spline surfaces continuity constraints are not necessary, since the basis functions $N_i$ already have the desired order of continuity.

From the above overview of constraints it follows that (analogous to curves), the constraints for surfaces can be combined into a linear system of equations

$$DP = e,$$

with $D \in \mathbb{R}^{m \times 3n}$, where $n$ is the number of control points of the surface. Since each constraint equation only depends on a small number of control points, the matrix $D$ is sparse.

**Internal energy**

For all three surface representations the internal energy of the surface is a linear combination of the bend and stretch energy, as described in section 4.3. The simple approximations of
the bend and stretch energy \( E_{\text{bend, simple}} \) and \( E_{\text{stretch, simple}} \) as well as the data dependent approximations \( E_{\text{bend, data}} \) and \( E_{\text{stretch, data}} \) have been implemented.

From the expressions for the bend and stretch energy derived in section 4.3, it follows that the internal energy \( E_{\text{int}} \) of a surface \( \mathbf{F} \) with canonical representation (4.4) can be written as a quadratic expression in the control points \( \mathbf{P}_i \):

\[
E_{\text{int}}(\mathbf{F}) = \sum_{i,j} a_{ij} \mathbf{P}_i \cdot \mathbf{P}_j.
\]

The corresponding matrix \([a_{ij}]\) will be denoted as \( A_{\text{int}} \). Just like the internal energy of a B-spline curve is composed of the internal energies of its segments, the internal energy of the surfaces considered here is composed of the internal energies of their patches. A difference is that for the surface case the indices \( \{i\} \) and \( \{j\} \) cannot be permuted such that the matrix \( A_{\text{int}} \) has a small bandwidth.

For a single patch that depends on the control points \( \{\mathbf{P}_{i_p}\}_{p=1}^s \), the internal energy can be written as

\[
\sum_{p=1}^s \sum_{q=1}^s a_{pq} \mathbf{P}_{i_p} \cdot \mathbf{P}_{i_q},
\]

where \( \{i_p\} \) is the set of indices on which the patch depends. The coefficients \( a_{pq} \) are given by the relations (4.14), (4.16), (4.19) and (4.21), with \( \Omega \) the domain of this patch. For the simple approximations, these coefficients can be computed exactly, whereas the coefficients for the data dependent approximations have to be computed using numerical integration.

**Simple approximations** A cubic tensor product B-spline patch depends on 16 control points \( (s = 16) \). Due to the uniform knot spacing the 256 coefficients \( a_{pq} \) are for each patch the same and have to be computed only once.

A cubic Bézier triangle depends on 10 control points \( (s = 10) \). Since the domain patches for the patches are not the same, the coefficients \( a_{pq} \) are not the same for each patch. However, they can be expressed as linear combinations of the same set of integrals. Consider a Bézier triangle with representation (4.23). If we interpret the Bernstein polynomials \( B_{ijk}^n \) in this representation as functions of two variables by putting \( B_{ijk}^n(u,v) = B_{ijk}^n(u,v,1-u-v) \), the basis functions can be written as \( B_{ijk}^n \circ \lambda \), with \( \lambda : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) an affine mapping, defined by the barycentric coordinate functions \( \lambda_i \) in (4.23). The domain of the triangle is then given by \( \lambda^{-1}(T_2) \), with \( T_2 \) the 2-simplex, which is defined as \( \{(x,y) \in \mathbb{R}^2 | x \geq 0, y \geq 0, x+y \leq 1\} \). The coefficients \( a_{pq} \) in (4.31) have therefore one of the following forms:

\[
\int_{\lambda^{-1}(T_2)} \frac{\partial (B_p \circ \lambda)}{\partial x} \frac{\partial (B_q \circ \lambda)}{\partial x} + \frac{\partial (B_p \circ \lambda)}{\partial y} \frac{\partial (B_q \circ \lambda)}{\partial y} \, dx \, dy,
\]

\[
\int_{\lambda^{-1}(T_2)} \frac{\partial^2 (B_p \circ \lambda)}{\partial x^2} \frac{\partial (B_q \circ \lambda)}{\partial x} + \frac{\partial^2 (B_p \circ \lambda)}{\partial y^2} \frac{\partial (B_q \circ \lambda)}{\partial y} \, dx \, dy,
\]

where \( B_p \) and \( B_q \) are cubic Bernstein polynomials, with \( p \) and \( q \) index triples. By applying the affine parameter transformation \((u,v) = \lambda(x,y)\), these integrals can be transformed.
to integrals over the 2-simplex $T_2$. Using the chain rule, one can easily prove that these transformed integrals are linear combinations of the following integrals:

\[
\begin{align*}
&\int_{T_2} B_{pu} B_{qu} \, du \, dv, \quad \int_{T_2} B_{pu} B_{qv} \, du \, dv, \quad \int_{T_2} B_{pu} B_{qv} \, du \, dv, \\
&\int_{T_2} B_{puu} B_{qu} \, du \, dv, \quad \int_{T_2} B_{puu} B_{qv} \, du \, dv, \quad \int_{T_2} B_{puu} B_{qvv} \, du \, dv, \\
&\int_{T_2} B_{puu} B_{qvv} \, du \, dv, \quad \int_{T_2} B_{pqu} B_{qu} \, du \, dv, \quad \int_{T_2} B_{pqu} B_{qv} \, du \, dv.
\end{align*}
\]

It is therefore sufficient to compute these integrals only once for all triples $p$ and $q$ with $|p| = |q| = 3$.

The number of control points on which a triangular B-spline patch depends is variable. The reason for this is that the basis functions of a neighbor patch may overlap with the domain of the patch. Unfortunately, for triangular B-splines the coefficients $a_{pq}$ of the simple approximations can't be computed as efficient as for tensor product B-spline surfaces or triangular Bézier surfaces. The main reason is that triangular B-splines are not polynomial inside a patch, but piecewise polynomial instead. Splitting the domain triangles of the patches into polygons that correspond to polynomial pieces of the surface would require a complex administration. Therefore another approach is followed. In section 4.6 it was shown that a basis function $N_i$ can be represented as a sum of products of polynomials with constant simplex splines (4.26), or in other words, polynomials with the half-open convex hull of a triangle as their domain. These polynomials are all explicitly computed and stored, together with the corresponding constant simplex splines. For efficiency reasons, polynomials corresponding to the same constant simplex spline are grouped together. Partial derivatives of a basis function can be written in the same representation, with the only difference that the polynomials $p_k$ in (4.26) have a lower degree. In [Fra95], routines are described to compute an integral of a product of two of these polynomials over an arbitrary convex polygon. This suffices to compute the coefficients $a_{pq}$. Since this approach is very elaborate, it is probably more efficient to compute the coefficients using numerical integration (see below).

**Data dependent approximations** For the data dependent approximations, the coefficients $a_{pq}$ of a single patch are computed with the numerical integration routines from section 2.5.5. For each patch, the integrands in (4.16) and (4.21) are computed in a fixed number of points. To do this, the first and second partial derivatives of the basis functions $N_i$ need to be computed in these points. For uniform tensor product B-spline surfaces these values need to be computed only once, since they are the same for each patch. For triangular Bézier surfaces it also suffices to compute these values for only one patch (say with the 2-simplex $T_2$ as its domain), since the corresponding values of a patch with an arbitrary domain triangle can be expressed as linear combinations of them. Finally, for triangular B-splines there is no other way than computing these values for all patches, since the configuration of the knots may be different for all patches.
4.6 IMPLEMENTATION AND RESULTS

External energy

For the external energy, the operators described in section 3.4 can be chosen: the point to point/line/plane attractors, the curve/region to point/line/plane/region attractors and the normal to vector/plane attractors.

These external energies can also be expressed in the concatenation vector $P$ (4.5). From the external energy representations given in section 4.4 it follows that they can all be written in the form

$$E_{\text{attractor}}(F) = P^T A_{\text{attractor}} P + B_{\text{attractor}}^T P + C_{\text{attractor}},$$

(4.33)

for some matrix $A_{\text{attractor}} \in \mathbb{R}^{3n \times 3n}$, some vector $B_{\text{attractor}} \in \mathbb{R}^{3n}$ and some constant $C_{\text{attractor}} \in \mathbb{R}$. The energy contributions of the corresponding repellors have the following form:

$$E_{\text{repellor}}(F) = \frac{1}{P^T A_{\text{attractor}} P + B_{\text{attractor}}^T P + C_{\text{attractor}}}.$$  

(4.34)

For operators that apply to a single point with parameter $(u_0, v_0)$, the components of $A_{\text{attractor}}, B_{\text{attractor}}$ and $C_{\text{attractor}}$ are expressions in the values $N'_i(u_0, v_0)$ and its first and second partial derivatives.

For operators that apply to a region $\Gamma$, the components of $A_{\text{attractor}}, B_{\text{attractor}}$ and $C_{\text{attractor}}$ can be expressed in the following integrals:

$$\int_{\Gamma} N_i N_j \, du dv,$$

$$\int_{\Gamma} N_i u N_j \, du dv,$$

$$\int_{\Gamma} N_i v N_j \, du dv,$$

$$\int_{\Gamma} N_i N_j \, du dv,$$

(4.35)

for $0 \leq i, j < n$. For all three surface types, the region $\Gamma$ is restricted to arbitrary convex polygons. For tensor product B-spline surfaces and triangular Bézier surfaces, these polygons are split into trapezoids of the form:

$$\{(u, v) \in \mathbb{R}^2|a \leq u \leq b \land a_0 + a_1 u \leq v \leq b_0 + b_1 u\},$$

(4.36)

with the extra condition that the region must lie within the domain of one patch (see figure 4.12). This extra condition is needed to ensure that the integrands in (4.35) are polynomial expressions on the trapezoid.

For tensor product B-spline surfaces, the integrals (4.35) can be reduced to one-dimensional integrals. For instance, for a domain $R$ of the form (4.36):

$$\int_R N^3_{ij}(u, v) N^3_{kl}(u, v) \, du dv = \int_R N^3_i(u) N^3_j(v) N^3_k(u) N^3_l(v) \, du dv$$

$$= \int_a^b N^3_i(u) N^3_k(u) \left[ \int_{a_0 + a_1 u}^{b_0 + b_1 u} N^3_j(v) N^3_l(v) \, dv \right] \, du,$$

where $N^3_i$ and $N^3_{ij}$ are cubic one and two-dimensional B-spline functions. Now suppose the region $R$ lies inside the patch with domain $[p, p + 1] \times [q, q + 1]$. Using the translation
Figure 4.12: For the integration, a convex polygon is split into ‘vertical’ trapezoids.

\((x, y) \mapsto (x + 2 - p, y + 2 - q)\), we can transform the integral over the region \(R\) into an integral over \(R' \subset [2,3] \times [2,3]\). Due to the property (3.28) of uniform B-spline curves, the resulting integral is an expression in the polynomials \(N_i^3(t)\), \(0 \leq i \leq 3\), \(t \in [2,3]\). It is therefore sufficient to only store these polynomials.

For triangular Bézier patches, the same parameter transformation \((u, v) = \lambda(x, y)\) is applied to the integrals (4.35) as to the coefficients of the internal energy (4.32). The transformed integrals are linear combinations of the following integrals, with \(\Gamma' = \lambda(\Gamma)\) as the integration domain:

\[
\int_{\Gamma'} B_p B_q du dv, \int_{\Gamma'} B_{pu} B_{qv} du dv, \int_{\Gamma'} B_{pu} B_{qv} du dv, \int_{\Gamma'} B_{pu} B_{qv} du dv,
\]

with \(B_p\) and \(B_q\) cubic Bernstein polynomials. This domain \(\Gamma'\) can be split into regions \(R\) of the form (4.36). The Bernstein polynomials are stored as powerbase polynomials in \(u\) and \(v\). Routines are written to multiply these polynomials and their partial derivatives and to integrate them over a standard domain of the form (4.36). This suffices to compute the coefficients for triangular Bézier surfaces.

For triangular B-spline surfaces, the integrals (4.35) are computed using numerical integration. The integration points are found by splitting the domain \(\Gamma\) into triangles and then determining integration points for these triangles.

### 4.6.2 Solving the minimization problem

The energy minimization of a surface is a nonlinear programming problem (see section 2.5.3). In this section the problem will be formulated in a general form, and the solution methods that have been implemented will be discussed.

In section 4.6.1 it was shown how the constraints on a surface that have been considered can be written as a linear system of equations (4.30) in the concatenation vector \(\mathbf{P}\) of the control points. The goal function of the minimization problem is the sum of the internal and external energies of the surface. If we denote this sum as \(f(\mathbf{P})\), the energy minimization problem can be formulated as

\[
\begin{align*}
\text{minimize} & \quad f(\mathbf{P}) \\
\text{such that} & \quad D\mathbf{P} = \mathbf{e}
\end{align*}
\]

for some \(D \in \mathbb{R}^{m \times 3n}\) and \(\mathbf{e} \in \mathbb{R}^m\).
4.6 IMPLEMENTATION AND RESULTS

Analogous to curves, two different cases can be distinguished. If no repellors are applied, the total energy \( f(P) \) is a quadratic function in the vector \( P \), and can thus be written as \( f(P) = P^TAP + b^TP + c \) for some \( A \in \mathbb{R}^{3n \times 3n} \), \( b \in \mathbb{R}^{3n} \) and \( c \in \mathbb{R} \). The matrices \( A \) and \( D \) are sparse. Especially for surfaces these matrices have to be stored using a sparse representation, since the number of control points \( n \) may be very large. Just like for curves, this quadratic programming problem is solved using a null space method (if \( m \) is small) or a Lagrangian method (if \( m \) is large). See section 3.6.6 for further details.

If repellors are applied, the goal function \( f(P) \) also contains nonlinear terms of the form (4.34). The minimization problem then becomes more difficult to solve. Analogous to curves, this system is solved in two steps. First, the constraints are eliminated by rewriting them into the form \( P = Fy + g \). The remaining unconstrained minimization problem is solved for \( y \) using the nonlinear conjugate gradient method (see section 2.5.2).

4.6.3 Interactive program

For triangular B-splines an interactive program has been developed. The main purpose is to experiment with variational surface modeling with external energy operators. In [vG95] the design of the graphical user interface is thoroughly described. This section gives a short overview of the functionality of the program.

The surface can be viewed as a two-dimensional wire-frame rendering displayed in a window. For this, a parallel projection is used (see section 2.6.3). The direction of projection is always chosen perpendicular to the projection plane. Using the mouse, the user can freely rotate, translate and zoom in or out on the scene. The constraints and the external energy operators applied to the surface are also displayed graphically. To prevent that the screen becomes too cluttered, the user can decide which of the following items are visible: the surface, the control points of the surface, the constraints, the external energy operators, the axes of the coordinate system and some auxiliary lines that connect the constraints and operators with the surface.

The program supports the following user interaction:

- Patches can be added or deleted.
- The internal energy of the surface can be set to the simple approximation \( \alpha E_{\text{stretch,simple}} + (1-\alpha)E_{\text{tend,simple}} \) or the data dependent approximation \( \alpha E_{\text{stretch, data}} + (1-\alpha)E_{\text{tend, data}} \). The weight factor \( \alpha \) can be freely chosen.
- The previously mentioned constraints and external energy operators can be entered graphically.
- The control points, constraints and external energy operators can be edited by direct manipulation using the mouse.
- The minimal energy surface for the current selection of operators and internal energy is computed when requested.
4.6.4 Implementation aspects

To support the computation of minimal energy surfaces and the development of an interactive program, a number of C++ class libraries have been written. In the first place a library for curve and surface representations is developed, containing C++ classes for Bézier and B-spline curves, tensor product B-spline surfaces, triangular Bézier surfaces and triangular B-spline surfaces. These classes support the following:

- Computation of geometric info like evaluation of points, normal vectors, principal curvatures, directional derivatives etc.
- Computation of intersections with straight lines and projections of points (used for the interaction).
- Computation of expressions for the constraints, internal and external energy in terms of control points.
- Saving in different formats (wire-frame renderings in Postscript and colored renderings in VRML).

Other object oriented libraries for curves and surfaces with some of this functionality are described in [RR94] and [Kol95].

In the second place a numerical C++ library is developed for solving quadratic programming problems, linear systems of equations (directly or iteratively), and nonlinear programming problems and for performing numerical integration. For the dense matrix classes, the Meschach-library has been used (see [Stu94]), which is a little bit outdated, but nevertheless very reliable and complete library for matrix computations.

In the third and last place a C++ library containing all kinds of geometric objects like linesegments, lines, circles, triangles, planes, polygons, triangulations etc. This library is primarily used to support the interactive program.

4.6.5 Examples

A number of experiments with variational surface modeling have been done. A few of them will be described below.

Iterative application of data dependent fairness functionals

Just like for curves (see section 3.6.7), the data dependent fairness functionals for surfaces have also been applied iteratively. Given an initial surface $F_0$, a new surface $F_1$ is computed by minimizing a data dependent energy functional with $F_0$ as a reference surface. This surface $F_1$ is used as a reference surface for the computation of $F_2$ etc.

The data dependent stretch energy functional (4.20) was applied iteratively to a surface with a fixed boundary. The result was that the sequence of surfaces $\{F_k\}_{k=0,1,...}$ converges towards a surface with minimal area (a minimal surface), see figure 4.13.
4.6 IMPLEMENTATION AND RESULTS

Figure 4.13: The result of iteratively applying the data dependent stretch energy functional is a surface with minimal area.

The values of the stretch energy (i.e. the area) of the iterands can be found in the left columns of table 4.1.

The data dependent bend energy functional (4.15) was also applied to a surface with a fixed boundary. For the initial surface a part of a cylinder was chosen. The result was that in this case (just like for curves) the method doesn’t seem to converge (see the right columns of table 4.1). In the first two iterations the bend energy decreases significantly, but then (apparently in the neighborhood of a minimum of the bend energy (4.10)) the method becomes unstable and the value of the bend energy is increasing again. Figure 4.13 shows the surface after two iterations.

Figure 4.14: The cylinder like surface after two iterations with the data dependent bend energy functional.
Table 4.1: The results of two experiments with the iterative application of the data dependent approximations for the stretch and bend energy.

**Triangular B-spline surfaces**

The experiences with triangular B-spline surfaces haven't been very successful. For one single patch the results seem all right. The patch in figure 4.16 was once minimized with the simple approximation of the stretch energy (the left picture) and once with the simple approximation of the bend energy (the right picture).

Both results are in agreement with the theory from section 4.3. However, as soon as two or more patches are used, the results are not so good. Figure 4.16 (left) shows a triangular B-spline surface consisting of two patches. This surface was obtained by applying four point to point constraints at the corners and a normal region to vector attractor to a triangle in the center. Although at first sight the result may seem satisfying, a curvature plot (figure 4.16, right) reveals that there are regions of high curvature along the common patch boundary. This problem occurred in most of the experiments that have been done with triangular B-spline surfaces. There are three possible explanations for this phenomenon. In the first place,
4.6 IMPLEMENTATION AND RESULTS

Figure 4.15: Left: a triangular B-spline patch minimized with the simple approximation of the stretch energy. Right: the same patch minimized with the simple approximation of the bend energy.

Figure 4.16: The result of applying a normal region to vector attractor to a triangular B-spline surface consisting of two patches.

the knot placement of the surface can be unattractive. The knot placement has been done on an ad hoc basis. It hasn’t been investigated what the effect of the knot placement is on the behavior of the surface along the patch boundaries. In the second place, along the patch boundaries a triangular B-spline consists of many polynomial pieces on very small domain polygons. This might cause problems for the numerical integration. In the third and last place an error could have been made in the implementation.

Lagrange equations

When the number of control points of a surface is large, and the number of linear constraints is too high to eliminate them directly, the quadratic programming problem associated with variational modeling is solved using the Lagrangian method (see section 2.5.4). For this
/newpage method the linear system (2.39)

\[
\begin{pmatrix}
A + A^T & D \\
D & 0
\end{pmatrix}
\begin{pmatrix}
x \\
\lambda
\end{pmatrix} =
\begin{pmatrix}
-b \\
e
\end{pmatrix}
\]

has to be solved. It is necessary that the energy matrix \( A \) and the constraint matrix \( D \) are stored sparse. This linear system has been solved using stabilized orthogonal directions (STOD) and the conjugate residual method (CR), see section 2.5.1. The experiences with both methods haven’t been favorable; their convergence is rather slow. The following experiment has been done. As an initial surface a flat surface with a rectangular shape in the \( xy \)-plane is chosen. The boundary of this surface is fixed and four random point to point constraints are applied to the surface. The corresponding Lagrange equations are solved using STOD. This has been done for surfaces with varying numbers of control points. The results of this experiment are found in table 4.2.

<table>
<thead>
<tr>
<th></th>
<th>10 × 10</th>
<th>15 × 15</th>
<th>20 × 20</th>
<th>25 × 25</th>
<th>30 × 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>228</td>
<td>212</td>
<td>259</td>
<td>234</td>
<td>249</td>
<td>231</td>
</tr>
<tr>
<td>248</td>
<td>227</td>
<td>242</td>
<td>298</td>
<td>262</td>
<td>235</td>
</tr>
<tr>
<td>326</td>
<td>277</td>
<td>300</td>
<td>323</td>
<td>319</td>
<td>304</td>
</tr>
<tr>
<td>420</td>
<td>359</td>
<td>410</td>
<td>440</td>
<td>433</td>
<td>393</td>
</tr>
<tr>
<td>567</td>
<td>498</td>
<td>551</td>
<td>587</td>
<td>582</td>
<td>548</td>
</tr>
<tr>
<td>234</td>
<td>249</td>
<td>231</td>
<td>254</td>
<td>229</td>
<td>255</td>
</tr>
<tr>
<td>211</td>
<td>262</td>
<td>235</td>
<td>240</td>
<td>244</td>
<td>283</td>
</tr>
<tr>
<td>305</td>
<td>319</td>
<td>304</td>
<td>306</td>
<td>312</td>
<td>312</td>
</tr>
<tr>
<td>407</td>
<td>433</td>
<td>393</td>
<td>396</td>
<td>407</td>
<td>407</td>
</tr>
<tr>
<td>541</td>
<td>582</td>
<td>548</td>
<td>546</td>
<td>556</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: The number of iterations of the STOD-method for surfaces with different numbers of control points. The iteration was stopped when the length of the residual was decreased with a factor of at least 100000.

The number of iterations needed to get a reliable approximate solution appears to be very high. A positive result is that the number of iterations doesn’t increase dramatically when the number of control points is increased. For better results, probably a preconditioner should be applied. Perhaps also another method could be used that exploits the special structure of the Lagrange matrix.
4.6 IMPLEMENTATION AND RESULTS

Modeling examples

In this section a few small examples of variational surface modeling with external energy operators is given. The first figure 4.17 shows a rather simple shape, that was modeled with very few operators. However, it is not obvious how to create such a shape with variational modeling using constraints only.

![Figure 4.17: This simple shape was modeled using four point to point constraints and two region to plane attractors.](image)

Modeling with repellors is a rather unstable way of modeling. If a repellor is combined with another operator, it is often hard to predict what the result will be. Therefore it seems better to combine a repellor with one or more constraints. Just like in the curve case, where a point repellor was combined with a point to line constraint (see figure 3.13), the results of a repellor become predictable when the surface is limited in its movements. Figure 4.18 shows an initial surface to which a region to plane repellor will be applied. The four corner points are fixed and the four points in the middle of each edge are only allowed to slide along the lines. The repellor plane is situated beneath the surface, which makes the surface move upwards. The result of this experiment is depicted in figure 4.19. The difference between the left and right surface is caused by increasing the weight factor corresponding to the repellor.

The last figure 4.20 is another example of how a small number of operators can have a large effect. This shape was generated by five normal region to vector attractors (and the boundary was constrained).
Figure 4.18: An initial surface with four point to point and four point to line constraints

Figure 4.19: These two surfaces were obtained by applying a region to plane attractor to the initial surface in the previous figure. The weight factor was increased for the right surface, which causes a more extreme effect.
Figure 4.20: This surface was deformed using five normal region to vector attractors. The directions corresponding to the attractors are displayed as large arrows.
Chapter 5

Conclusions and further work

5.1 Conclusions

Variational modeling is a rapid evolving and very powerful way of modeling. With little effort a designer can model a pleasant shape by applying some geometric constraints and letting the computer generate a smooth curve or surface with minimal energy that satisfies these constraints. However, modeling with constraints only has its limitations. In this thesis a number of design operators is introduced that can be used to deform a minimal energy surface. These operators are successfully applied in variational modeling applications for a number of curve and surface representations. It has been shown that most of these operators have an intuitive and clearly visual effect. The operators give the designer an enormous extra modeling power: even with a few operators a surface can be drastically deformed.

The introduction of the design operators also causes a potential problem. Most deformation effects can be achieved in numerous different ways, with all kinds of combinations of operators. It is often hard to tell which operators should be selected to achieve a specific goal, and how they should be applied. To find this out a lot of experiments still need to be done. However, the first experiments with this kind of modeling indicate that it will be at least a useful addition to existing methods.

The most often used fairness functional in variational modeling is the thin plate energy functional. However, for interactive modeling this functional is far too complex. Therefore often quadratic approximations to the thin plate energy are used. In general these approximations can be very poor. A promising alternative for these approximations is formed by the so-called data dependent fairness functionals, as they were introduced by Greiner. A part of this dissertation is devoted to data dependent approximations to the surface area (or stretch energy) and the thin plate energy (or bend energy). It has been shown that the data dependent approximation of the surface area can be applied to iteratively compute a minimal surface. The data dependent approximation of the thin plate energy however, can not be used iteratively to compute a minimum of the thin plate energy. It has been shown (for curves) that the iterative method is not stable in the neighborhood of a minimum, and this has been verified experimentally. This doesn’t make the data dependent approximation of the thin plate energy useless: applying this functional once or twice yields much better
results than the current approximations.

An important aspect of variational modeling is the ratio between the speed at which a minimal energy surface is computed and the effectiveness of the modeling method (i.e. how many iterations does it take to create the desired curve or surface?). Most of the experiments that have been done for surfaces took somewhere between a few seconds and a few minutes for surfaces with approximately 100 control points. Since the current implementation can be optimized at some points, it will probably be possible in the near future to model in real time with surfaces of this size. For larger surfaces this still seems a problem. Solving the Lagrange equations iteratively using methods like STOD or CR (see section 2.5.1) doesn’t seem to be the right solution. Perhaps other types of basis functions like wavelets need to be used (see [GC95]).

The question whether or not triangular B-spline surfaces are suitable for variational modeling cannot be answered at this point. First of all an explanation needs to be given for the poor results that have been found. Besides that, more research needs to be done to topics like the knot placement (which currently is a non-trivial task), boundary interpolation and the design of efficient evaluation algorithms.

5.2 Limitations

The modeling with external energy operators as described in this dissertation has its limitations. The most important one is that deformations are always in conflict with the fairness functional, which tries to keep the surface smooth. It is therefore difficult to create surfaces with sharp bending. Of course the weight of an operator may be increased to a very high level to force it to do its job properly. However, this may have unpredictable effects for the rest of the surface, since the operator with a very high weight will dominate the total energy of the surface. Other problems are that it is sometimes hard to predict what will happen if several operators are combined, and that it is sometimes difficult to choose the right weight factors.

5.3 Improvements and further work

The already mentioned difficulties in creating regions with higher curvature may be solved in the following way. The designer often has a clear idea of where the surface should be smooth and where regions with higher bending are allowed or even desired. If the designer could assign these regions, the contribution of the thin plate energy might be lowered at these places (for example by multiplying them with a small weight factor). If this concept is feasible (which has to be verified by experimenting) the modeling with fairness functionals (and particularly the modeling with external energy operators) would be greatly enhanced.

To support the designer in the use of external energy operators, it seems a good idea to combine several operators into new ones with a specific goal. This will make it possible to offer the designer higher level operators with only a few parameters that can be manipulated. An example could be an operator that is used to sharpen a corner of a surface.
5.3 IMPROVEMENTS AND FURTHER WORK

The data dependent fairness functionals have so far only been applied to the thin plate functional and to the surface area. It would be interesting to see if it is possible to define the same kind of data dependent approximations to the fairness functionals from Rando and Roulier (see the sections 3.1 and 4.1). Another question that remains open is if there exist other data dependent fairness functionals for the thin plate energy that can be used in an iterative method.

A last subject of further research is of course the development of a more extended set of operators (not only the quadratic ones described in this dissertation). For example, it might be useful to have an operator that attracts curves or surfaces towards a sphere.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


Samenvatting

Het modelleren van een ruimtelijke kromme of een oppervlak door het minimaliseren van hun energie resulteert in een gladde kromme of oppervlak. Deze manier van modelleren (ook wel ‘variational modeling’ genoemd) is aantrekkelijk, omdat de ontwerper niet wordt lastig gevallen met de precieze representatie van de kromme of het oppervlak.

Het vervormen van zo’n kromme (oppervlak) gebeurt in het algemeen op twee verschillende manieren. De eerste manier is het rechtstreeks manipuleren van de vrijheidsgraden van zo’n kromme of oppervlak. Met name voor oppervlakken is dit een complexe en tijd-droevende bezigheid, vooral doordat het aantal vrijheidsgraden overweldigend groot kan zijn. De tweede manier is het opleggen van een aantal geometrische randvoorwaarden, waarna een minimale energie kromme of oppervlak via een geautomatiseerde procedure kan worden berekend. Aangezien er vervormingen denkbaar zijn die zich niet gemakkelijk laten uitvoeren door middel van constraints is er behoefte aan methoden voor betere controle over de vorm van minimale energie krommen of oppervlakken. In dit proefschrift wordt een mogelijke nieuwe aanpak voorgesteld die bestaat uit operatoren die een bijdrage leveren aan de energie van de kromme (of oppervlak). Deze operatoren hebben in het algemeen een intuïtief defor- matie effect op het oppervlak. Door middel van een gewichtsfactor kan de gebruiker regelen hoe sterk het effect is van een operator.

Een belangrijk aspect van variational modeling is de keuze van de energie functionaal die geminimaliseerd wordt. De veelvuldig toegepaste kwadratische benaderingen van de buigings- en strekkings-energie hebben als nadeel dat de resultaten vaak verre van optimaal zijn. Onlangs is door Greiner een nieuwe klasse van data-afhankelijke kwadratise be- naderingen geïntroduceerd die tot betere resultaten kunnen leiden. In dit proefschrift wordt zowel theoretisch als praktisch onderzoek gedaan naar deze data-afhankelijke functionals. Het is onder andere gebleken dat de data afhankelijke benadering van de buigingsenergie zich niet leent voor het iteratief minimaliseren van de exacte buigingsenergie. Desondanks is deze benadering een nuttig instrument, omdat de kwaliteit van de gevonden oplossingen in het algemeen hoger is dan de oplossingen van de simpele benaderingen die meestal worden gebruikt.

De hierboven genoemde manieren van variational modeling met energie operatoren en data-afhankelijke energie functionals is toegepast op verschillende klassen van krommen en oppervlakken, zoals Bézier driehoeken, DMS-oppervlakken (driehoekige B-splines) en tensor produkt B-spline curves en oppervlakken. Er wordt veel aandacht besteed aan allerlei aspecten die bij de implementatie van variational modeling op deze representaties een rol spelen.
Acknowledgements

There are a number of people that I would like to thank for their support in the past few years.

First of all thanks to Kees van Overveld, Remco Veltkamp, Hennie ter Morsche, Hans Hagen and Dieter Hammer for taking part in the reading committee and for carefully reading the manuscripts of this thesis. I also would like to thank Günther Greiner and his colleagues for making my stay in Erlangen very enjoyable, and for sharing his inspiring ideas on geometric modeling. I would like to thank Jan de Graaf and Arnold Reusken for their help and interest in some mathematical issues, Huub for his invaluable help in overcoming the peculiarities of compilers and linkers, and Edwin (2x), Han-Wen, Jochem, Michael and Paul for their programming efforts.

Then, I would like to thank my colleagues for the ‘magic’ moments and my parents and sister Ina for keeping their patience with me during the last six months. Finally, I especially would like to thank Remco for his cooperation during the first years of this research and for keeping me motivated and Marja for showing interest in matters that are far more important than doing research at a university.
Curriculum Vitae

Wieger Wesselink was born on December 28, 1966 in Zutphen. In 1985, after his pre-university education (VWO) at the Baudartius College in Zutphen, he started to study Applied Mathematics at the University of Twente. In 1991 he wrote a thesis on the qualitative description of the Newton differential equation. In September of that year he received his Master’s degree.

From 1991 until 1996 he was employed at the Eindhoven University of Technology. He has carried out the research for this thesis as a member of the Computer Graphics group.