MICHAEL KNIELY

RIEMANNIAN METHODS FOR OPTIMIZATION IN SHAPE SPACE

Masterarbeit

zur Erlangung des akademischen Grades eines
Master of Science
an der Naturwissenschaftlichen Fakultät der
Karl-Franzens-Universität Graz

Begutachter:
Ao.Univ.-Prof. Mag.rer.nat. Dr.techn. Wolfgang Ring
Institut für Mathematik und Wissenschaftliches Rechnen

2012

Contents

Chapter 1

Introduction

The aim of this thesis is to investigate the applicability of two optimization algorithms in shape space and to apply them to the shape from shading (SFS) problem. More precisely, we use the steepest descent method and the nonlinear conjugate gradient (NCG) method to solve the SFS-problem in a certain shape space which we endow with an appropriate Riemannian inner product. Instead of steps along straight lines we shall take steps along geodesics with respect to the Riemannian metric. Moreover, we will have to use the concept of parallel displacement in order to apply the NCG-algorithm.

In the context of optimization in vector spaces one often pursues the following idea. Given a vector space V, a function $f : V \to \mathbb{R}$ and a point $x_0 \in V$, one chooses a descent direction $v \in V$. Then a linesearch method with a certain step-size control is performed to find a scalar $\alpha \in \mathbb{R}$. The new iterate is then defined by $x_1 := x_0 + \alpha \cdot v$. This idea essentially uses the underlying vector space structure of V. First, the descent direction v is a priori in the tangent space $T_{x_0}V$; but since V is a vector space, $T_{x_0}V$ can be identified with V. Second, the definition of x_1 makes use of the operations $+$ and $\cdot$ in V. In the more general setting of Riemannian manifolds such identifications and vector operations are not at hand, hence, one has to use a different strategy, for example the following one. Given a manifold M, a function $f : M \to \mathbb{R}$ and a point $x_0 \in M$, one chooses a descent direction $v \in T_{x_0}M$. Then one calculates the geodesic $u : \mathbb{R} \to M$ through x_0 parametrized by arc length with $u(0) = x_0$ and tangent vector $\dot{u}(0) = v$. Afterwards a linesearch method with a certain step-size control is performed to find a scalar $\alpha \in \mathbb{R}$. The new iterate is then defined by $x_1 := u(\alpha)$. The descent direction depends for sure on the optimization method which is used. Subsequently, we shall concentrate ourselves on the (geodesic) steepest descent and the (geodesic) NCG-method using the Fletcher-Reeves scheme. The latter method has been analyzed by W. Ring and B. Wirth [7] in the context of Riemannian manifolds.

A special Riemannian manifold is a shape space, which is endowed with a certain Riemannian inner product. In general, a shape space is a set whose elements can be identified with geometrical objects. These objects may be smooth curves or surfaces as well as polygons and other types of geometrical shapes. For typical examples, which are recently studied in research,

see the article of P. Michor and D. Mumford [6] for infinite-dimensional shape spaces and the article of M. Kilian et al. [4] for finite-dimensional shape spaces. However, we will use the shape space of triangular meshes in $\mathbb{R}^3$ to solve the SFS-problem. This shape space shall be endowed with different Riemannian metrics in order to compare the results of the optimization algorithms for these different metrics in the shape space.

Roughly speaking, the SFS-Problem is the following: Given a shading image of a surface, i.e. an image of the surface which is illuminated in a certain way, we want to reconstruct this surface. The first approach towards a solution of this problem was presented by Horn in [2]. The basic idea is to determine several paths on the surface, so called characteristics. In order to reconstruct the surface topography sufficiently well, several characteristics which are close enough to each other are necessary. See [2] and also [5] for a more detailed description of this approach. However, there are several other methods which have been proposed within the last decades to solve the SFS-Problem. An overview of these methods is given in [8].

The organization of the thesis is as follows. Chapter 2 is devoted to the theoretical studies which are necessary to implement the steepest descent algorithm and the NCG-algorithm in a shape space with a Riemannian metric. In chapter 3 we apply these minimization algorithms to solve the SFS-problem for three different shapes. In addition, we compare the obtained results for different Riemannian metrics in an appropriate shape space. Finally, a short chapter with remarks and an outlook to future research concludes the thesis.

In detail, we establish all the results from the literature in section 2.1 which we need for the remainder of the thesis. Subsection 2.1.1 collcets fundamental definitions, like manifolds and tangent vectors. Subsection 2.1.2 derives the geodesic equation and the equation of parallel translation for a general connection on a manifold. To the end, we proof that there exists a unique torsion free and metric connection on each Riemannian manifold – the *Levi-Civita-connection*. Subsection 2.1.3 introduces the notion of shape spaces and provides several examples. Besides, various technical notations will be defined in this subsection. Afterwards, we construct various Riemannian metrics in section 2.2. In section 2.3 we deduce the explicit geodesic equations for the considered metrics, and in section 2.4 we establish the explicit equations of parallel translation for these metrics.

After a short introduction to the SFS-problem, we define in section 3.1 that function on the shape space which we will minimize in order to solve the SFS-problem. For this case, we also calculate the optimal descent direction for each Riemannian metric. In section 3.2 we present the implementation of the two considered optimization techniques. The steepest descent method and the function evaluating the geodesic equations will be discussed in subsection 3.2.1; the NCG-method together with the function calculating the parallel translate of a vector is explained in subsection 3.2.2. Finally, we collect in section 3.3 the results which we obtained with the different Riemannian metrics and the two minimization algorithms. In this context, we shall compare numerical facts as well as the visual impression of the reconstructed surfaces.

Chapter 2

Construction Of Optimization Algorithms In Shape Space

2.1 Tools from Differential Geometry

2.1.1 Preliminaries

The concepts of differential geometry presented below are nowadays standard techniques, so there will be nothing new to researchers. Instead, the subsections 2.1.1 and 2.1.2 should be seen as a collection of facts which will be necessary or important for the subsequent studies in the thesis. The following ideas and proofs are mainly based on [1] and [3].

Definition 2.1. A *manifold* M^n of dimension n is a set satisfying the following properties.

- M^n is a connected Hausdorff space with a countable base at each $p \in M^n$.
- There exists an open covering $\mathcal{C}$ of M^n with the following property. For every $U \in \mathcal{C}$ there exists an open set $\Omega \subset \mathbb{R}^n$ and a homeomorphism $x_U : U \to \Omega$. (We call U a *(coordinate) patch*, x_U a *(coordinate) map* (or *local coordinates* of M^n) and (U, x_U) a *(coordinate) chart*.)

Moreover, we call a manifold M^n a *differentiable manifold*, if for all $U, V \in \mathcal{C}$ with $U \cap V \neq \emptyset$,

$$x_V \circ x_U^{-1} : x_U(U \cap V) \to x_V(U \cap V)$$

is differentiable.

Definition 2.2. Let M^n be a differentiable manifold.

1. A pair (W, y) of an open set $W \subset M^n$ and a homeomorphism $y : W \to y(W) \subset \mathbb{R}^n$ is called *compatible* with M^n, if for all charts (U, x_U) with $U \cap W \neq \emptyset$,

$$y \circ x_U^{-1} : x_U(U \cap W) \to y(U \cap W) \quad \text{and} \quad x_U \circ y^{-1} : y(U \cap W) \to x_U(U \cap W)$$

are differentiable.

2. We call
$$\mathcal{A} := \{(W,y)\big|(W,y) \text{ is compatible with } M^n\}$$
an *atlas* of M^n.

Definition 2.3. Let M^n be a differentiable manifold, $p \in M^n$ and $\mathcal{A}_p := \{(U,x_U) \in \mathcal{A}\big|\ p \in U\}$. Then, a *tangent vector* X at p is a map
$$X : \begin{cases} \mathcal{A}_p & \to \mathbb{R}^n, \\ (U,x_U) & \mapsto X_U = (X_U^1,...,X_U^n) \end{cases}$$
such that for all $(U,x_U),(V,x_V) \in \mathcal{A}_p$,
$$X_V^i = \sum_{j=1}^n \left(\frac{\partial x_V^i}{\partial x_U^j}(p)\right) X_U^j.$$

The *tangent space* T_pM^n to M^n at p is the set of all tangent vectors to M^n at p, and the *tangent bundle*
$$TM^n := \bigcup_{p\in M^n} T_pM^n$$
is the union of all tangent spaces T_pM^n.

Remark 2.4. Alternatively, one may also use the following equivalent definition of a tangent vector. Let M^n be a differentiable manifold, $p \in M^n$ and $\mathcal{A}_p := \{(U,x_U) \in \mathcal{A}\big|\ p \in U\}$. Now, let $0 \in I \subset \mathbb{R}$ be an open interval and $\gamma : I \to M^n$ be a differentiable curve with $\gamma(0) = p$. Then, for all $(U,x_U) \in \mathcal{A}_p$ there exists an open interval $0 \in J_U \subset I$ such that $\gamma(J_U) \subset U$ and
$$\gamma_U := x_U \circ \gamma \ :\ J_U \to \mathbb{R}^n$$
is differentiable. Hence, we may consider the vector
$$X_U = (X_U^1,...,X_U^n) := \gamma_U'(0) \in \mathbb{R}^n.$$
A tangent vector X at p can now be defined as the collection of all vectors X_U with $(U,x_U) \in \mathcal{A}_p$; formally we write
$$X = (X_U)_{(U,x_U)\in\mathcal{A}_p}.$$
Furthermore, we also have that for all $(U,x_U),(V,x_V) \in \mathcal{A}_p$,
$$X_V = \frac{d}{dt}(x_V \circ \gamma)(0) = \frac{d}{dt}(x_V \circ x_U^{-1} \circ x_U \circ \gamma)(0) = D(x_V \circ x_U^{-1})\left((x_U \circ \gamma)(0)\right) \cdot \frac{d}{dt}(x_U \circ \gamma)(0)$$
and, consequently,
$$X_V^i = \left(\left(\frac{\partial x_V}{\partial x_U}(p)\right)_{ij} \cdot X_U\right)_i = \sum_{j=1}^n \left(\frac{\partial x_V^i}{\partial x_U^j}(p)\right) X_U^j.$$

Remark 2.5. Let M^n be a differentiable manifold, $p \in M^n$, U, V coordinate patches containing p, $f \in C^\infty(M^n)$ and $X \in T_pM^n$. Then,

$$\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_V^i}(p) \right) X_V^i = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_V^i}(p) \right) \sum_{j=1}^{n} \left(\frac{\partial x_V^i}{\partial x_U^j}(p) \right) X_U^j = \sum_{j=1}^{n} \left(\frac{\partial f}{\partial x_U^j}(p) \right) X_U^j.$$

Consequently, the following definition is independent of the coordinates used.

Definition 2.6. Let M^n be a differentiable manifold, $p \in M^n$, $(x^1, ..., x^n)$ local coordinates on M^n around p, $f \in C^\infty(M^n)$ and $X \in T_pM^n$. Then, we define

$$X(f) := \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x^i}(p) \right) X^i.$$

Remark 2.7. In the situation of definition 2.6, one immediately sees that T_pM^n is a real vector space and that

$$\left\{ \frac{\partial}{\partial x^1}\Big|_p, ..., \frac{\partial}{\partial x^n}\Big|_p \right\}$$

is a basis of T_pM^n. Consequently, each tangent vector $X \in T_pM^n$ can be identified with a differential operator on smooth functions $f \in C^\infty(M^n)$.

Definition 2.8. Let M^n be a differentiable manifold, (U, x) a coordinate chart. Then, a *vector field* X on U is a map

$$X : \begin{cases} U & \to TM^n, \\ p & \mapsto X_p = \sum\limits_{i=1}^{n} X^i(p) \frac{\partial}{\partial x^i}\big|_p. \end{cases}$$

where $X^i \in C^\infty(U)$ holds for all $i \in \{1, ..., n\}$. The set of all tangent vector fields on M^n is denoted by $\mathcal{V}(M^n)$.

Remark 2.9. For the remainder of the thesis, we make the following conventions. Every manifold is assumed to be a differentiable manifold; and every curve in a manifold is assumed to be of class C^∞. In addition, we shall often use the notation

$$\partial_i := \frac{\partial}{\partial u^i}$$

for the basis vectors of the tangent space T_pM^n to a manifold M^n at a point $p \in M^n$. If M^n is a manifold with a (Pseudo-)Riemannian metric $\langle \cdot, \cdot \rangle$, we denote the (pseudo-)metric tensor and its inverse by

$$g_{ij} := \langle \partial_i, \partial_j \rangle \quad \text{and} \quad g^{ij} := (G^{-1})_{ij} \quad \text{where} \quad G := (g_{ij})_{ij}.$$

Furthermore, we use the *Einstein summation convention*: Any index which occurs twice in a product is to be summed from 1 up to the space dimension, e.g.

$$a^i b^j \partial_i \partial_j = \sum_{i=1}^{n} \sum_{j=1}^{n} a^i b^j \partial_i \partial_j.$$

2.1.2 Connections, Geodesics and Parallel Transport

Definition 2.10. Let M^n be a manifold. Then, a map $D : \mathcal{V}(M^n) \times \mathcal{V}(M^n) \to \mathcal{V}(M^n)$ is called a *connection* on M^n, if the following properties are satisfied for all $v, w, X, Y \in \mathcal{V}(M^n)$, $a, b \in \mathbb{R}$ and $f \in C^\infty(M^n)$:

$$\begin{aligned} D_X(av + bw) &= aD_X v + bD_X w, \\ D_{aX+bY} v &= aD_X v + bD_Y v, \\ D_X(fv) &= X(f)v + fD_X v. \end{aligned}$$

Moreover, we call a connection D *torsion free*, if for all $X, Y \in \mathcal{V}(M^n)$,

$$[X, Y] = D_X Y - D_Y X,$$

where $[X, Y]$ denotes the Lie bracket. If M^n is a Riemannian manifold with metric $\langle \cdot, \cdot \rangle$, we call a connection D *metric*, if for all $Z \in \mathcal{V}(M^n)$,

$$X\langle Y, Z\rangle = \langle D_X Y, Z\rangle + \langle Y, D_X Z\rangle.$$

Definition 2.11. Let M^n be a manifold, D a connection on M^n, U a coordinate patch and $(e_1, ..., e_n)$ a basis of $T_p M^n$ for all $p \in U$. Then, the symbols ω_{jk}^i defined via

$$D_{e_j} e_k = e_i \omega_{jk}^i$$

are called the *coefficients of the connection* D with respect to $(e_1, ..., e_n)$.

Definition 2.12. Let M^n be a manifold with local coordinates $(u^1, ..., u^n)$, D a connection on M^n, $x = x(u(t))$ a curve in M^n, $Y \in \mathcal{V}(M^n)$ and $T \in \mathcal{V}(M^n)$ such that $T = dx/dt$ along x.

1. x is called a *geodesic*, if
$$D_T T = 0.$$
2. Y is said to be *parallel displaced* along x, if
$$D_T Y = 0.$$

Proposition 2.13. *Let M^n be a manifold with local coordinates $(u^1, ..., u^n)$, D a connection on M^n, $x = x(u(t))$ a curve in M^n, $Y \in \mathcal{V}(M^n)$, $T \in \mathcal{V}(M^n)$ such that $T = dx/dt$ along x.*

1. x is a geodesic, if and only if

$$\frac{d^2 u^i}{dt^2} + \omega_{jk}^i \frac{du^j}{dt} \frac{du^k}{dt} = 0 \qquad \text{for all } \; i \in \{1, ..., n\}. \tag{2.1}$$

2. Y is parallel displaced along u, if and only if

$$\frac{dY^i}{dt} + \omega_{jk}^i \frac{du^j}{dt} Y^k = 0 \qquad \text{for all } \; i \in \{1, ..., n\}. \tag{2.2}$$

Proof. Let

$$T = \frac{dx}{dt} = \frac{du^i}{dt}\frac{\partial}{\partial u^i} = T^i \frac{\partial}{\partial u^i}$$

denote the tangent vector field of x.

1. We use the properties of the connection D and find

$$\begin{aligned} D_T T &= D_{T^j \frac{\partial}{\partial u^j}} \left(T^k \frac{\partial}{\partial u^k} \right) = T^j D_{\frac{\partial}{\partial u^j}} \left(T^k \frac{\partial}{\partial u^k} \right) = T^j \left(\frac{\partial T^k}{\partial u^j}\frac{\partial}{\partial u^k} + T^k D_{\frac{\partial}{\partial u^j}} \frac{\partial}{\partial u^k} \right) \\ &= T^j \left(\frac{\partial T^k}{\partial u^j}\frac{\partial}{\partial u^k} + T^k \omega^i_{jk} \frac{\partial}{\partial u^i} \right) = T^j \left(\frac{\partial T^i}{\partial u^j} + T^k \omega^i_{jk} \right) \frac{\partial}{\partial u^i} \\ &= \left(\frac{du^j}{dt}\frac{\partial T^i}{\partial u^j} + \omega^i_{jk} T^j T^k \right) \frac{\partial}{\partial u^i} = \left(\frac{dT^i}{dt} + \omega^i_{jk} T^j T^k \right) \frac{\partial}{\partial u^i}. \end{aligned}$$

   Thus, the claim follows since $(\partial/\partial u^1, ..., \partial/\partial u^n)$ form a basis for each tangent space.

2. Using the same arguments as above and the representation

$$Y = Y^i \frac{\partial}{\partial u^i}$$

   one finds

$$\begin{aligned} D_T Y &= D_{T^j \frac{\partial}{\partial u^j}} \left(Y^k \frac{\partial}{\partial u^k} \right) = T^j D_{\frac{\partial}{\partial u^j}} \left(Y^k \frac{\partial}{\partial u^k} \right) = T^j \left(\frac{\partial Y^k}{\partial u^j}\frac{\partial}{\partial u^k} + Y^k D_{\frac{\partial}{\partial u^j}} \frac{\partial}{\partial u^k} \right) \\ &= T^j \left(\frac{\partial Y^k}{\partial u^j}\frac{\partial}{\partial u^k} + Y^k \omega^i_{jk} \frac{\partial}{\partial u^i} \right) = T^j \left(\frac{\partial Y^i}{\partial u^j} + Y^k \omega^i_{jk} \right) \frac{\partial}{\partial u^i} \\ &= \left(\frac{du^j}{dt}\frac{\partial Y^i}{\partial u^j} + \omega^i_{jk} T^j Y^k \right) \frac{\partial}{\partial u^i} = \left(\frac{dY^i}{dt} + \omega^i_{jk} T^j Y^k \right) \frac{\partial}{\partial u^i}, \end{aligned}$$

   the desired representation. □

Definition 2.14. In the sequel, we call equation (2.1) the *geodesic equation* and equation (2.2) the *equation of parallel translation*.

Lemma 2.15. *Let M^n be a manifold, D a torsion free connection on M^n and $(u^1, ..., u^n)$ be local coordinates for M^n. Then*

$$\omega^i_{jk} = \omega^i_{kj}$$

for all $i, j, k \in \{1, ..., n\}$ with respect to $(\partial/\partial u^1, ..., \partial/\partial u^n)$.

Proof. Let X, Y be tangent vector fields on M^n. We know that the i-th component of the Lie bracket is given by

$$[X, Y]^i = X^j \frac{\partial Y^i}{\partial u^j} - Y^j \frac{\partial X^i}{\partial u^j}$$

and that

$$\begin{aligned} (D_X Y - D_Y X)^i &= (D_{\partial_j X^j} \partial_k Y^k - D_{\partial_j Y^j} \partial_k X^k)^i = (X^j D_{\partial_j} \partial_k Y^k - Y^j D_{\partial_j} \partial_k X^k)^i \\ &= \left(X^j \frac{\partial Y^k}{\partial u^j} \partial_k + X^j Y^k D_{\partial_j} \partial_k - Y^j \frac{\partial X^k}{\partial u^j} \partial_k - X^k Y^j D_{\partial_j} \partial_k \right)^i \\ &= X^j \frac{\partial Y^i}{\partial u^j} + X^j Y^k \omega^i_{jk} - Y^j \frac{\partial X^i}{\partial u^j} - X^k Y^j \omega^i_{jk}. \end{aligned}$$

Thus, one finds after changing indices

$$0 = X^j Y^k (\omega^i_{jk} - \omega^i_{kj}),$$

and therefore, $\omega^i_{jk} = \omega^i_{kj}$ for all $i, j, k \in \{1, ..., n\}$. □

Remark 2.16. Due to the result of Lemma 2.15, torsion free connections are often called *symmetric* since its coefficients are symmetric in the two lower indices. In the sequel, we will prefer the term torsion free for such a connection.

Definition 2.17. Let M^n be a manifold with a connection D, $U \subset \mathbb{R}$, $x : U \to M^n$, $x = x(t)$ be a curve, $Y \in \mathcal{V}(M^n)$ and $T \in \mathcal{V}(M^n)$ such that $T = dx/dt$ along x. Then we set, along x,

$$\frac{D}{dt} Y := \frac{DY}{dt} := D_T Y.$$

Lemma 2.18. *Let M^n be a manifold with a torsion free connection D, $U \subset \mathbb{R}^2$ open with coordinates (u, v) and $x : U \to M^n$ be twice continuously differentiable. Then*

$$\frac{D}{\partial u}\left(\frac{\partial x}{\partial v}\right) = \frac{D}{\partial v}\left(\frac{\partial x}{\partial u}\right).$$

Proof. At first, we choose local coordinates $(y^1, ..., y^n)$ of M^n. Then we know that $\partial x/\partial u = (\partial y^i/\partial u)\partial/\partial y^i$ and $\partial x/\partial v = (\partial y^j/\partial v)\partial/\partial y^j$. Consequently,

$$\begin{aligned}
\frac{D}{\partial u}\left(\frac{\partial x}{\partial v}\right) &= D_{\frac{\partial y^i}{\partial u}\frac{\partial}{\partial y^i}}\left(\frac{\partial y^j}{\partial v}\frac{\partial}{\partial y^j}\right) = \frac{\partial y^i}{\partial u} D_{\frac{\partial}{\partial y^i}}\left(\frac{\partial y^j}{\partial v}\frac{\partial}{\partial y^j}\right) \\
&= \frac{\partial y^i}{\partial u}\left(\frac{\partial^2 y^j}{\partial y^i \partial v}\frac{\partial}{\partial y^j} + \frac{\partial y^j}{\partial v} D_{\frac{\partial}{\partial y^i}}\left(\frac{\partial}{\partial y^j}\right)\right) \\
&= \frac{\partial^2 y^j}{\partial u \partial v}\frac{\partial}{\partial y^j} + \frac{\partial y^i}{\partial u}\frac{\partial y^j}{\partial v}\omega^k_{ij}\frac{\partial}{\partial y^k}
\end{aligned}$$

and since D is torsion free, $\omega^k_{ij} = \omega^k_{ji}$. Hence, the last expression is symmetric in u and v, which prooves the claim. □

Lemma 2.19. *Let M^n be a manifold with a metric connection D, $x(t)$ a curve in M^n, $U, V \in \mathcal{V}(M^n)$ and $T \in \mathcal{V}(M^n)$ such that $T = dx/dt$ along x. Then, along x,*

$$\frac{d}{dt}\langle U, V\rangle = \left\langle \frac{DU}{dt}, V\right\rangle + \left\langle U, \frac{DV}{dt}\right\rangle.$$

Proof. From the characterization of tangent vectors to a manifold as differential operators, we know that $T\langle U, V\rangle$ is the directional derivative of $\langle U, V\rangle$ in direction T, therefore $T\langle U, V\rangle = d/dt\langle U, V\rangle$ along x. Thus, using definition 2.17, we find

$$\frac{d}{dt}\langle U, V\rangle = T\langle U, V\rangle = \langle D_T U, V\rangle + \langle U, D_T V\rangle = \left\langle \frac{DU}{dt}, V\right\rangle + \left\langle U, \frac{DV}{dt}\right\rangle$$

since D was assumed to be a metric connection. □

Theorem 2.20. *Let M^n be a Riemannian manifold with metric $\langle\cdot,\cdot\rangle$. Then there exists a unique metric and torsion free connection $\nabla : \mathcal{V}(M^n) \times \mathcal{V}(M^n) \to \mathcal{V}(M^n)$. This connection is given by*

$$\langle \nabla_X Y, Z\rangle = \frac{1}{2}\big(X\langle Y,Z\rangle + Y\langle Z,X\rangle - Z\langle X,Y\rangle - \langle X,[Y,Z]\rangle + \langle Y,[Z,X]\rangle + \langle Z,[X,Y]\rangle\big) \quad (2.3)$$

and with local coordinates $(x^1,...,x^n)$,

$$\omega^i_{jk} = \frac{1}{2}g^{li}\left(\frac{\partial g_{lj}}{\partial x^k} + \frac{\partial g_{kl}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^l}\right) \quad (2.4)$$

with respect to $(\partial/\partial x^1,...,\partial/\partial x^n)$.

Proof. First, we show the uniqueness of such a connection. For this case, let ∇ be a metric and torsion free connection and $X,Y,Z \in \mathcal{V}(M^n)$. Then

$$\begin{aligned} & X\langle Y,Z\rangle + Y\langle Z,X\rangle - Z\langle X,Y\rangle \\ = \;& \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle + \langle \nabla_Y Z, X\rangle + \langle Z, \nabla_Y X\rangle - \langle \nabla_Z X, Y\rangle - \langle X, \nabla_Z Y\rangle \\ = \;& 2\langle \nabla_X Y, Z\rangle - \langle Z,[X,Y]\rangle + \langle Y,[X,Z]\rangle + \langle X,[Y,Z]\rangle \end{aligned}$$

which proves the claimed representation using $[X,Z] = -[Z,X]$. Hence, such a connection is unique.

Now, define for each fixed $X,Y \in \mathcal{V}(M^n)$ the smooth covector field α such that $\alpha(Z)$ for $Z \in \mathcal{V}(M^n)$ is the right-hand-side of equation (2.3). Then $\alpha(Z)$ is $\mathbb{R}$-linear in Z. At each point $p \in M^n$ the tangent space T_pM^n is finite dimensional and

$$i : T_pM^n \to T_p^*M^n, \quad i(u)(v) = \langle u, v\rangle$$

is an injective linear map since the Riemannian metric $\langle\cdot,\cdot\rangle$ is nondegenerate; hence, i must be an isomorphism and $T_p^*M^n$ can be identified with T_pM^n via i. Now, $\alpha \in T_p^*M^n$, and therefore, there exists a unique vector $A(p) \in T_pM^n$ such that $\alpha(Z(p)) = \langle A(p), Z(p)\rangle$. Consequently, there exists a unique vector field $A \in \mathcal{V}(M^n)$ such that

$$\alpha(Z) = \langle A, Z\rangle.$$

In deed, A is smooth since α is a smooth covector field. We then set $\nabla_X Y := A$ and have to show, that this defines a metric and torsion free connection. At first, $\mathbb{R}$-linearity is clear since the right-hand-side of equation (2.3) is linear in X and Y. Furthermore, observe that for $f \in C^\infty(M^n)$,

$$\begin{aligned} & \langle \nabla_X(fY), Z\rangle \\ = \;& \frac{1}{2}\left(X\langle fY,Z\rangle + fY\langle Z,X\rangle - Z\langle X,fY\rangle - \langle X,[fY,Z]\rangle + \langle fY,[Z,X]\rangle + \langle Z,[X,fY]\rangle\right) \\ = \;& f\langle \nabla_X Y, Z\rangle + \frac{1}{2}\left(\langle X(f)Y, Z\rangle - \langle X, Z(f)Y\rangle + \langle X, Z(f)Y\rangle + \langle Z, X(f)Y\rangle\right) \\ = \;& f\langle \nabla_X Y, Z\rangle + \langle X(f)Y, Z\rangle. \end{aligned}$$

Thus ∇ defines a connection on M^n. Finally, equation 2.3 yields

$$\begin{aligned}
& \langle \nabla_X Y, Z\rangle + \langle \nabla_X Z, Y\rangle \\
= \;& \frac{1}{2}\left(X\langle Y,Z\rangle + Y\langle Z,X\rangle - Z\langle X,Y\rangle - \langle X,[Y,Z]\rangle + \langle Y,[Z,X]\rangle + \langle Z,[X,Y]\rangle\right) \\
+ \;& \frac{1}{2}\left(X\langle Z,Y\rangle + Z\langle Y,X\rangle - Y\langle X,Z\rangle - \langle X,[Z,Y]\rangle + \langle Z,[Y,X]\rangle + \langle Y,[X,Z]\rangle\right) \\
= \;& X\langle Y,Z\rangle
\end{aligned}$$

and

$$\begin{aligned}
& \langle \nabla_X Y, Z\rangle - \langle \nabla_Y X, Z\rangle \\
= \;& \frac{1}{2}\left(X\langle Y,Z\rangle + Y\langle Z,X\rangle - Z\langle X,Y\rangle - \langle X,[Y,Z]\rangle + \langle Y,[Z,X]\rangle + \langle Z,[X,Y]\rangle\right) \\
- \;& \frac{1}{2}\left(Y\langle X,Z\rangle + X\langle Z,Y\rangle - Z\langle Y,X\rangle - \langle Y,[X,Z]\rangle + \langle X,[Z,Y]\rangle + \langle Z,[Y,X]\rangle\right) \\
= \;& \langle [X,Y], Z\rangle,
\end{aligned}$$

therefore, ∇ is a metric and torsion free connection.

It remains to show the formula for the coefficients ω^i_{jk}. For this case, we consider a patch U, the k-th coordinate curve x^k and vector fields $X, Y \in \mathcal{V}(M^n)$. Along this curve,

$$\frac{\partial}{\partial x^k}\langle X,Y\rangle = \frac{\partial}{\partial x^k}\langle \partial_i X^i, \partial_j Y^j\rangle = \frac{\partial}{\partial x^k}(g_{ij}X^iY^j) = \frac{\partial g_{ij}}{\partial x^k}X^iY^j + g_{ij}\frac{\partial X^i}{\partial x^k}Y^j + g_{ij}X^i\frac{\partial Y^j}{\partial x^k}$$

and

$$\begin{aligned}
& \langle \nabla_{\partial_k} X, Y\rangle + \langle X, \nabla_{\partial_k} Y\rangle \\
= \;& \langle \nabla_{\partial_k}\partial_i X^i, \partial_j Y^j\rangle + \langle \partial_i X^i, \nabla_{\partial_k}\partial_j Y^j\rangle \\
= \;& \left\langle \frac{\partial X^i}{\partial x^k}\partial_i + X^i\nabla_{\partial_k}\partial_i, \partial_j Y^j\right\rangle + \left\langle \partial_i X^i, \frac{\partial Y^j}{\partial x^k}\partial_j + Y^j\nabla_{\partial_k}\partial_j\right\rangle \\
= \;& \langle \partial_i,\partial_j\rangle\frac{\partial X^i}{\partial x^k}Y^j + \langle \partial_l\omega^l_{ki},\partial_j\rangle X^iY^j + \langle \partial_i,\partial_j\rangle X^i\frac{\partial Y^j}{\partial x^k} + \langle \partial_i,\partial_l\omega^l_{kj}\rangle X^iY^j \\
= \;& g_{ij}\frac{\partial X^i}{\partial x^k}Y^j + g_{lj}\omega^l_{ki}X^iY^j + g_{ij}X^i\frac{\partial Y^j}{\partial x^k} + g_{il}\omega^l_{kj}X^iY^j.
\end{aligned}$$

Since ∇ is metric, we have

$$\frac{\partial g_{ij}}{\partial x^k} = g_{lj}\omega^l_{ki} + g_{il}\omega^l_{kj}.$$

Moreover, we know that $\omega^i_{jk} = \omega^i_{kj}$ for all $i,j,k \in \{1,...,n\}$ since ∇ is torsion free. Now, we find

$$\frac{\partial g_{lj}}{\partial x^k} + \frac{\partial g_{kl}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^l} = g_{ij}\omega^i_{kl} + g_{li}\omega^i_{kj} + g_{il}\omega^i_{jk} + g_{ki}\omega^i_{jl} - g_{ik}\omega^i_{lj} - g_{ji}\omega^i_{lk} = 2g_{li}\omega^i_{jk},$$

which finally gives

$$\omega^i_{jk} = \frac{1}{2}g^{li}\left(\frac{\partial g_{lj}}{\partial x^k} + \frac{\partial g_{kl}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^l}\right)$$

and finishes the proof. □

Definition 2.21. The connection ∇ characterized in Theorem 2.20 is called the *Levi-Civita connection* and its coefficients, from now on denoted by Γ^i_{jk}, are called the *Christoffel symbols*.

Definition 2.22. Let M^n be a Riemannian manifold with metric $\langle\cdot,\cdot\rangle$ and C a curve in M^n.

A *variation* of the curve C is a twice continuously differentiable map

$$x : \begin{cases} [0,L]\times(-1,1) & \to M^n, \\ (s,\alpha) & \mapsto x(s,\alpha), \end{cases}$$

where $x(s,0)$ is the parametrization of C and L the length of C; moreover, we demand that s is the arc length parameter for C.

We define the *arc length functional*

$$L(\alpha) := \int_0^L \left\langle \frac{\partial x(s,\alpha)}{\partial s}, \frac{\partial x(s,\alpha)}{\partial s} \right\rangle^{\frac{1}{2}} ds,$$

which is the length of the curve $x(\cdot,\alpha)$.

Proposition 2.23. *Let M^n be a Riemannian manifold with metric $\langle\cdot,\cdot\rangle$, ∇ the Levi-civita connection, C be a geodesic in M^n and $x(s,\alpha)$ be a variation of C such that $x(0,\alpha) = x(0,0)$ and $x(1,\alpha) = x(1,0)$ for all $\alpha \in (-1,1)$. Then*

$$L'(0) = 0.$$

In other words, a geodesic is a critical point of the arc length functional for variations which keep the endpoints fixed.

Proof. Let $x(s,\alpha)$ be a variation of a geodesic C in M^n such that $x(0,\alpha) = x(0,0)$ and $x(1,\alpha) = x(1,0)$ for all $\alpha \in (-1,1)$. At first, ∇ is a metric connection and Lemma 2.19 yields

$$\frac{\partial}{\partial\alpha}\left\langle \frac{\partial x}{\partial s}, \frac{\partial x}{\partial s}\right\rangle = 2\left\langle \frac{\nabla}{\partial\alpha}\left(\frac{\partial x}{\partial s}\right), \frac{\partial x}{\partial s}\right\rangle$$

and

$$\frac{\partial}{\partial s}\left\langle \frac{\partial x}{\partial\alpha}, \frac{\partial x}{\partial s}\right\rangle = \left\langle \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial\alpha}\right), \frac{\partial x}{\partial s}\right\rangle + \left\langle \frac{\partial x}{\partial\alpha}, \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial s}\right)\right\rangle.$$

In addition, ∇ is torsion free, hence,

$$\frac{\nabla}{\partial\alpha}\left(\frac{\partial x}{\partial s}\right) = \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial\alpha}\right)$$

due to Lemma 2.18. Together, we deduce

$$\begin{aligned} L'(\alpha) &= \frac{1}{2}\int_0^L \left\langle \frac{\partial x}{\partial s}, \frac{\partial x}{\partial s}\right\rangle^{-\frac{1}{2}} \frac{\partial}{\partial\alpha}\left\langle \frac{\partial x}{\partial s}, \frac{\partial x}{\partial s}\right\rangle ds \\ &= \int_0^L \left\langle \frac{\partial x}{\partial s}, \frac{\partial x}{\partial s}\right\rangle^{-\frac{1}{2}} \left\langle \frac{\nabla}{\partial\alpha}\left(\frac{\partial x}{\partial s}\right), \frac{\partial x}{\partial s}\right\rangle ds \\ &= \int_0^L \left\langle \frac{\partial x}{\partial s}, \frac{\partial x}{\partial s}\right\rangle^{-\frac{1}{2}} \left\langle \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial\alpha}\right), \frac{\partial x}{\partial s}\right\rangle ds. \end{aligned}$$

Now, s is the arc length parameter for $x(\cdot, 0)$; consequently,

$$\begin{aligned} L'(0) &= \int_0^L \left\langle \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial \alpha}\right), \frac{\partial x}{\partial s}\right\rangle \\ &= \int_0^L \left(\frac{\partial}{\partial s}\left\langle \frac{\partial x}{\partial \alpha}, \frac{\partial x}{\partial s}\right\rangle - \left\langle \frac{\partial x}{\partial \alpha}, \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial s}\right)\right\rangle\right) ds \\ &= \left\langle \frac{\partial x}{\partial \alpha}(1,0), \frac{\partial x}{\partial s}(1,0)\right\rangle - \left\langle \frac{\partial x}{\partial \alpha}(0,0), \frac{\partial x}{\partial s}(0,0)\right\rangle - \int_0^L \left\langle \frac{\partial x}{\partial \alpha}, \frac{\nabla}{\partial s}\left(\frac{\partial x}{\partial s}\right)\right\rangle ds \\ &= -\int_0^L \left\langle \frac{\partial x}{\partial \alpha}, \nabla_T T\right\rangle ds \\ &= 0 \end{aligned}$$

if $T \in \mathcal{V}(M^n)$ with $T = \partial x(s,0)/\partial s$ along C denotes the tangent vector field along C. □

2.1.3 Shape Spaces

Generally, a *shape space* is a set whose elements can be identified with geometrical objects, like smooth surfaces, polgons and so on. This definition (or better characterization) includes now a large variety of such shape spaces, which may be finite-dimensional as well as infinite-dimensional.

In contrast to geodesics and parallel transport on manifolds, the concept of shape spaces is a more recent topic of modern research, with a focus on theoretical issues (e.g. [6]) as well as geometric applications (e.g. [4]). First, we will have a short look at different shape spaces and the corresponding practical applicability for certain geometric problems. Finally, we introduce some notations which will be used in the sequel, and we reformulate the geodesic equation and the equation of parallel translation for practical reasons.

Some typical examples for infinite-dimensional shape spaces together with possible Riemannian metrics are studied by P. Michor and D. Mumford in [6]. They consider, for example, the set

$$\mathcal{S}_1 := \mathrm{Emb}(S^1, \mathbb{R}^2)/\operatorname{Diff}(S^1)$$

of the manifold of C^∞ embeddings of S^1 into $\mathbb{R}^2$ modulo the group of C^∞ diffeomorphisms of S^1. For sure, one could also work just with $\mathcal{S}_2 := \mathrm{Emb}(S^1, \mathbb{R}^2)$, but then there are different elements in $\mathcal{S}_2$ which would be identified with the same object. Consider for example

$$i_1 : \begin{cases} S^1 \equiv [0, 2\pi] & \to \mathbb{R}^2, \\ \phi & \mapsto (\cos(\phi), \sin(\phi)) \end{cases} \quad \text{and} \quad i_2 : \begin{cases} S^1 \equiv [0, 2\pi] & \to \mathbb{R}^2, \\ \phi & \mapsto (\sin(\phi), \cos(\phi)) \end{cases}$$

which are different elements in $\mathcal{S}_2$ but correspond to the same point set in $\mathbb{R}^2$, the unit circle. To overcome this ambiguity, one usually has to deal with quotient spaces. In our example, i_1 and i_2 belong to the same coset in $\mathcal{S}_1$ since i_1 and i_2 are just two different parametrizations of the unit circle in $\mathbb{R}^2$ – and such reparametrizations are identified with each other in $\mathcal{S}_1$. Furthermore, the authors show that one may also consider the shape space $\mathcal{S}_3$ of all unparametrized C^∞ simple closed curves in $\mathbb{R}^2$; more precisely, they claim

$$\mathcal{S}_3 \cong \mathrm{Emb}(S^1, \mathbb{R}^2)/\operatorname{Diff}(S^1).$$

In contrast to the shape spaces described above, which are mainly of theoretical interest, let us now consider some finite-dimensional shape spaces. In [4], M. Kilian et al. study geometric modeling tasks, such as shape morphing and deformation transfer, using the shape space $\mathcal{S}_4$ of triangular meshes in $\mathbb{R}^3$ with a fixed connectivity graph and a given number of nodes. Clearly, $\mathcal{S}_4$ can be identified with $\mathbb{R}^{3N}$ where N denotes the number of nodes. Recall that the connectivity graph of a mesh in $\mathbb{R}^3$ is the graph which describes the neighbourhood relations of the nodes. The task is then to equip $\mathcal{S}_4$ with a useful Riemannian metric, which is in general different from the Euclidean inner product in $\mathbb{R}^{3N}$. The choice of the metric depends on the problem and the desired result. If a shape, i.e. a triangular mesh, should be deformed into a certain way but preserve all the pairwise distances between two points, then one will look for a Riemannian metric which strongly penalizes non-rigid deformations. In detail, the metric should yield a geodesic in $\mathcal{S}_4$, which consists of shapes being as-rigid-as-possible transformed. In a similar way, they introduce an as-isometric-as-possible metric

$$\langle X, Y\rangle^I := \sum_{(p,q)} \langle X_p - X_q, p - q\rangle \langle Y_p - Y_q, p - q\rangle$$

on $\mathcal{S}_4$. Here, $M \in \mathcal{S}_4$, $X, Y \in T_M\mathcal{S}_4$ and $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product in $\mathbb{R}^3$; moreover, the sum is taken over all edges (p, q) of the mesh M. Per definition, a deformation of a surface is isometric if and only if the distances measured *on* the surface are preserved during the deformation. For triangular meshes this is equivalent to the fact that the length of each edge remains constant. If there are no isometric deformations except translations and rotations, we have to deal with deformations being as-isometric-as-possible. And exactly these deformations yield shorter distances in $\mathcal{S}_4$, if one uses this metric. Consequently, the resulting geodesic joins shapes in $\mathcal{S}_4$ which are as-isometric-as-possible transformed; see [4] for further details. However, the as-rigid-as-possible and the as-isometric-as-possible metric are only Riemannian pseudo-metrics, since a rigid body motion, respectively an isometric deformation, has norm zero. To obtain a Riemannian metric, one may add a small regularization term like a multiple of an L^2-type metric

$$\langle X, Y\rangle^{L^2} := \sum_{p\in M} w_p \langle X_p, Y_p\rangle$$

where w_p denotes the area of the triangles adjacent to p. This is done in [4] and the result, which is obviously a Riemannian metric, then reads

$$\langle X, Y\rangle^I_\lambda := \langle X, Y\rangle^I + \lambda \langle X, Y\rangle^{L^2}$$

where $\lambda \in \mathbb{R}_{>0}$.

Furthermore, one could also consider the shape space $\mathcal{S}_5$ of all quad meshes with a fixed connectivity graph and a given number of nodes. But $\mathcal{S}_5$ is in some sense very similar to $\mathcal{S}_4$, since the only difference is the changed connectivity. Hence, the essential ingredients for a shape space of meshes (with a finite number of nodes) is the connectivity graph and the number of nodes, which are both the same for all meshes.

For our studies on shape optimization in the context of Riemannian geometry, we shall always consider the shape space $\mathcal{S}$ of triangulated meshes embedded in $\mathbb{R}^3$ with a fixed connectivity graph and a given number of nodes N. These surfaces may be either the boundary

$\partial\Omega$ of a subset $\Omega \subset \mathbb{R}^3$ with finite volume, or the graph of a function from $\mathbb{R}^2$ into $\mathbb{R}$. For a shape $M \in \mathcal{S}$ we will use $P \subset \mathbb{R}^3$ to denote the set of all nodes of M. For sure, M and P refer to the same object, the triangular mesh, but from different perspectives; on the one hand, M describes the mesh as an element in an abstract shape space, whereas P characterizes the mesh as a finite subset of $\mathbb{R}^3$. We also use the notation $\mathcal{N}(p)$ for the set containing p and all nodes of M which share a common edge with p, and $\mathcal{T}(p)$ for the set of all triangles of M which have p as a vertex. Moreover, we define $C \subset P^2$ as the set of all $(p, q) \in P^2$ which are neighbouring points.

We explained above that $\mathcal{S}$ can be identified with $\mathbb{R}^{3N}$. But depending on the problem, one will use special Riemannian metrics to endow the shape space $\mathcal{S}$ with a certain geometry. Subsequently, we will also introduce Riemannian metrics on $\mathcal{S}$ which are different from the Euclidean inner product in $\mathbb{R}^{3N}$. Strictly speaking, these metrics have to be defined on each tangent space $T_M\mathcal{S}$ for $M \in \mathcal{S}$; this will be done in the next section.

However, we will restrict our admissible deformations of a triangular mesh to those which are normal to the surface, in detail, every node $p \in P$ may only be moved along the local surface normal vector $n_p \in \mathbb{R}^3$. Consequently, our deformation fields are given by

$$(X_p)_{p\in P} = (\kappa_p n_p)_{p\in P} \in T_M\mathcal{S} \tag{2.5}$$

with $\kappa_p \in \mathbb{R}$. Since a deformation of a mesh $M \in \mathcal{S}$ is a curve in $\mathcal{S}$, we only consider curves in $\mathcal{S}$ whose tangent vectors are given by (2.5). Hence, we do not consider all possible tangent vectors in the $3N$-dimensional tangent space $T_M\mathcal{S}$ but only those described ones, which are contained in an N-dimensional subspace of $T_M\mathcal{S}$. Therefore, all the admissible deformations of a surface $M \in \mathcal{S}$ are uniquely determined by the vector $\vec{\kappa} := (\kappa_p)_{p\in P} \in \mathbb{R}^N$. Now, we have to define properly the normal vector $n_p \in \mathbb{R}^3$ of a triangulated mesh at a vertex $p \in P$. We decide to define it the following way:

$$n_p := \left\| \sum_{t\in\mathcal{T}(p)} (t_2 - p) \times (t_3 - p) \right\|^{-1} \sum_{t\in\mathcal{T}(p)} (t_2 - p) \times (t_3 - p) \tag{2.6}$$

where the vertices of all triangles in the mesh M are indexed in the same counterclockwise orientation, $p = t_1$ for all $t \in \mathcal{T}(p)$ and $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^3$. Note that (2.6) is a weighted average of the normal vectors to the triangles adjacent to p. In detail, the normal vectors of those triangles are more involved, whose area is large. This is the case, because we first sum over all normal vectors around p and then normalize the resulting vector. However, we could also have taken the sum of all normalized normal vectors and then normalize again but this would require more computational effort and ignore the area of the triangles around p. Below, we will often use the notation $\vec{n} := (n_p)_{p\in P} \in \mathbb{R}^{3N}$ for the concatenation of all normal *vectors* n_p with $p \in P$.

Let us now formulate the geodesic equation within this setting. For this case, we assume to have a Riemannian metric on the shape space $\mathcal{S}$; then, due to Theorem 2.20, we may consider the Levi-Civita connection ∇ with its coefficients $\Gamma^\gamma_{\alpha\beta}$ – the Christoffel symbols. Now, from Proposition 2.13, we see that we can easily reduce the geodesic equation to the following system

of first order differential equations:

$$\begin{cases} \dot{u}^\gamma &= T^\gamma, \\ \dot{T}^\gamma &= -\Gamma^\gamma_{\alpha\beta} T^\alpha T^\beta, \end{cases}$$

where $(u^\gamma)_{\gamma\in\{1,\dots,3N\}} = (p)_{p\in P}$ is just a different notation for the concatenation of all nodes $p \in P$ to a vector in $\mathbb{R}^{3N}$. Since we only allow deformations along the local surface normals, we also have $\dot{p} = \kappa_p n_p \in \mathbb{R}^3$ and, hence, $\vec{T} = \vec{\kappa} \cdot \vec{\vec{n}} \in \mathbb{R}^{3N}$ where $\cdot$ stands for the scalar multiplication of the corresponding entries of $\vec{\kappa}$ and $\vec{\vec{n}}$. In a similar way, one may rewrite the equation of parallel transport from Proposition 2.13 as

$$\dot{X}^\gamma = -\Gamma^\gamma_{\alpha\beta} X^\alpha T^\beta$$

where $\vec{X}$ is the parallel translate of an initial vector $\vec{X}_0$, tangent to the shape space $\mathcal{S}$, along a geodesic u with tangent vector $\vec{T}$. Again, we are only interested in tangent vectors $\vec{X}$ which are locally given by $X_p = \lambda_p n_p$ and, therefore, read $\vec{X} = \vec{\lambda} \cdot \vec{\vec{n}} \in \mathbb{R}^{3N}$. However, it is not clear up to now that the geodesic equation or the equation of parallel translation admits solutions, for which $\vec{T} = \vec{\kappa} \cdot \vec{\vec{n}}$, respectively $\vec{X} = \vec{\lambda} \cdot \vec{\vec{n}}$ holds. For sure, we know from the Theorem of Picard and Lindelöf, that both differential equations have unique solutions at least within a sufficiently small time interval I. But we do not know whether this solution is in these N-dimensional subspaces of $T_{M(t)}\mathcal{S}$ for all $t \in I$ provided this is true for the initial data. Nevertheless, we shall see in the next sections that we are able to show the unique existence of such solutions via deducing an explicit formula for $\dot{\kappa}$, respectively $\dot{\lambda}$. This works at least for those metrics which we consider, if we accept a slight approximation at some point.

2.2 Riemannian Metrics

We now introduce some Riemannian metrics on the tangent space $T_M\mathcal{S}$ to the space of triangular meshes $\mathcal{S}$ in some shape $M \in \mathcal{S}$. We start with the definition of the metric and deduce the inner product of two canonical basis vectors of $\mathbb{R}^{3N}$.

Let us begin with the Euclidean metric

$$\langle X, Y\rangle^{Eu} := \sum_{p\in P} \langle x_p, y_p\rangle$$

where $X = (x_p)_{p\in P}, Y = (y_p)_{p\in P} \in T_M\mathcal{S}$ with $x_p, y_p \in \mathbb{R}^3$. So far, the metric is defined on the whole tangent space at $M \in \mathcal{S}$. Specifically for $N_1 = (\kappa_p n_p)_{p\in P}, N_2 = (\lambda_p n_p)_{p\in P} \in T_M\mathcal{S}$ we obtain

$$\langle N_1, N_2\rangle^{Eu} = \sum_{p\in P} \kappa_p \lambda_p.$$

Obviously, this is a very simple metric for vectors which consist of local normal vectors. Let now $e^i_p \in \mathbb{R}^{3N}$ be the vector having zeros in all entries except in the i-th component of the part corresponding to $p \in M$. Then, one immediately sees from the definition of the metric that

$$\langle e^i_p, e^j_q\rangle^{Eu} = \delta_{pq}\delta_{ij}$$

where $\delta_{\cdot,\cdot}$ denotes the Kronecker-Delta. In addition, it is obvious that $\langle\cdot,\cdot\rangle^{Eu}$ actually defines a Riemannian metric (not only a pseudo metric) on the shape space $\mathcal{S}$ and, consequently, no regularization is necessary.

Another possible Riemannian metric on the shape space $\mathcal{S}$ is the H^n-type metric

$$\langle X, Y\rangle_0^{H^n} := \sum_{(p,q)\in C} \left\langle \frac{x_p - x_q}{\|p-q\|^n}, p-q \right\rangle \left\langle \frac{y_p - y_q}{\|p-q\|^n}, p-q \right\rangle$$

for $n \in \mathbb{N}$, $X = (x_p)_{p\in P}, Y = (y_p)_{p\in P} \in T_M\mathcal{S}$ and vectors $x_p, y_p \in \mathbb{R}^3$. We shall emphasize that the notation is motivated by Hölder type estimates, and not by Sobolev spaces. Since the right-hand-side only defines a Riemannian pseudo-metric, we write $\langle\cdot,\cdot\rangle_0^{H^n}$ and define the H^n-metric via the following regularization of $\langle\cdot,\cdot\rangle_0^{H^n}$ with $\rho \in \mathbb{R}_{>0}$:

$$\langle X, Y\rangle^{H^n} := \sum_{(p,q)\in C} \left\langle \frac{x_p - x_q}{\|p-q\|^n}, p-q \right\rangle \left\langle \frac{y_p - y_q}{\|p-q\|^n}, p-q \right\rangle + \rho \sum_{p\in P} \langle x_p, y_p\rangle.$$

A special case in this general definition is $n = 0$; this is the so-called as-isometric-as-possible metric, which has already been discussed in subsection 2.1.3. However, the idea behind the H^n-metric is to adopt the as-isometric-as-possible metric in such a way that small distances between two points, and therefore also nearly singular triangles, are penalized more. But although this inner product is able to prevent the local contraction of several points to one point, it is still possible that self-intersections of the surface occur in the process of deforming. This is not surprisung since the metric only takes the distances from one point to each of its neighbours into account and two not neighbouring points may still become arbitrary close to each other. Now, we are interested in deformations of a shape along its local surface normals, hence, we also state the special form of the metric

$$\langle N_1, N_2\rangle^{H^n} = \sum_{(p,q)\in C} \left\langle \frac{\kappa_p n_p - \kappa_q n_q}{\|p-q\|^n}, p-q \right\rangle \left\langle \frac{\lambda_p n_p - \lambda_q n_q}{\|p-q\|^n}, p-q \right\rangle + \rho \sum_{p\in P} \kappa_p \lambda_p$$

with $N_1 = (\kappa_p n_p)_{p\in P}, N_2 = (\lambda_p n_p)_{p\in P} \in T_M\mathcal{S}$ and n_p the local surface normal at the point $p \in M$. To obtain the inner product of two canonical basis vectors $e_p^i, e_q^j \in \mathbb{R}^{3N}$ with $p, q \in M$ and $i \in \{1,2,3\}$, we have to distinguish whether the two vectors have their non-zero entry at the same point p or at different points p and q. From the definition of the H^n-metric, it is clear that $\langle e_p^i, e_q^j\rangle^{H^n} = 0$ if p and q are not neighbouring points. Thus, the two cases $q = p$ and $q \in \mathcal{N}(p)\backslash\{p\}$ remain, where $\mathcal{N}(p)$ is the set of neighbouring points of p. The result directly follows from the definition and is given by

$$\left\langle e_p^i, e_p^j \right\rangle^{H^n} = \rho\delta_{ij} + \sum_{q\in\mathcal{N}(p)\backslash\{p\}} \frac{(p^i - q^i)(p^j - q^j)}{\|p-q\|^{2n}}$$

and

$$\left\langle e_p^i, e_q^j \right\rangle^{H^n} = -\frac{(p^i - q^i)(p^j - q^j)}{\|p-q\|^{2n}} \quad \text{for } q \in \mathcal{N}(p)\backslash\{p\}.$$

For our purposes the Riemannian metrics introduced above are sufficient, since we will also have a closer look at the differences between these metrics in applications.

2.3 Geodesic Equations

Now, we will establish the systems of geodesic equations which correspond to the above metrics. For this reason, we manipulate the generic geodesic equation in such a way to get rid of the Christoffel symbols, but we will see that for the H^n-metric we have to make an approximation at a certain point. Consequently, the resulting approximate geodesic equation is quite cheap to evalute but, in general, it is not exact any more.

We start with a triangular mesh $M \in \mathcal{S}$ which has N nodes in $\mathbb{R}^3$; the collection of these nodes is denoted with $P \subset \mathbb{R}^3$. In subsection 2.1.3 we stated

$$\begin{cases} \dot{u}^\gamma &= T^\gamma, \\ \dot{T}^\gamma &= -\Gamma^\gamma_{\alpha\beta} T^\alpha T^\beta, \end{cases}$$

where M is characterized with a vector $u \in \mathbb{R}^{3N}$. In addition,

$$\vec{T} = \vec{\kappa} \cdot \vec{\vec{n}} \in \mathbb{R}^{3N} \quad \text{and} \quad \dot{\vec{T}} = \dot{\vec{\kappa}} \cdot \vec{\vec{n}} + \vec{\kappa} \cdot \dot{\vec{\vec{n}}} \in \mathbb{R}^{3N}$$

where $\vec{\kappa} \in \mathbb{R}^N$, $\vec{\vec{n}} \in \mathbb{R}^{3N}$ and $\cdot$ stands for the scalar multiplication of the corresponding entries of $\vec{\kappa}$ and $\vec{\vec{n}}$. The aim of this section is to derive a formula for each $\dot{\kappa}_p$ with $p \in P$ using the system of geodesic equations.

Independent from the chosen metric, we can do the following calculation. Let $\langle \cdot, \cdot \rangle^{\mathcal{S}}$ denote one of the Riemannian metrics on $\mathcal{S}$ defined above and

$$\tilde{n}_p := \left(0^T_{\mathbb{R}^3}, ..., 0^T_{\mathbb{R}^3}, n_p^T, 0^T_{\mathbb{R}^3}, ..., 0^T_{\mathbb{R}^3}\right)^T \in \mathbb{R}^{3N}$$

be a vector with all entries zero except at those components which correspond to the point $p \in P$. Then we get

$$\begin{aligned} \left\langle \dot{\vec{\kappa}} \cdot \vec{\vec{n}} + \vec{\kappa} \cdot \dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{\mathcal{S}} &= \left\langle -e_\gamma \Gamma^\gamma_{\alpha\beta} T^\alpha T^\beta, \tilde{n}_p \right\rangle^{\mathcal{S}} = \left\langle -e_\gamma \Gamma^\gamma_{\alpha\beta} T^\alpha T^\beta, e_\delta \tilde{n}_p^\delta \right\rangle^{\mathcal{S}} \\ &= -\langle e_\gamma, e_\delta \rangle^{\mathcal{S}} \Gamma^\gamma_{\alpha\beta} T^\alpha T^\beta \tilde{n}_p^\delta = -\Gamma^\gamma_{\alpha\beta} g_{\gamma\delta} T^\alpha T^\beta \tilde{n}_p^\delta \\ &= -\frac{1}{2} \left(\frac{\partial g_{\delta\alpha}}{\partial u^\beta} + \frac{\partial g_{\beta\delta}}{\partial u^\alpha} - \frac{\partial g_{\alpha\beta}}{\partial u^\delta} \right) T^\alpha T^\beta \tilde{n}_p^\delta \\ &= -\frac{1}{2} \sum_{k=1}^{3} n_p^k \sum_{q,r \in \mathcal{N}(p)} \sum_{l,m=1}^{3} T_q^l T_r^m \underbrace{\left(\frac{\partial \langle e_p^k, e_q^l \rangle}{\partial r^m} + \frac{\partial \langle e_r^m, e_p^k \rangle}{\partial q^l} - \frac{\partial \langle e_q^l, e_r^m \rangle}{\partial p^k} \right)}_{=: A} \end{aligned}$$

where the last equality holds true since both metrics, $\langle \cdot, \cdot \rangle^{Eu}$ and $\langle \cdot, \cdot \rangle^{H^n}$ are local in the sense that they satisfy $\langle e_s^i, e_t^j \rangle = 0$ for all $i, j \in \{1, 2, 3\}$ if $s, t \in P$ are not neighbouring points. The task is now to calculate $A = A(p, q, r, k, l, m)$ for the different Riemannian metrics.

At first, we consider the metric $\langle \cdot, \cdot \rangle^{Eu}$. In this case, $A = 0$ since $\langle e_p^i, e_q^j \rangle^{Eu} = \delta_{pq} \delta_{ij}$ is a constant independent of the coordinates of p and q. Moreover,

$$\left\langle \dot{\vec{\kappa}} \cdot \vec{\vec{n}} + \vec{\kappa} \cdot \dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{Eu} = \dot{\kappa_p} \langle n_p, n_p \rangle + \kappa_p \langle \dot{n_p}, n_p \rangle = \dot{\kappa_p}$$

and therefore, the system of geodesic equations for the metric $\langle\cdot,\cdot\rangle^{Eu}$ reads

$$\begin{cases} \dot{p} &= \kappa_p n_p, \\ \dot{\kappa_p} &= 0, \end{cases} \tag{2.7}$$

for all $p \in P$. This pair of equations describes the deformation of a surface along its local normal vectors where κ, the speed of the deformation, remains constant.

Next, we come to the more elaborate calculation of A for the H^n-metric. For this reason, we distinguish four cases where the calculations are quite trivial but, nevertheless, require some concentration. So let $p \in P$, $q, r \in \mathcal{N}(p)$ and $k, l, m \in \{1, 2, 3\}$.

Case 1: $q = p \ \wedge \ r = p$**:**

$$\begin{aligned} A &= \delta_{km} \sum_{s\in\mathcal{N}(p)\setminus\{p\}} \frac{p^l - s^l}{\|p-s\|^{2n}} + \delta_{lm}\frac{p^k - s^k}{\|p-s\|^{2n}} - 2n\frac{(p^k - s^k)(p^l - s^l)(p^m - s^m)}{\|p-s\|^{2n+2}} + \\ &\quad \delta_{ml} \sum_{s\in\mathcal{N}(p)\setminus\{p\}} \frac{p^k - s^k}{\|p-s\|^{2n}} + \delta_{kl}\frac{p^m - s^m}{\|p-s\|^{2n}} - 2n\frac{(p^m - s^m)(p^k - s^k)(p^l - s^l)}{\|p-s\|^{2n+2}} - \\ &\quad \delta_{lk} \sum_{s\in\mathcal{N}(p)\setminus\{p\}} \frac{p^m - s^m}{\|p-s\|^{2n}} - \delta_{mk}\frac{p^l - s^l}{\|p-s\|^{2n}} + 2n\frac{(p^l - s^l)(p^m - s^m)(p^k - s^k)}{\|p-s\|^{2n+2}} \\ &= 2 \sum_{s\in\mathcal{N}(p)\setminus\{p\}} \left(\delta_{lm}\frac{p^k - s^k}{\|p-s\|^{2n}} - n\frac{(p^k - s^k)(p^l - s^l)(p^m - s^m)}{\|p-s\|^{2n+2}}\right). \end{aligned}$$

Case 2: $q = p \ \wedge \ r \neq p$**:**

$$\begin{aligned} A &= -\delta_{km}\frac{p^l - r^l}{\|p-r\|^{2n}} - \delta_{lm}\frac{p^k - r^k}{\|p-r\|^{2n}} + 2n\frac{(p^k - r^k)(p^l - r^l)(p^m - r^m)}{\|p-r\|^{2n+2}} - \\ &\quad \delta_{kl}\frac{p^m - r^m}{\|p-r\|^{2n}} - \delta_{ml}\frac{p^k - r^k}{\|p-r\|^{2n}} + 2n\frac{(p^k - r^k)(p^m - r^m)(p^l - r^l)}{\|p-r\|^{2n+2}} + \\ &\quad \delta_{lk}\frac{p^m - r^m}{\|p-r\|^{2n}} + \delta_{mk}\frac{p^l - r^l}{\|p-r\|^{2n}} - 2n\frac{(p^l - r^l)(p^m - r^m)(p^k - r^k)}{\|p-r\|^{2n+2}} \\ &= -2\left(\delta_{lm}\frac{p^k - r^k}{\|p-r\|^{2n}} - n\frac{(p^k - r^k)(p^l - r^l)(p^m - r^m)}{\|p-r\|^{2n+2}}\right). \end{aligned}$$

Case 3: $q \neq p \ \wedge \ r = p$**:**

$$\begin{aligned} A &= -\delta_{km}\frac{p^l - q^l}{\|p-q\|^{2n}} - \delta_{lm}\frac{p^k - q^k}{\|p-q\|^{2n}} + 2n\frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}} - \\ &\quad \delta_{kl}\frac{p^m - q^m}{\|p-q\|^{2n}} - \delta_{ml}\frac{p^k - q^k}{\|p-q\|^{2n}} + 2n\frac{(p^k-q^k)(p^m-q^m)(p^l-q^l)}{\|p-q\|^{2n+2}} + \\ &\quad \delta_{mk}\frac{p^l - q^l}{\|p-q\|^{2n}} + \delta_{lk}\frac{p^m - q^m}{\|p-q\|^{2n}} - 2n\frac{(p^m-q^m)(p^l-q^l)(p^k-q^k)}{\|p-q\|^{2n+2}} \\ &= -2\left(\delta_{lm}\frac{p^k - q^k}{\|p-q\|^{2n}} - n\frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}}\right). \end{aligned}$$

Case 4: $q \neq p \ \wedge \ r \neq p$**:** Now we consider the following two cases.

Case 4.1: $q = r$**:**

$$\begin{aligned} A &= \delta_{km}\frac{p^l - q^l}{\|p-q\|^{2n}} + \delta_{lm}\frac{p^k - q^k}{\|p-q\|^{2n}} - 2n\frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}} + \\ &\quad \delta_{kl}\frac{p^m - q^m}{\|p-q\|^{2n}} + \delta_{ml}\frac{p^k - q^k}{\|p-q\|^{2n}} - 2n\frac{(p^k-q^k)(p^m-q^m)(p^l-q^l)}{\|p-q\|^{2n+2}} - \\ &\quad \delta_{lk}\frac{p^m - q^m}{\|p-q\|^{2n}} - \delta_{mk}\frac{p^l - q^l}{\|p-q\|^{2n}} + 2n\frac{(p^l-q^l)(p^m-q^m)(p^k-q^k)}{\|p-q\|^{2n+2}} \\ &= 2\left(\delta_{lm}\frac{p^k - q^k}{\|p-q\|^{2n}} - n\frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}}\right). \end{aligned}$$

Case 4.2: $q \neq r$**:**

$$A = 0 + 0 - 0 = 0.$$

Now, we use these expressions for A and find

$$
\begin{aligned}
& -\frac{1}{2}\sum_{k=1}^{3} n_p^k \sum_{q,r\in\mathcal{N}(p)} \sum_{l,m=1}^{3} T_q^l T_r^m A(p,q,r,k,l,m) \\
= & -\sum_{k,l=1}^{3} n_p^k \sum_{s\in\mathcal{N}(p)\setminus\{p\}} \left(T_p^l T_p^l \frac{p^k - s^k}{\|p-s\|^{2n}} - n\sum_{m=1}^{3} T_p^l T_p^m \frac{(p^k-s^k)(p^l-s^l)(p^m-s^m)}{\|p-s\|^{2n+2}} \right) + \\
& \sum_{k,l=1}^{3} n_p^k \sum_{r\in\mathcal{N}(p)\setminus\{p\}} \left(T_p^l T_r^l \frac{p^k - r^k}{\|p-r\|^{2n}} - n\sum_{m=1}^{3} T_p^l T_r^m \frac{(p^k-r^k)(p^l-r^l)(p^m-r^m)}{\|p-r\|^{2n+2}} \right) + \\
& \sum_{k,l=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \left(T_q^l T_p^l \frac{p^k - q^k}{\|p-q\|^{2n}} - n\sum_{m=1}^{3} T_q^l T_p^m \frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}} \right) - \\
& \sum_{k,l=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \left(T_q^l T_q^l \frac{p^k - q^k}{\|p-q\|^{2n}} - n\sum_{m=1}^{3} T_q^l T_q^m \frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}} \right) \\
= & \sum_{k,l=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p^k - q^k}{\|p-q\|^{2n}} \Bigg(-\left(T_p^l\right)^2 + 2T_p^l T_q^l - \left(T_q^l\right)^2 + \\
& n\frac{p^l-q^l}{\|p-q\|^2} \sum_{m=1}^{3} (p^m-q^m) \left(T_p^l T_p^m - T_p^l T_q^m + T_q^l T_q^m - T_q^l T_p^m \right) \Bigg) \\
= & \sum_{k=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p^k - q^k}{\|p-q\|^{2n}} \Bigg(\sum_{l=1}^{3} -\left(T_p^l - T_q^l\right)^2 + \\
& n\sum_{l=1}^{3} \frac{p^l-q^l}{\|p-q\|^2} \sum_{m=1}^{3} (p^m-q^m) \left(T_p^l - T_q^l\right)\left(T_p^m - T_q^m\right) \Bigg) \\
= & \sum_{k=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p^k - q^k}{\|p-q\|^{2n}} \left(-\|T_p - T_q\|^2 + n\frac{\langle p-q, T_p - T_q\rangle^2}{\|p-q\|^2} \right) \\
= & \left\langle n_p, \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p-q}{\|p-q\|^{2n}} \left(n\frac{\langle p-q, T_p - T_q\rangle^2}{\|p-q\|^2} - \|T_p - T_q\|^2 \right) \right\rangle
\end{aligned}
$$

where $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product in $\mathbb{R}^3$. The last step to obtain the geodesic equations in the shape space $\mathcal{S}$ for the H^n-metric is to manipulate the left-hand-side of the equation

$$
\left\langle \dot{\vec{\kappa}}\cdot\vec{\vec{n}} + \vec{\kappa}\cdot\dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{H^n} = \left\langle n_p, \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p-q}{\|p-q\|^{2n}} \left(n\frac{\langle p-q, T_p - T_q\rangle^2}{\|p-q\|^2} - \|T_p - T_q\|^2 \right) \right\rangle
$$

in such a way to get an explicit formula for $\dot{\kappa}_p$. But to achieve this we have to make an

approximation at some point.

$$
\begin{aligned}
&\left\langle \dot{\vec{\kappa}} \cdot \vec{\vec{n}} + \vec{\kappa} \cdot \dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{H^n} \\
= &\left\langle \sum_{q\in P} \sum_{i=1}^{3} \left(\dot{\kappa}_q n_q^i + \kappa_q \dot{n}_q^i\right) e_q^i, \sum_{j=1}^{3} n_p^j e_p^j \right\rangle^{H^n} = \sum_{q\in P} \sum_{i,j=1}^{3} \left(\dot{\kappa}_q n_q^i + \kappa_q \dot{n}_q^i\right) n_p^j \left\langle e_q^i, e_p^j \right\rangle^{H^n} \\
= &\sum_{i,j=1}^{3} \left(\dot{\kappa}_p n_p^i + \kappa_p \dot{n}_p^i\right) n_p^j \left(\rho\delta_{ij} + \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{(p^i - q^i)(p^j - q^j)}{\|p-q\|^{2n}} \right) - \\
&\sum_{q\in\mathcal{N}(p)\setminus\{p\}} \sum_{i,j=1}^{3} \left(\dot{\kappa}_q n_q^i + \kappa_q \dot{n}_q^i\right) n_p^j \frac{(p^i - q^i)(p^j - q^j)}{\|p-q\|^{2n}} \\
= &\rho\left(\dot{\kappa}_p \langle n_p, n_p\rangle + \kappa_p \langle \dot{n}_p, n_p\rangle\right) + \dot{\kappa}_p \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \sum_{i,j=1}^{3} n_p^i n_p^j (p^i - q^i)(p^j - q^j) + \\
&\kappa_p \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \sum_{i,j=1}^{3} \dot{n}_p^i n_p^j (p^i - q^i)(p^j - q^j) - \\
&\sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{\dot{\kappa}_q}{\|p-q\|^{2n}} \sum_{i,j=1}^{3} n_q^i n_p^j (p^i - q^i)(p^j - q^j) - \\
&\sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{\kappa_q}{\|p-q\|^{2n}} \sum_{i,j=1}^{3} \dot{n}_q^i n_p^j (p^i - q^i)(p^j - q^j)
\end{aligned}
$$

In order to get rid of the $\dot{\kappa}_q$, we do the following approximation:

$$\dot{\kappa}_q \approx \dot{\kappa}_p \quad \text{for all} \quad q \in \mathcal{N}(p)\setminus\{p\}. \tag{2.8}$$

Otherwise, we had to solve a band-structured linear system for $\dot{\vec{\kappa}}$ provided the system admits a unique solution; but this might not be the case. However, we use a lumping method and concentrate all coefficients of the system on its main diagonal. If we use that n_p is normalized and consequently $\langle n_p, n_p\rangle = 1$ and $\langle \dot{n}_p, n_p\rangle = 0$ for all $p \in P$, we can further simplify the above expressions to find

$$
\begin{aligned}
&\left\langle \dot{\vec{\kappa}} \cdot \vec{\vec{n}} + \vec{\kappa} \cdot \dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{H^n} \\
= &\rho\dot{\kappa}_p + \dot{\kappa}_p \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \sum_{i,j=1}^{3} (n_p^i - n_q^i) n_p^j (p^i - q^i)(p^j - q^j) + \\
&\sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \sum_{i,j=1}^{3} (\kappa_p \dot{n}_p^i - \kappa_q \dot{n}_q^i) n_p^j (p^i - q^i)(p^j - q^j) \\
= &\rho\dot{\kappa}_p + \dot{\kappa}_p \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \langle n_p - n_q, p-q\rangle \langle n_p, p-q\rangle + \\
&\sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \langle \kappa_p \dot{n}_p - \kappa_q \dot{n}_q, p-q\rangle \langle n_p, p-q\rangle.
\end{aligned}
$$

Now, we deduce

$$\dot{\kappa}_p \left(\rho + \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q \rangle \right\rangle \right) =$$
$$\left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \left(n \frac{\langle p-q, T_p - T_q \rangle^2}{\|p-q\|^2} - \langle \kappa_p \dot{n}_p - \kappa_q \dot{n}_q, p-q \rangle - \|T_p - T_q\|^2 \right) \right\rangle$$

and in the case that the left-hand-side is not zero – which is the case if $\rho \in \mathbb{R}_{>0}$ is sufficiently large – we have

$$\dot{\kappa}_p = \frac{\left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \left(n \frac{\langle p-q, T_p - T_q \rangle^2}{\|p-q\|^2} - \langle \kappa_p \dot{n}_p - \kappa_q \dot{n}_q, p-q \rangle - \|T_p - T_q\|^2 \right) \right\rangle}{\rho + \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q \rangle \right\rangle}.$$

Finally, we are able to state the system of (approximate) geodesic equations in the shape space $\mathcal{S}$ for the H^n-metric, which is given by

$$\begin{cases} \dot{p} &= \kappa_p n_p, \\ \dot{\kappa}_p &= \dfrac{\left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \left(n \frac{\langle p-q, T_p - T_q \rangle^2}{\|p-q\|^2} - \langle \kappa_p \dot{n}_p - \kappa_q \dot{n}_q, p-q \rangle - \|T_p - T_q\|^2 \right) \right\rangle}{\rho + \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q \rangle \right\rangle} \end{cases} \tag{2.9}$$

for all $p \in P$.

2.4 Equations of Parallel Transport

The next task is to derive an explicit representation of the equations of parallel translation. For this case, let $u(t)$ be a geodesic in the shape space $\mathcal{S}$, $T(t)$ be the field of tangent vectors to the geodesic, $M \in \mathcal{S}$ a triangular mesh with N nodes and $P \subset \mathbb{R}^3$ the set of all nodes of M. Then we know from subsection 2.1.3 that

$$\dot{X}^\gamma = -\Gamma^\gamma_{\alpha\beta} X^\alpha T^\beta$$

defines the parallel translate X along the geodesic u of an initial vector X_0 tangent to $\mathcal{S}$. Here,

$\vec{T} = \vec{\kappa} \cdot \vec{\vec{n}} \in \mathbb{R}^{3N}$,

$$\vec{X} = \vec{\lambda} \cdot \vec{\vec{n}} \in \mathbb{R}^{3N} \quad \text{and} \quad \dot{\vec{X}} = \dot{\vec{\lambda}} \cdot \vec{\vec{n}} + \vec{\lambda} \cdot \dot{\vec{\vec{n}}} \in \mathbb{R}^{3N}$$

where $\vec{\kappa}, \vec{\lambda} \in \mathbb{R}^N$ and $\vec{\vec{n}} \in \mathbb{R}^{3N}$. Again, $\cdot$ stands for the scalar multiplication of the corresponding entries of $\vec{\kappa}$ respectively $\vec{\lambda}$ and $\vec{\vec{n}}$. The following steps towards a formula for each $\dot{\lambda}_p$ with $p \in P$ are at some points quite similar to the calculation of the geodesic equations, hence, we will discuss analogous manipulations only briefly.

Let $\langle \cdot, \cdot \rangle^{\mathcal{S}}$ denote one of the Riemannian metrics on $\mathcal{S}$ defined above, and let

$$\tilde{n}_p := \left(0^T_{\mathbb{R}^3}, ..., 0^T_{\mathbb{R}^3}, n_p^T, 0^T_{\mathbb{R}^3}, ..., 0^T_{\mathbb{R}^3}\right)^T \in \mathbb{R}^{3N}$$

be a vector with nontrivial entries only at those components which correspond to the point $p \in P$. Then one finds

$$\begin{aligned}
\left\langle \dot{\vec{\lambda}} \cdot \vec{\vec{n}} + \vec{\lambda} \cdot \dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{\mathcal{S}} &= \left\langle -e_\gamma \Gamma^\gamma_{\alpha\beta} X^\alpha T^\beta, \tilde{n}_p \right\rangle^{\mathcal{S}} = \left\langle -e_\gamma \Gamma^\gamma_{\alpha\beta} X^\alpha T^\beta, e_\delta \tilde{n}_p^\delta \right\rangle^{\mathcal{S}} \\
&= -\langle e_\gamma, e_\delta \rangle^{\mathcal{S}} \Gamma^\gamma_{\alpha\beta} X^\alpha T^\beta \tilde{n}_p^\delta = -\Gamma^\gamma_{\alpha\beta} g_{\gamma\delta} X^\alpha T^\beta \tilde{n}_p^\delta \\
&= -\frac{1}{2} \left(\frac{\partial g_{\delta\alpha}}{\partial u^\beta} + \frac{\partial g_{\beta\delta}}{\partial u^\alpha} - \frac{\partial g_{\alpha\beta}}{\partial u^\delta} \right) X^\alpha T^\beta \tilde{n}_p^\delta \\
&= -\frac{1}{2} \sum_{k=1}^{3} n_p^k \sum_{q,r \in \mathcal{N}(p)} \sum_{l,m=1}^{3} X_q^l T_r^m \underbrace{\left(\frac{\partial \langle e_p^k, e_q^l \rangle}{\partial r^m} + \frac{\partial \langle e_r^m, e_p^k \rangle}{\partial q^l} - \frac{\partial \langle e_q^l, e_r^m \rangle}{\partial p^k} \right)}_{=: A},
\end{aligned}$$

where $A = A(p, q, r, k, l, m)$ has already been calculated for the metrics $\langle \cdot, \cdot \rangle^{Eu}$ and $\langle \cdot, \cdot \rangle^{H^n}$.

Since $A = 0$ for the metric $\langle \cdot, \cdot \rangle^{Eu}$, we immediately get the equation of parallel translation for this metric,

$$\dot{\lambda}_p = \left\langle \dot{\vec{\lambda}} \cdot \vec{\vec{n}} + \vec{\lambda} \cdot \dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{Eu} = 0 \tag{2.10}$$

for all $p \in P$. Consequently, $\lambda_p(t)$ is constant for every $p \in P$ and $\vec{X}(t) = \vec{\lambda} \cdot \vec{\vec{n}}(t)$ only depends on all the normal vectors n_p for $p \in P$.

For the H^n-metric, we proceed along the same lines as for the geodesic equations, which

results in

$$
\begin{aligned}
& -\frac{1}{2}\sum_{k=1}^{3} n_p^k \sum_{q,r\in\mathcal{N}(p)} \sum_{l,m=1}^{3} X_q^l T_r^m A(p,q,r,k,l,m) \\
= & -\sum_{k,l=1}^{3} n_p^k \sum_{s\in\mathcal{N}(p)\setminus\{p\}} \left(X_p^l T_p^l \frac{p^k - s^k}{\|p-s\|^{2n}} - n\sum_{m=1}^{3} X_p^l T_p^m \frac{(p^k-s^k)(p^l-s^l)(p^m-s^m)}{\|p-s\|^{2n+2}} \right) + \\
& \sum_{k,l=1}^{3} n_p^k \sum_{r\in\mathcal{N}(p)\setminus\{p\}} \left(X_p^l T_r^l \frac{p^k - r^k}{\|p-r\|^{2n}} - n\sum_{m=1}^{3} X_p^l T_r^m \frac{(p^k-r^k)(p^l-r^l)(p^m-r^m)}{\|p-r\|^{2n+2}} \right) + \\
& \sum_{k,l=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \left(X_q^l T_p^l \frac{p^k - q^k}{\|p-q\|^{2n}} - n\sum_{m=1}^{3} X_q^l T_p^m \frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}} \right) - \\
& \sum_{k,l=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \left(X_q^l T_q^l \frac{p^k - q^k}{\|p-q\|^{2n}} - n\sum_{m=1}^{3} X_q^l T_q^m \frac{(p^k-q^k)(p^l-q^l)(p^m-q^m)}{\|p-q\|^{2n+2}} \right) \\
= & \sum_{k,l=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p^k-q^k}{\|p-q\|^{2n}} \Bigg(-\left(X_p^l - X_q^l\right)\left(T_p^l - T_q^l\right) + \\
& n\frac{p^l-q^l}{\|p-q\|^2} \sum_{m=1}^{3} (p^m-q^m)\left(X_p^l - X_q^l\right)\left(T_p^m - T_q^m\right) \Bigg) \\
= & \sum_{k=1}^{3} n_p^k \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p^k-q^k}{\|p-q\|^{2n}} \left(-\langle X_p - X_q, T_p - T_q\rangle + n\frac{\langle X_p - X_q, p-q\rangle\langle T_p - T_q, p-q\rangle}{\|p-q\|^2} \right) \\
= & \left\langle n_p, \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{p-q}{\|p-q\|^{2n}} \left(n\frac{\langle X_p - X_q, p-q\rangle\langle T_p - T_q, p-q\rangle}{\|p-q\|^2} - \langle X_p - X_q, T_p - T_q\rangle \right) \right\rangle
\end{aligned}
$$

where $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product in $\mathbb{R}^3$. Similarly to the considerations above, we have to isolate $\dot{\vec{\lambda}}$. We use the approximation from equation (2.8) and get

$$
\begin{aligned}
\left\langle \dot{\vec{\lambda}}\cdot\vec{\vec{n}} + \vec{\lambda}\cdot\dot{\vec{\vec{n}}}, \tilde{n}_p \right\rangle^{H^n} = & \rho\dot{\lambda}_p + \dot{\lambda}_p \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \langle n_p - n_q, p-q\rangle\langle n_p, p-q\rangle + \\
& \sum_{q\in\mathcal{N}(p)\setminus\{p\}} \frac{1}{\|p-q\|^{2n}} \langle \lambda_p\dot{n}_p - \lambda_q\dot{n}_q, p-q\rangle\langle n_p, p-q\rangle.
\end{aligned}
$$

Analogously, one finds

$$\dot{\lambda}_p \left(\rho + \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q \rangle \right\rangle \right) = \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \right.$$
$$\left. \left(n \frac{\langle X_p - X_q, p-q \rangle \langle T_p - T_q, p-q \rangle}{\|p-q\|^2} - \langle \lambda_p \dot{n}_p - \lambda_q \dot{n}_q, p-q \rangle - \langle X_p - X_q, T_p - T_q \rangle \right) \right\rangle$$

and if $\rho \in \mathbb{R}_{>0}$ is large enough, we arrive at

$$\dot{\lambda}_p = \left(\rho + \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q \rangle \right\rangle \right)^{-1} \left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \right.$$
$$\left. \left(n \frac{\langle X_p - X_q, p-q \rangle \langle T_p - T_q, p-q \rangle}{\|p-q\|^2} - \langle \lambda_p \dot{n}_p - \lambda_q \dot{n}_q, p-q \rangle - \langle X_p - X_q, T_p - T_q \rangle \right) \right\rangle \tag{2.11}$$

for all $p \in P$. All together, the parallel translate $X(t) = \vec{\lambda}(t) \cdot \vec{\vec{n}}(t) \in T_{M(t)}\mathcal{S}$ of a tangent vector $X_0 = \vec{\lambda}_0 \cdot \vec{\vec{n}}(0) \in T_M\mathcal{S}$ along a geodesic u in the shape space $\mathcal{S}$ can be calculated from $\vec{\lambda}_0$ together with the above equations for $\dot{\vec{\lambda}}(t)$.

Chapter 3

Application To Shape From Shading

The basic idea behind Shape From Shading (SFS) is the following: Given a shading image of a surface, i.e. an image of the surface which is illuminated in a certain way, we want to reconstruct this surface. Generally, the shading image is given as a gray level image and the underlying surface is interpreted as the graph of a function $F : D \subset \mathbb{R}^2 \to \mathbb{R}$ where $D \subset \mathbb{R}^2$ stands for the set where the shading values of the surface are prescribed. Moreover, we may choose a coordinate system (x, y, z) of $\mathbb{R}^3$ such that the direction of the observer coincides with the negative z-direction.

The first method to reconstruct the surface structure from a shading image was presented by Horn in [2]. His idea is to determine several paths on the surface which start from a set of points $(x, y, F(x, y))$ where $F(x, y)$, $F_x(x, y)$ and $F_y(x, y)$ are given. These paths are called characteristics. In order to get an impression of the resulting shape, one may calculate various characteristics which are close enough to each other. In detail, Horn suggested to choose a small curve around a local maximum or minimum of F; this curve serves then as the set of initial points for the characteristics. Around the extremal point the surface can be approximated with a concave or convex parabola. Hence, we can calculate $F(x, y)$, $F_x(x, y)$ and $F_y(x, y)$ approximately, if we can estimate the curvature of the surface at the extremal point. Finally, one arrives at a system of five ordinary differential equations for x, y, z, p and q where $z \equiv F(x, y)$, $p \equiv F_x(x, y)$ and $q \equiv F_y(x, y)$. However, it may happen that the resulting characteristics are restricted to a certain region on the surface. Generally, these regions are bounded by various types of edges, e.g. discontinuities of ∇F, view edges and shadow edges. For further details see [2] and also [5].

Within the last decades further approaches have been proposed. An overview of these methods and their applicabilities in diferent situations is given in [8]. The authors of this paper divide these methods into four categories. The first one contains algorithms which propagate the information about the surface from a set of points over the surface. Horn's approach described above is a special propagation method. The second category is made of those algorithms which minimize a certain functional. Such a functional involves in general a

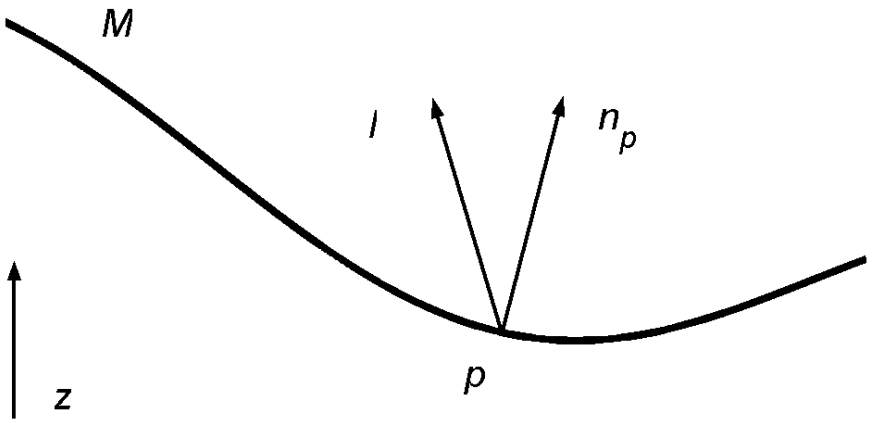

Figure 3.1: The (negative) direction of the incident light l together with the normal vector n_p at a point p on the surface M.

data-fit term and a certain regularization term. The following constraints, which may serve as possible regularization terms, are often used in the minimization process. Such constraints may enforce a smooth surface, or a surface for which $F_{xy} = F_{yx}$, or a surface whose shading image has the same intensity gradients as the given shading image. Thirdly, there are some algorithms which assume a certain local surface type. In this case, the reconstructed surface is approximated with patches which have a prescribed geometrical shape. In [8], an algorithm is presented which locally approximates the surface with spherical patches. Finally, the fourth group of algorithms uses a linearization of the reflectance function

$$R : \begin{cases} D & \to [0,1], \\ (x,y) & \mapsto R(F_x(x,y), F_y(x,y)), \end{cases}$$

which assigns to each pair (x, y) the shading value of a given shape at this point. Generally, R is a nonlinear function with respect to F_x and F_y. However, such a linearization makes sense only if the linear terms in the Taylor series expansion of R dominate. Otherwise, the reconstructed surface might have an essentially different topography as the underlying surface.

3.1 Objective Functional and its Gradient

Here, we choose an approach to the SFS-problem where we formulate a useful functional and minimize it with appropriate optimization techniques. This functional shall be defined in such a way that it attains, for a given shading image, its minimizer at a shape which is as similar as possible to that shape from which the shading image is taken. For sure, we also have to

use some apriori-informations to choose appropriate parameters for the functional as well as for the optimization process; such parameters may control, for example, the smoothness of the resulting shape, the step length of one optimization step or the Riemannian metric which is used in the shape space.

In the sequel, we assume to have a gray level shading image of the shape which we want to reconstruct. In addition, we consider this shape as the graph of a function from a subset $D \subset \mathbb{R}^2$ into $\mathbb{R}$. Doing so, we choose a coordinate system (x, y, z) of $\mathbb{R}^3$ where the z-axis is the direction of the observer. Furthermore, $l \in \mathbb{R}^3$ shall denote the negative direction of the incident light; see Figure 3.1. In the last chapter we introduced the shape space $\mathcal{S}$ of triangulated meshes with fixed connectivity and a given number of nodes N. Within this shape space $\mathcal{S}$ we will now obtain a mesh M which fits to the given shading image. Let also P denote the set of all nodes of M and $(s_p^*)_{p\in P}$ be the collection of the given shading values at the points $p \in P$. Moreover, we assume that the shading image is taken from a Lambertian surface, i.e. that the shading value at the point p is given by the special reflectance function

$$R(p) = \langle n_p, l \rangle.$$

Now let us define a functional f which assigns to a triangular mesh a non-negative real number; this number should be small, if the shading image of the mesh approximately coincides with the given one, and large otherwise. But f should also penalize meshes which are far away from being smooth, hence, we will add a certain regularization term to the data-fit term. Since $\mathcal{S}$ can be identified with $\mathbb{R}^{3N}$, we may define $f : \mathbb{R}^{3N} \to \mathbb{R}_{\geq 0}$ via

$$f(P) := \frac{1}{2} \sum_{p\in P} \Big(\langle n_p(P), l \rangle - s_p^* \Big)^2 + \frac{\alpha}{2} \sum_{(p,q)\in C} \|n_p(P) - n_q(P)\|^2$$

with $\alpha \in \mathbb{R}_{\geq 0}$. First of all, the data-fit term measures differences between the given and the current shading image of the mesh M with nodes P. But the regularization term with its weight α also takes the difference between two neighbouring normal vectors into account; therefore, the functional f "prefers" meshes which are more smooth in the sense of mildly varying normal vectors. In general, such a regularization is necessary since the given shading image only determines the inner product $\langle n_p, l \rangle$ at every point $p \in P$ but not n_p itself; thus, without any regularization of f, the minimization algorithm might find a mesh which perfectly fits to the given shading image but has many spurious edges, when compared to the original shape. However, if α is chosen too large, then the minimization algorithm will not be able to reconstruct an edge which actually appears in the original shape. But this is a well-known problem within the context of inverse problems in general.

In order to minimize the function f using a sensitivity based optimization algorithm, we need to calculate its gradient ∇f. Therefore, we have to find all the partial derivatives $\partial n_p / \partial r^k$ for $p, r \in P$ and $k \in \{1, 2, 3\}$. Let $p \in P$, then

$$n_p = \underbrace{\left(\sum_{t\in\mathcal{T}(p)} (t_2 - p) \times (t_3 - p) \right)}_{=: A} \underbrace{\left(\sum_{s\in\mathcal{T}(p)} \sum_{t\in\mathcal{T}(p)} \Big\langle (s_2 - p) \times (s_3 - p), (t_2 - p) \times (t_3 - p) \Big\rangle \right)^{-\frac{1}{2}}}_{=: B}.$$

Let now $r \in P$ and $k \in \{1,2,3\}$, then either $p = r$ or $p \in \mathcal{N}(r)\backslash\{r\}$ or $p \notin \mathcal{N}(r)$. Above we assumed that all triangles in the mesh M are indexed in the same counterclockwise orientation and that $p = t_1$ for all $t \in \mathcal{T}(p)$; then t_i denotes the i-th vertex of the triangle t. Now we also use the notation $t(t_i = r)_j$ for the j-th vertex of that triangle t whose i-th vertex is r; it will be clear from the context which triangle t is meant, so there will be no ambiguity.

Case 1: $p = r$**:**

$$\begin{aligned}
\frac{\partial n_p}{\partial r^k} &= \left(\sum_{t\in\mathcal{T}(p)} -e_k \times (t_3 - p) - (t_2 - p) \times e_k \right) B \\
&- \frac{AB^3}{2} \Bigg(\sum_{s\in\mathcal{T}(p)} \sum_{t\in\mathcal{T}(p)} \Big\langle -e_k \times (s_3 - p) - (s_2 - p) \times e_k, (t_2 - p) \times (t_3 - p) \Big\rangle \\
&+ \Big\langle (s_2 - p) \times (s_3 - p), -e_k \times (t_3 - p) - (t_2 - p) \times e_k \Big\rangle \Bigg) \\
&= \left(e_k \times \sum_{t\in\mathcal{T}(p)} (t_2 - t_3) \right) B \\
&- AB^3 \left\langle e_k \times \sum_{s\in\mathcal{T}(p)} (s_2 - s_3), \sum_{t\in\mathcal{T}(p)} (t_2 - p) \times (t_3 - p) \right\rangle .
\end{aligned}$$

Case 2: $p \in \mathcal{N}(r)\backslash\{r\}$**:**

$$\begin{aligned}
\frac{\partial n_p}{\partial r^k} &= e_k \times \Big(t(t_2 = r)_3 - t(t_3 = r)_2 \Big) B \\
&- \frac{AB^3}{2} \Bigg(\sum_{t\in\mathcal{T}(p)} \Big\langle e_k \times \Big(s(s_2 = r)_3 - s(s_3 = r)_2 \Big), (t_2 - p) \times (t_3 - p) \Big\rangle \\
&+ \sum_{s\in\mathcal{T}(p)} \Big\langle (s_2 - p) \times (s_3 - p), e_k \times \Big(t(t_2 = r)_3 - t(t_3 = r)_2 \Big) \Big\rangle \Bigg) \\
&= \left(e_k \times \Big(t(t_2 = r)_3 - t(t_3 = r)_2 \Big) \right) B \\
&- AB^3 \left\langle e_k \times \Big(s(s_2 = r)_3 - s(s_3 = r)_2 \Big), \sum_{t\in\mathcal{T}(p)} (t_2 - p) \times (t_3 - p) \right\rangle .
\end{aligned}$$

Case 3: $p \notin \mathcal{N}(r)$**:**

$$\frac{\partial n_p}{\partial r^k} = 0.$$

Now, we come to the calculation of $\partial f/\partial r^k$.

$$\begin{aligned}
\frac{\partial f}{\partial r^k} &= \sum_{p\in P}\Big(\langle n_p, l\rangle - s_p^*\Big)\left\langle \frac{\partial n_p}{\partial r^k}, l\right\rangle + \alpha \sum_{(p,q)\in C}\left\langle n_p - n_q, \frac{\partial n_p}{\partial r^k} - \frac{\partial n_q}{\partial r^k}\right\rangle \\
&= \sum_{p\in P}\Big(\langle n_p, l\rangle - s_p^*\Big)\left\langle \frac{\partial n_p}{\partial r^k}, l\right\rangle + 2\alpha \sum_{(p,q)\in C}\left\langle n_p - n_q, \frac{\partial n_p}{\partial r^k}\right\rangle \\
&= \sum_{p\in P}\Big(\langle n_p, l\rangle - s_p^*\Big)\left\langle \frac{\partial n_p}{\partial r^k}, l\right\rangle + \alpha \sum_{p\in P}\sum_{q\in\mathcal{N}(p)\setminus\{p\}}\left\langle n_p - n_q, \frac{\partial n_p}{\partial r^k}\right\rangle \\
&= \sum_{p\in P}\left\langle \Big(\langle n_p, l\rangle - s_p^*\Big) l + \alpha \sum_{q\in\mathcal{N}(p)\setminus\{p\}} n_p - n_q, \frac{\partial n_p}{\partial r^k}\right\rangle \\
&= \sum_{p\in \mathcal{N}(r)}\left\langle \Big(\langle n_p, l\rangle - s_p^*\Big) l + \alpha \sum_{q\in\mathcal{N}(p)\setminus\{p\}} n_p - n_q, \frac{\partial n_p}{\partial r^k}\right\rangle \qquad (3.1)
\end{aligned}$$

where we used that $\partial n_p/\partial r^k = 0$ for $p \notin \mathcal{N}(r)$. Besides, we employ for an explicit calculation of $\partial f/\partial r^k$ the formula for $\partial n_p/\partial r^k$ with $p \in \mathcal{N}(r)$ obtained above.

Now, we are interested in the optimal descent direction for the function f. Since we only consider deformations of the mesh M along the local normal vectors, we look for a descent direction given by $(\kappa_r n_r)_{r\in P}$. However, we have to be precise and explain in which sense the descent direction shall be *optimal*. The usual gradient is for sure the optimal descent direction in the shape space $\mathcal{S}$ with respect to the Euclidean metric $\langle\cdot,\cdot\rangle^{Eu}$ – but in general not for other Riemannian metrics. To obtain the optimal descent direction for a general Riemannian metric $\langle\cdot,\cdot\rangle^{\mathcal{S}}$, we have to solve the following problem:

$$\begin{cases} \min \quad J\left((\kappa_r)_{r\in P}\right) := \sum\limits_{r\in P}\sum\limits_{k=1}^{3} \frac{\partial f}{\partial r^k}\kappa_r n_r^k, \\ \qquad\; e\left((\kappa_r)_{r\in P}\right) := \langle \vec{\kappa}\cdot\vec{\vec{n}}, \vec{\kappa}\cdot\vec{\vec{n}}\rangle^{\mathcal{S}} - 1 = 0 \end{cases}$$

with $J : \mathbb{R}^N \to \mathbb{R}$ and $e : \mathbb{R}^N \to \mathbb{R}$. The first order necessary optimality condition for this problem reads

$$\nabla J\left((\kappa_r)_{r\in P}\right) + \lambda^* \nabla e\left((\kappa_r)_{r\in P}\right) = 0, \qquad e\left((\kappa_r)_{r\in P}\right) = 0,$$

with $\lambda^* \in \mathbb{R}$, respectively,

$$\left(\sum_{k=1}^{3}\frac{\partial f}{\partial r^k} n_r^k\right)_{r\in P} + 2\lambda^*\left(\langle \vec{\kappa}\cdot\vec{\vec{n}}, \tilde{n}_r\rangle^{\mathcal{S}}\right)_{r\in P} = 0, \qquad \langle \vec{\kappa}\cdot\vec{\vec{n}}, \vec{\kappa}\cdot\vec{\vec{n}}\rangle^{\mathcal{S}} = 1.$$

If we use the Euclidean metric $\langle\cdot,\cdot\rangle^{\mathcal{S}} = \langle\cdot,\cdot\rangle^{Eu}$, then we have $\langle \vec{\kappa}\cdot\vec{\vec{n}}, \tilde{n}_r\rangle^{Eu} = \kappa_r$, and consequently

$$\left(\left\langle \left(\frac{\partial f}{\partial r^k}\right)_{k\in\{1,2,3\}}, n_r\right\rangle\right)_{r\in P} + 2\lambda^*(\kappa_r)_{r\in P} = 0$$

for all $r \in P$. Moreover, the second order necessary optimality condition yields $\lambda^* \geq 0$. If $\lambda^* = 0$, then n_r is orthogonal to $(\partial f/\partial r^k)_{k\in\{1,2,3\}}$ for each $r \in P$. Hence, the directional

derivative of f in the direction of each local normal vector is zero, and consequently, the current mesh is a critical point of the function f. Otherwise, if $\lambda^* > 0$, then

$$\kappa_r = -\frac{1}{2\lambda^*}\left\langle \left(\frac{\partial f}{\partial r^k}\right)_{k\in\{1,2,3\}}, n_r \right\rangle.$$

Since λ^* is a constant, we may use the N-dimensional vector

$$(\kappa_r)_{r\in P} = \left(-\left\langle \left(\frac{\partial f}{\partial r^k}\right)_{k\in\{1,2,3\}}, n_r \right\rangle \right)_{r\in P} \tag{3.2}$$

to characterize the optimal direction $(\kappa_r n_r)_{r\in P}$ to deform the mesh M.

In the case that $\langle\cdot,\cdot\rangle^{\mathcal{S}} = \langle\cdot,\cdot\rangle^{H^n}$ with $n \in \mathbb{N}$, we use the approximation

$$\kappa_q \approx \kappa_p \quad \text{for all} \quad q \in \mathcal{N}(p)\backslash\{p\}.$$

In the same way, we proceeded in sections 2.3 and 2.4 to simplify a band-structured linear system for the variables $\dot{\kappa}_p$ and to obtain an explicit - but approximate - solution. Here, we follow the same lines and find

$$\langle \vec{\kappa}\cdot\vec{\vec{n}}, \tilde{n}_r\rangle^{H^n} = \kappa_r \left(\rho + \left\langle n_p, \sum_{q\in\mathcal{N}(p)\backslash\{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q\rangle \right\rangle \right).$$

Analogously to above, the second order necessary optimality condition implies $\lambda^* \geq 0$ provided $\rho \in \mathbb{R}_{>0}$ is sufficiently large. Moreover, $\lambda^* > 0$ unless the current mesh M is a critical point of the function f. Therefore, the optimal direction for the H^n-metric to deform the mesh M is determined by

$$\begin{aligned}(\kappa_r)_{r\in P} \quad = \quad & \left(-\left(\rho + \left\langle n_p, \sum_{q\in\mathcal{N}(p)\backslash\{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q\rangle \right\rangle \right)^{-1} \right. \\ & \left. \left\langle \left(\frac{\partial f}{\partial r^k}\right)_{k\in\{1,2,3\}}, n_r \right\rangle \right)_{r\in P}\end{aligned} \tag{3.3}$$

which is the same vector as for the Euclidean metric but now every component is weighted with a certain scalar.

Let us also explicitly rewrite the directional derivative of f in the direction of a local normal vector n_r for $r \in P$.

$$\begin{aligned}\left\langle \left(\frac{\partial f}{\partial r^k}\right)_{k\in\{1,2,3\}}, n_r \right\rangle \quad &= \quad \sum_{k=1}^{3} \sum_{p\in\mathcal{N}(r)} \left\langle \Big(\langle n_p, l\rangle - s_p^*\Big) l + \alpha \sum_{q\in\mathcal{N}(p)\backslash\{p\}} n_p - n_q, \frac{\partial n_p}{\partial r^k} \right\rangle n_r^k \\ &= \quad \sum_{p\in\mathcal{N}(r)} \left\langle \Big(\langle n_p, l\rangle - s_p^*\Big) l + \alpha \sum_{q\in\mathcal{N}(p)\backslash\{p\}} n_p - n_q, \sum_{k=1}^{3} \frac{\partial n_p}{\partial r^k} n_r^k \right\rangle.\end{aligned}$$

Generally, what remains to solve the Shape-From-Shading problem for a given gray level shading image of a Lambertian shape M is to make an appropriate choice for each of the subsequent points:

- A Riemannian metric for the shape space $\mathcal{S}$,
- in case an H^n-metric is used, a value for the regularization parameter $\rho \in \mathbb{R}_{>0}$,
- a value for the penalty parameter $\alpha \in \mathbb{R}_{\geq 0}$,
- values for several parameters in the optimization algorithm.

The optimal choices depend, for sure, on the image, the information about the underlying shape and the algorithm which is used; hence, we will collect and discuss appropriate choices in section 3.3.

3.2 Optimization Algorithms

We concentrate our interest on two optimization methods in the shape space S of triangulated meshes with fixed connectivity and N points. First, we consider the basic steepest descent method, which can be implemented quite straight forward. Second, we study the nonlinear conjugate gradient method using the Fletcher-Reeves scheme. The convergence of this method in Riemannian manifolds has been analyzed in [7]. For these algorithms we need all the theoretical concepts introduced in the previous sections, for example, the calculation of geodesics and parallel translates in Riemannian manifolds.

We use MATLAB to implement these optimization methods. In detail, we write two MATLAB-functions which contain the steepest descent algorithm and the NCG-algorithm. Moreover, we also use two functions which evaluate the right-hand-side of the geodesic equations and calculate the parallel translate of a tangent vector along a (discretized) curve in $\mathcal{S}$. These functions will be discussed in the following subsections. However, we also use several functions which we need for technical reasons; these functions are the following ones. The function `crossp` returns the cross product of two three-dimensional vectors; `shade` interpolates the value of the shading image at each point in the given domain. The functions `n` and `n_t` calculate the normal vector to a triangular mesh at a given point, and its derivative respectively; `t` simply returns $\kappa_p n_p$ for given $p \in P$ and $\vec{\kappa}$. In addition, `f` and `gradf` evaluate the function f and expression (3.2), respectively (3.3). `innprod` calculates the Riemannian inner product of two tangent vectors, and `findmin` determines that point on a (discretized) curve in $\mathcal{S}$ where f is minimal. Finally, the functions `plotshape` and `savefig` are used to plot a shape and to generate the resulting image file.

3.2.1 Geodesic Steepest Descent Method

The idea of the geodesic steepest descent method in the shape space $\mathcal{S}$ is quite the same as in the context of vector spaces. One starts the algorithm at some point $x_0 \in \mathcal{S}$ and calculates the direction $v_0 \in T_{x_0}\mathcal{S}$ of steepest descent of the objective function f. But now this direction also depends on the Riemannian metric as we have seen in the previous section. Then, we may consider the geodesic u through x_0 with tangent vector v_0 and employ a line search method to minimize the function f along u. For this case, we calculate a given number of points y_i along

the geodesic u and evaluate f at exactly these points. We determine that point y_{i_0} where f is minimal; this gives us the next iterate x_1 where the procedure is repeated. If we proceed like this, all the points y_i are used and no further computational effort is necessary to calculate other intermediate points on the geodesic u.

The steepest descent method is implemented in the function `steepestdesc`; this function requires several input parameter. `box` is a matrix in $\mathbb{R}^{2\times 2}$ with the lower left and upper right corner of a rectangle in $\mathbb{R}^2$ which determines the x- and y-coordinates of the resulting shape. `tri` is a matrix which contains for each triangle the indices of its vertices, and `boundpts` is a vector whose entries are 1 if the corresponding node is at the boundary of the triangulated mesh and 0 otherwise. This vector can be used to keep the boundary of a mesh fixed during the optimization process. `neibtri`, `neibpts` and `edge` are matrices where each row corresponds to a node and contains a vector of indices of neighbouring nodes; `neibtri` collects the indices of the nodes of all adjacent triangles, `neibpts` the indices of all neighbouring nodes and `edge` the indices of those neighbouring nodes whose index is larger than that of the current node. Moreover, `light` defines the direction of the light source, `shadepts` contains the gray values of the shading image, `alpha` is the penalty parameter in the function f, `m` defines the used H^m-metric and `regul` the regularization parameter of this metric. Finally, `u0` $\in \mathbb{R}^{3N}$ denotes the initial mesh, `itereq` the number of steps along a geodesic as described above, `delta0` the initial step length and `maxit` the maximal number of updates of the triangular mesh.

At the beginning, the function calculates the optimal descent direction for f using the function `gradf`. Within the subsequent while-loops, the following steps are repeated. Firstly, points on the geodesic starting at `u` in direction `kappa` are calculated, secondly, the minimum of f along these points is determined, and thirdly, the new direction of steepest descent of f is calculated. We know from Lemma 2.19 (with $U = V = T$ the tangent vector to the geodesic x) that $\langle T, T\rangle$ is constant along the geodesic x for $T = dx/dt$. Here, we employ this result to `kappa`, which represents the tangent vector T to the geodesic x. At line 26, we calculate the norm `nrm` of the initial tangent vector; and after each step along the geodesic, we scale at line 33 the new tangent vector to have the length `nrm`. This correction makes the procedure of explicit Euler-steps also more stable. Besides, we need the function `equations` which calculates the change of all node positions `u` and of the vector `kappa`; we will describe this function below. In addition, the step length `epsilon` is defined at line 30 in such a way that it is small if the node positions rapidly change; the scalar `delta` then determines the normalized step length. Finally, `delta` can also be reduced at line 51 if `ind` $= 1$; this is the case when the minimum of f along the geodesic is attained for the current shape, then the current step length is too large and has to be reduced.

```
1  %% The steepest descent method
2
3  function [u, res] = steepestdesc(box, tri, boundpts, neibtri, neibpts, edge, ...
4                                   light, shadepts, alpha, m, regul, ...
5                                   u0, itereq, delta0, maxit)
6
7  %% Definition and initialization of several variables
```

```
8  npts = length(u0)/3;
9  h = 1;
10 u = u0;
11 delta = delta0;
12 res = zeros(1, maxit);
13
14 %% Initialization of the search direction
15 kappa = -gradf(box, boundpts, neibtri, neibpts, u', ...
16                light, shadepts, alpha, m, regul);
17
18 while h <= maxit
19     ind = 2;
20
21     while (ind > 1) && (h <= maxit)
22         v = zeros(itereq + 1, 3*npts);
23         v(1, :) = u;
24
25         %% Propagate the solution of the geodesic equation
26         nrm = sqrt(innprod(neibtri, edge, u', kappa, kappa, m, regul));
27         for k = 1:itereq
28             vec = equations(boundpts, neibtri, neibpts, [v(k, :), kappa],
29                             m, regul);
30             epsilon = delta/norm(vec(1:3*npts));
31             v(k + 1, :) = v(k, :) + epsilon*vec(1:3*npts)';
32             kappa = kappa + epsilon*vec(3*npts + 1:end)';
33             kappa = kappa*nrm ...
34                   / sqrt(innprod(neibtri, edge, u', kappa, kappa, m, regul));
35         end
36
37         %% Find the minimum of f along the geodesic path
38         ind = findmin(box, neibtri, edge, ...
39                       light, shadepts, alpha, itereq + 1, v);
40         u = v(ind, :);
41         res(h) = f(box, neibtri, edge, u', light, shadepts, alpha);
42         disp(['ind = ', num2str(ind), ',   f = ', num2str(res(h))]);
43
44         %% Calculate the new search direction
45         kappa = -gradf(box, boundpts, neibtri, neibpts, u', ...
46                        light, shadepts, alpha, m, regul);
47         plotshape(tri, neibtri, u, kappa, 4, [10 20]);
48         h = h + 1;
49     end
50
51     delta = delta/2;
52 end
```

The calculation of the right-hand-side of the geodesic equations (2.7) respectively (2.9) is performed in the function `equations`. The various input parameter have already been described above, except the variable `uk` which is just the concatenation of the vectors `u` and `kappa` to one vector in $\mathbb{R}^{4N}$. At the beginning, the current point on the geodesic and the corresponding tangent vector are determined from the input parameter. Within the for-loop

starting at line 19, all the normal vectors to the current mesh and the change of the current node positions are calculated. At the lines 48 and 51, the user may choose the Riemannian metric which is used for the geodesic equations.

In detail, the calculation of all the normal vectors is separated into the calculation of the weighted averaged normal vector and its length at each vertex p. If an H^n-metric is used, this is more efficient than to call the function `n` which simply returns the normal vector n_p at a vertex p. This will be explained in a moment. Afterwards, the change of the current node position is stored in the vector `vec`. If we decide to use the Euclidean inner product in the shape space $\mathcal{S}$, then $\dot{\kappa}_p = 0$ for all $p \in P$, see line 49. Otherwise, we also have to determine all the vectors $\dot{n}_p$ using the function `n_t`. But this does not require a lot of computational effort, if we use the intermediate results from the calculation of the normal vectors n_p. It remains to calculate $\dot{\kappa}_p$ for all $p \in P$, which is done within the for-loop starting at line 57. This calculation requires several variables for technical reasons. However, the main steps are realized at the lines 78 – 83, where the sums in the numerator and denominator of $\dot{\kappa}_p$ are calculated, compare equation (2.9). Finally, these values are used at line 86 to determine $\dot{\kappa}_p$.

```
1  %% Calculation of the right hand side of the system of geodesic equations
2
3  function vec = equations(boundpts, neibtri, neibpts, uk, m, regul)
4
5  %% Definition and initialization of several variables
6  npts = size(neibtri, 1);
7  vec = zeros(4*npts, 1);
8  nom = cell(npts, 1);
9  den = zeros(npts, 1);
10 nvec = cell(npts, 1);
11 ntvec = cell(npts, 1);
12 tang = cell(npts, 1);
13
14 %% Definition of the point on the geodesic and the corresponding tangent vector
15 u = uk(1:3*npts)';
16 kappa = uk(3*npts + 1:4*npts)';
17
18 %% Calculation of d/dt(u)
19 for delta = 1:npts
20     p = u(3*delta - 2:3*delta);
21     ind = neibtri{delta};
22     nneib = length(ind);
23
24     %% Calculation of the normal vector
25     temp = [0; 0; 0];
26     for k = 1:2:nneib
27         i = ind(k);
28         j = ind(k + 1);
29         temp = temp + crossp(u(3*i - 2:3*i) - p, u(3*j - 2:3*j) - p);
30     end
31     nom{delta} = temp;
32
```

```
33      if norm(temp) == 0;
34          den(delta) = 1;
35          nvec{delta} = temp;
36      else
37          den(delta) = 1/norm(temp);
38          nvec{delta} = temp/norm(temp);
39      end
40
41      %% Calculation of d/dt(u^delta)
42      tang{delta} = kappa(delta)*nvec{delta};
43      if boundpts(delta) == 0
44          vec(3*delta - 2:3*delta) = tang{delta};
45      end
46  end
47
48  % %% Calculation of d/dt(kappa) using the Euclidean inner product
49  % vec(3*npts + 1:4*npts) = 0;
50
51  %% Calculation of d/dt(kappa) using a Riemannian metric of H^m-type
52  %% Calculation of d/dt(n_p)
53  for delta = 1:npts
54      ntvec{delta} = n_t(neibtri, u, tang, nom{delta}, den(delta), delta);
55  end
56
57  for delta = 1:npts
58      if boundpts(delta) == 0
59          p = u(3*delta - 2:3*delta);
60          kappap = kappa(delta);
61          nvecp = nvec{delta};
62          ntp = ntvec{delta};
63          tp = tang{delta};
64          ind = neibpts{delta};
65          nneib = length(ind);
66
67          %% Calculation of d/dt(kappa^delta)
68          temp1 = [0; 0; 0];
69          temp2 = [0; 0; 0];
70          for k = 1:nneib;
71              l = ind(k);
72              q = u(3*l - 2:3*l);
73              kappaq = kappa(l);
74              nvecq = nvec{l};
75              ntq = ntvec{l};
76              tq = tang{l};
77
78              temp1 = temp1 + (p - q)/norm(p - q)^(2*m) ...
79                            * (m*((tp - tq)'*(p - q))^2/norm(p - q)^2 ...
80                            - (kappap*ntp - kappaq*ntq)'*(p - q) ...
81                            - norm(tp - tq)^2);
82
83              temp2 = temp2 + (p - q)/norm(p - q)^(2*m)*(nvecp - nvecq)'*(p - q);
84          end
```

```
85
86          vec(3*npts + delta) = (nvecp'*temp1)/(regul + nvecp'*temp2);
87      end
88  end
```

3.2.2 Geodesic Nonlinear Conjugate Gradient Method

In contrast to the steepest descent method, the nonlinear conjugate gradient (NCG) method also uses gradients and search directions from previous iterates to calculate the new search direction. Therefore, it is necessary to "compare" tangent vectors from different tangent spaces, say $X \in T_{M_1}\mathcal{S}$ and $Y \in T_{M_2}\mathcal{S}$ for different meshes $M_1, M_2 \in \mathcal{S}$. In $\mathbb{R}^n$, endowed with the Euclidean inner product, this is trivial since each tangent space $T_x\mathbb{R}^n$ is identified with $\mathbb{R}^n$ itself. But in general, we cannot identify $T_{M_2}\mathcal{S}$ with $T_{M_1}\mathcal{S}$. However, we may parallel translate the tangent vector X along a geodesic joining M_1 and M_2 to a tangent vector $X' \in T_{M_2}\mathcal{S}$; then one can compare the vectors X' and Y, since they are elements of the same tangent space $T_{M_2}\mathcal{S}$.

Generally, the NCG-algorithm follows the same ideas as the steepest descent algorithm presented in subsection 3.2.1, except that the search direction is calculated in a different manner. For this case, consider the old iterate $x_0 \in \mathcal{S}$, the gradient $\nabla f(x_0) \in T_{x_0}\mathcal{S}$, the search direction $v_0 \in T_{x_0}\mathcal{S}$, the new iterate $x_1 \in \mathcal{S}$, the gradient $\nabla f(x_1) \in T_{x_1}\mathcal{S}$ and the (discrete) geodesic path u joining x_0 and x_1. Then we parallel translate the vector v_0 along u to a vector $v_0' \in T_{x_1}\mathcal{S}$, which is in the same tangent space as $\nabla f(x_1)$. The new search direction $v_1 \in T_{x_1}\mathcal{S}$ is then calculated using the Fletcher-Reeves scheme,

$$v_1 := \nabla f(x_1) + \gamma v_0' \quad \text{with} \quad \gamma := \frac{\langle \nabla f(x_1), \nabla f(x_1) \rangle^{\mathcal{S}}}{\langle \nabla f(x_0), \nabla f(x_0) \rangle^{\mathcal{S}}}$$

where $\langle \cdot, \cdot \rangle^{\mathcal{S}}$ denotes the Riemannian metric in $\mathcal{S}$.

This algorithm is implemented in the function `ncg`, which has the same input arguments as the function `steepestdesc`, except from the variable `restart` which determines the number of iterations performed before the search direction is reset to the steepest descent direction. Except for some technical details, the function `ncg` coincides with the function `steepestdesc` up to line 47. Then the old negative gradient of f is stored and the new negative gradient of f is calculated. If the minimum of f along the geodesic path is attained for the current mesh, then a restart is performed, see line 56. The variable `newgrad` is true if and only if the search direction has just been reset but not in the previous iteration step. Then the condition in line 24 is true if `ind` $= 1$ and the search direction has been reset but not directly before. If the minimum of f is attained for the current mesh even for points along a geodesic in the direction of steepest descent, then the algorithms remains at the current iterate; consequently, `ind` $= 1$, `newgrad` is false (see line 59) and the condition at line 24 is false (always provided `h` $\leq$ `maxit`). If no restart is performed, then the new search direction is calculated as described above and the variable `newgrad` is set false. However, we employed the function `parallel`, which we describe below, to calculate the parallel translate of the old search direction to the

current iterate. Finally, the step length `delta` is reduced, if the algorithm stopped with the current step length `delta` and `h` $\leq$ `maxit`.

```
% The nonlinear conjugate gradient method using the Fletcher–Reeves–scheme

function [u, res] = ncg(box, tri, boundpts, neibtri, neibpts, edge, ...
                        light, shadepts, alpha, m, regul, ...
                        u0, itereq, delta0, maxit, restart)

%% Definition and initialization of several variables
npts = length(u0)/3;
h = 1;
u = u0;
delta = delta0;
res = zeros(1, maxit);

%% Initialization of the search direction
kappa1 = -gradf(box, boundpts, neibtri, neibpts, u', ...
                light, shadepts, alpha, m, regul);
lambda1 = kappa1;
newgrad = true;

while h <= maxit
    ind = 2;

    %% Calculate and plot the deformed shape
    while ((ind > 1) || (newgrad == true)) && (h <= maxit)
        uu = zeros(itereq + 1, 3*npts);
        kk = zeros(itereq + 1, npts);
        uu(1, :) = u;
        kk(1, :) = lambda1;

        %% Propagate the solution of the geodesic equation
        nrm = sqrt(innprod(neibtri, edge, u', lambda1, lambda1, m, regul));
        for k = 1:itereq
            vec = equations(boundpts, neibtri, neibpts, ...
                            [uu(k, :), kk(k, :)], m, regul);
            epsilon = delta/norm(vec(1:3*npts));
            uu(k + 1, :) = uu(k, :) + epsilon*vec(1:3*npts)';
            kk(k + 1, :) = kk(k, :) + epsilon*vec(3*npts + 1:end)';
            kk(k + 1, :) = kk(k + 1, :)*nrm/sqrt(innprod(neibtri, edge, ...
                           u', kk(k + 1, :), kk(k + 1, :), m, regul));
        end

        %% Find the minimum of f along the geodesic path
        ind = findmin(box, neibtri, edge, ...
                      light, shadepts, alpha, itereq + 1, uu);
        u = uu(ind, :);
        res(h) = f(box, neibtri, edge, u', light, shadepts, alpha);
        disp(['ind = ', num2str(ind), ',   f = ', num2str(res(h))]);

```

```
49          %% Save the old gradient and calculate the new gradient
50          kappa0 = kappa1;
51          kappa1 = -gradf(box, boundpts, neibtri, neibpts, u', ...
52                          light, shadepts, alpha, m, regul);
53
54          plotshape(tri, neibtri, u, kappa1, 4, [10 20]);
55
56          if (ind == 1) || (mod(h, restart) == 0)
57              %% A restart is performed
58              lambda1 = kappa1;
59              newgrad = ~newgrad;
60              disp('restart');
61          else
62              %% The old search direction is parallel translated
63              gamma = innprod(neibtri, edge, u', kappa1, kappa1, m, regul) ...
64                    / innprod(neibtri, edge, u', kappa0, kappa0, m, regul);
65              lambda0 = parallel(boundpts, neibtri, neibpts, edge, ...
66                                 [0:epsilon:(ind - 1)*epsilon], ...
67                                 uu(1:ind - 1,:), kk(1:ind - 1,:), lambda1, ...
68                                 m, regul);
69
70              %% The new search direction is calculated
71              lambda1 = kappa1 + gamma*lambda0;
72              newgrad = false;
73          end
74
75          plotshape(tri, neibtri, u, lambda1, 5, [10 20]);
76          h = h + 1;
77      end
78
79      delta = delta/2
80  end
```

The function `parallel` calculates the parallel translate of a given tangent vector along a geodesic using either equation (2.10) or (2.11). The input parameter `boundpts`, `neibtri`, `neibpts` and `edge` have already been described in subsection 3.2.1. The vector `epsilon` contains the step lengths along the geodesic path `uu` and `kk` represents the corresponding tangent vectors. `lambda` is the vector which is parallel displaced; `m` and `regul` are the same as above. Analogously to the function `equations`, we may choose the Riemannian metric which is used in the shape space $\mathcal{S}$. If we use the Euclidean metric, then the parallel translate is just `lambda` due to equation (2.10).

If we decide to use an H^m-metric, then the function follows quite the same strategy to calculate the parallel translate as in the calculation of the geodesic path. This is the case since the geodesic equation and the equation of parallel translation are very similar. At the beginning, the norm of `lambda` is calculated. Due to Lemma 2.19 (with U = V = Y the tangent vector represented by `lambda`), this norm is a constant under parallel displacement. Then the parallel translate of the vector is calculated at each point of the geodesic path; this is realized with the for-loop starting at line 18. Within this loop, the following steps are repeated. First, all the normal vectors to the current mesh are calculated and several intermediate results are

stored; this is done analogously to the function `equations`. In the same manner as above, all the vectors $\dot{n}_p$ are calculated at line 55. Now, the change of the tangent vector `lambda` is calculated, but since the boundary is fixed during the minimization process, we need not consider its parallel translate at the boundary. Now, we have to deal with several variables which we need for technical reasons. But similar to the function `equations` we calculate the numerator and the denominator of the right-hand-side of equation (2.11) within the lines 81 – 88. Then the parallel translate of `lambda` at the next iterate of the geodesic path is calculated at line 92. If the update is done for all points, then the vector is scaled at line 96 in such a way that is has the same norm as the initial tangent vector.

```
1  % Calculation of the parallel translate of a tangent vector along a geodesic
2
3  function vec = parallel(boundpts, neibtri, neibpts, edge, ...
4                          epsilon, uu, kk, lambda, m, regul)
5
6  % %% Parallel translation of lambda using the Euclidean inner product
7  % vec = lambda;
8
9  %% Parallel translation of lambda using a Riemannian metric of H^m-type
10 %% Definition and initialization of several variables
11 npts = size(neibtri, 1);
12 nsteps = length(epsilon);
13
14 %% Calculation of the norm of the initial tangent vector
15 nrm = sqrt(innprod(neibtri, edge, uu(1, :)', lambda, lambda, m));
16
17 %% Calculate the parallel translate at all points of the geodesic path
18 for h = 1:nsteps
19     u = uu(h, :)';
20     kappa = kk(h, :)';
21     lambda0 = lambda;
22     nom = cell(npts, 1);
23     den = zeros(npts, 1);
24     nvec = cell(npts, 1);
25     ntvec = cell(npts, 1);
26     tang = cell(npts, 1);
27
28     %% Calculation of d/dt(u)
29     for i = 1:npts
30         p = u(3*i - 2:3*i);
31         ind = neibtri{i};
32         nneib = length(ind);
33
34         %% Calculation of the normal vector
35         temp = [0; 0; 0];
36         for j = 1:2:nneib
37             k = ind(j);
38             l = ind(j + 1);
39             temp = temp + crossp(u(3*k - 2:3*k) - p, u(3*l - 2:3*l) - p);
```

```
        end
        nom{i} = temp;

        if norm(temp) == 0;
            den(i) = 1;
            nvec{i} = temp;
        else
            den(i) = 1/norm(temp);
            nvec{i} = temp/norm(temp);
        end
        tang{i} = kappa(i)*nvec{i};
    end

    %% Calculation of d/dt(n_p)
    for i = 1:npts
        ntvec{i} = n_t(neibtri, u, tang, nom{i}, den(i), i);
    end

    for i = 1:npts
        if boundpts(i) == 0
            p = u(3*i - 2:3*i);
            nvecp = nvec{i};
            ntp = ntvec{i};
            tp = tang{i};
            xp = lambda0(i)*nvecp;
            kappap = kappa(i);
            ind = neibpts{i};
            nneib = length(ind);

            %% Calculation of d/dt(lambda^delta)
            temp1 = [0; 0; 0];
            temp2 = [0; 0; 0];
            for k = 1:nneib;
                l = ind(k);
                q = u(3*l - 2:3*l);
                kappaq = kappa(l);
                nvecq = nvec{l};
                ntq = ntvec{l};
                tq = tang{l};
                xq = lambda0(l)*nvecq;

                temp1 = temp1 + (p - q)/norm(p - q)^(2*m) ...
                              * (m*((xp - xq)'*(p - q)) ...
                              * ((tp - tq)'*(p - q))/norm(p - q)^2 ...
                              - (kappap*ntp - kappaq*ntq)'*(p - q) ...
                              - (xp - xq)'*(tp - tq));

                temp2 = temp2 + (p - q)/norm(p - q)^(2*m) ...
                              * ((nvecp - nvecq)'*(p - q));
            end

            temp3 = (nvecp'*temp1)/(regul + nvecp'*temp2);
```

```
92              lambda(i) = lambda0(i) + epsilon(h)*temp3;
93          end
94      end
95
96      lambda = lambda*nrm ...
97             / sqrt(innprod(neibtri, edge, u, lambda, lambda, m, regul));
98  end
99  vec = lambda;
```

3.3 Results and Comparison of the Different Approaches

In this section, we present the results of the optimization algorithms, which we described in section 3.2, in different situations. For this case, we consider three gray level shading images. The first image contains the calculated shading values of a smooth synthetic surface which is the graph of a function. Besides, we use the same surface to generate various shading images; one image for a light direction which coincides with the direction of the observer and some images for the case of an oblique light source. The second shape which we want to reconstruct is the bottom of a small ceramic box. This shape has quite a challenging topography with smooth parts and a sharp circular elevation. Finally, we test our optimization methods with a shading image of the author's face.

In section 3.2 we discussed two algorithms, the geodesic steepest descent method and the geodesic NCG-method. However, we will see that the difference between these two methods depends on the problem but is generally quite small. Moreover, the visual results are nearly the same, hence, we will only present the images of the results obtained with the NCG-method. Furthermore, we have to find appropriate values for various parameter in the optimization algorithms; these values shall only depend on the shape which we want to reconstruct. We do this in order to compare the performance of the algorithms using different Riemannian metrics in the shape space $\mathcal{S}$; these metrics shall be the Euclidean metric, the H^0- and the H^2-metric. Below, we collect these values for each shape and discuss the advantages of these metrics in the different situations.

First, we consider a synthetic surface, which is the graph of the following function $g : [-1,1] \times [-1,1] \to \mathbb{R}$,

$$g(x,y) = e^{\frac{1}{x^2-1}} e^{\frac{1}{y^2-1}} \left(\cos\left(10\sqrt{(x+.2)^2+y^2}\right) + \cos\left(10\sqrt{(x-.2)^2+y^2}\right)\right).$$

This surface is shown in three different perspectives in Figure 3.2. If the direction of the light coincides with the z-axis, we cannot initialize the algorithms with a triangulated planar surface since this is already a critical point of the function f (see equation (3.1)). Hence, we use a triangulated mesh which slightly differs from a plane. Then, we reconstruct the shape in two steps. First, we use a coarse mesh (with edge length 0.1) to construct a shape which has the correct surface topography, see Figure 3.3. If the resolution is too coarse, then we may not reconstruct all details of the topography; otherwise, if the resolution is too fine, the reconstruction is generally not smooth enough. An increase of the regularization term of f is not the best choice since this also causes the final reconstruction to be overly smooth in

comparison to the given data. However, this first optimization process results in a certain shape. Now, we refine the corresponding mesh and use this refined mesh (with edge length 0.05) as the initialization of the second optimization process on a finer level. This finer level is also the resolution which will be used for the reconstruction of the remaining shapes.

Figure 3.4 shows the final reconstruction of the surface for the three considered Riemannian metrics. The convergence of the two optimization processes is presented in the Figures 3.5 and 3.6. One sees that the difference between the steepest descent and the NCG-method is quite small; however, the reconstruction is a bit more smooth, if the H^0- or the H^2-metric is used. Besides, Table 3.1 contains the values of the function f for the initial and final shapes of the two optimization processes, if the steepest descent method or the NCG-method is used. These results were obtained with the following choices for the parameters which we already described in section 3.2.

- light direction $l = (0, 0, 1)$, regularization $\alpha = 0.05$,
- $\rho = 0.001$ for the H^0-metric and $\rho = 30$ for the H^2-metric,
- `itereq` $= 3$, `maxit` $= 50$, `delta` $= 0.01$ and `restart` $= 5$ for the first optimization process,
- `itereq` $= 3$, `maxit` $= 20$, `delta` $= 0.05$ and `restart` $= 5$ for the second optimization process.

For the H^2-metric a higher regularization is necessary as for the H^0-metric since $\|p-q\|^{2n}$ is much smaller for $n = 2$, and hence, the absolute value of

$$\left\langle n_p, \sum_{q \in \mathcal{N}(p) \setminus \{p\}} \frac{p-q}{\|p-q\|^{2n}} \langle n_p - n_q, p-q \rangle \right\rangle$$

may be much larger. This will be the same for the remaining shapes.

The second shape is the bottom of a small ceramic box. We use a gray level image, which is a part of a usual jpg-image, to reconstruct the surface. However, the surface of the ceramic is also specular and thus some reflexions appear in the image. These reflexions are removed in the course of preprocessing the shading image. Figure 3.7 also shows the initial shape for the optimization algorithms, which is a flat parabolic surface. Here, we only use one resolution to reconstruct the surface. In Figure 3.8 the shading images of the initial shape and of the reconstructed shape for the H^2-metric are plotted. Figure 3.9 shows the final result for the three different Riemannian metrics. Similar to the synthetic surface, the results for the different metrics only slightly differ. In addition, Figure 3.10 shows the convergence of the algorithms and Table 3.2 the final values of f for the steepest descent and NCG-method. Here, we see that the NCG-algorithm reaches a smaller value of f than the steepest descent method after the same number of iterations. The choices for the parameters in the algorithms are

- light direction $l = (0, 0, 1)$, regularization $\alpha = 0.1$,
- $\rho = 0.01$ for the H^0-metric and $\rho = 100$ for the H^2-metric,

- `itereq` $= 3$, `maxit` $= 100$, `delta` $= 0.02$ and `restart` $= 5$ for the optimization process.

However, we have to keep in mind that the higher the regularization is, the more Euclidean is the performance of the metric. But due to numerical tests, we have to choose quite a high regularization for the H^2-metric; consequently, the behaviour is similar to the Euclidean metric.

A challenging problem is the reconstruction of a face from a shading image. Firstly, this is the case since a face consists of rather smooth parts together with regions where the local gradient is quite large. Secondly, some parts of the face may even not be visible, for example the side of the nose; strictly speaking, we therefore cannot consider the face as the graph of a function as explained in section 3.1. And thirdly, a face is generally not an ideal Lambertian surface. Especially, the black eyebrow and eyelash have to be removed in the image; otherwise, the algorithms "interpret" the eyelash as a region with a large gradient, in contrast to the reality. Figure 3.11 shows the initial shape for the reconstruction and the shading image of the author's face, where the eyebrow and eyelash are at least toned down. However, neither the steepest descent method nor the NCG-method converges to a final shape which is sufficiently close to any face. Only within the first 30 iterations, the reconstruction is similar to a face. Thus, we compare the intermediate results for the Euclidean metric, the H^0- and the H^2-metric after 10, 20 and 30 iterations (see Figures 3.13 – 3.15). Besides, Figure 3.12 contains the shading images of the initial shape and of the reconstruction of the face after 30 iterations using the H^2-metric. Figure 3.16 shows the convergence of the NCG-algorithm for these three metrics. However, we see that the values of f only slightly decrease during the first 30 iterations; hence, one has to adopt the functional f, and maybe also the used Riemannian metric, to find a realistic reconstruction of the face. Table 3.3 contains certain values of f, which are within the same range for both, the steepest descent method and the NCG-method. For the algorithms we use

- light direction $l = (0, 0, 1)$, regularization $\alpha = 0.1$,
- $\rho = 0.001$ for the H^0-metric and $\rho = 10$ for the H^2-metric,
- `itereq` $= 3$, `maxit` $= 30$, `delta` $= 0.02$ and `restart` $= 5$ for the optimization process.

Finally, we apply the steepest descent method and the NCG-method to reconstruct the synthetic shape, but now from the shading image of an oblique light source at $l = (0.1, 0, 1)$, $l = (0, 0.1, 1)$ and $l = (0.1, 0.1, 1)$. Strictly speaking we use the normalized vector $\bar{l} = l/\|l\|$. For this case, we use exactly the same approach as for the reconstruction of the synthetic surface described above, except from the number of iterations which we increase to 100 iterations for both, the coarse-grid and the fine-grid-optimization process. In addition, we use the negative of the initial shape used above for the cases $l = (0, 0.1, 1)$ and $l = (0.1, 0.1, 1)$; and we also invert the final shape for the case $l = (0, 0.1, 1)$. The results are shown in Figure 3.17 for the steepest descent method and in Figure 3.18 for the NCG-method. The best result is obtained with a light source at $l = (0, 0.1, 1)$. For the case $l = (0.1, 0, 1)$, the algorithms are able to reconstruct the global topography, whereas for $l = (0.1, 0.1, 1)$ neither the steepest descent nor the NCG-algorithm sufficiently reconstructs the given surface. Moreover, the convergence of

the second optimization process is plotted in Figure 3.19 for the steepest descent and in Figure 3.20 for the NCG-method. One sees that the values of f for the last iterates are quite close to their limit; only the steepest descent method for $l = (0.1, 0, 1)$ seems to minimize f (relevantly) also within the next few iterations. Furthermore, Table 3.4 compares the values of f for the initial and final shapes of the two optimization processes.

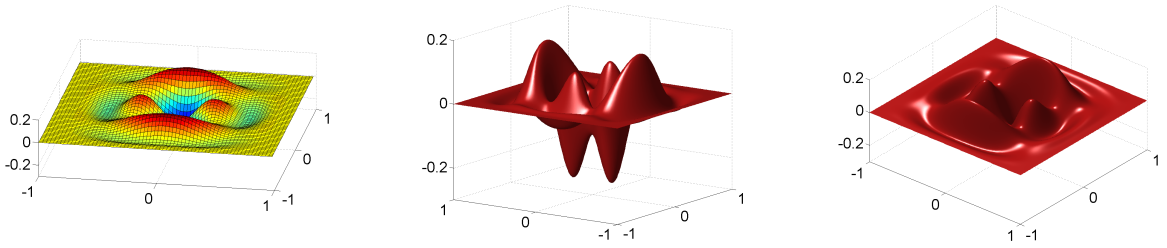

Figure 3.2: The synthetic shape shown in three different perspectives. Note the different scales of the z-axis.

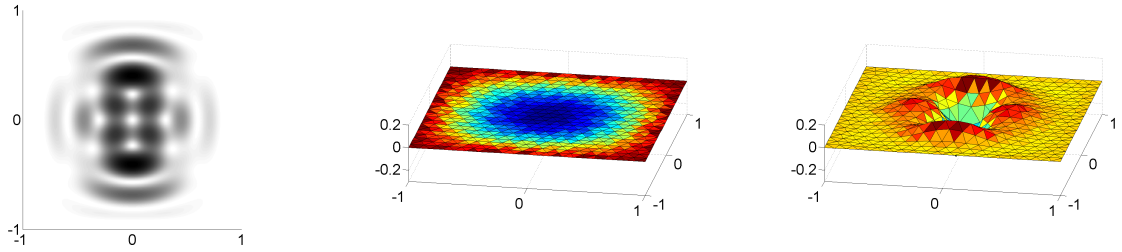

Figure 3.3: The shading image of the synthetic shape, the initial shape and the reconstructed surface after the coarse-grid-optimization process.

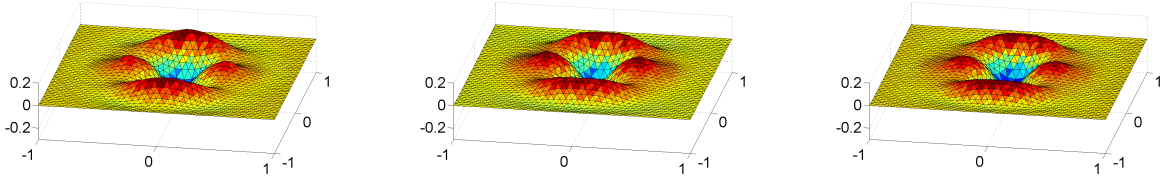

Figure 3.4: Reconstruction of the synthetic shape using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

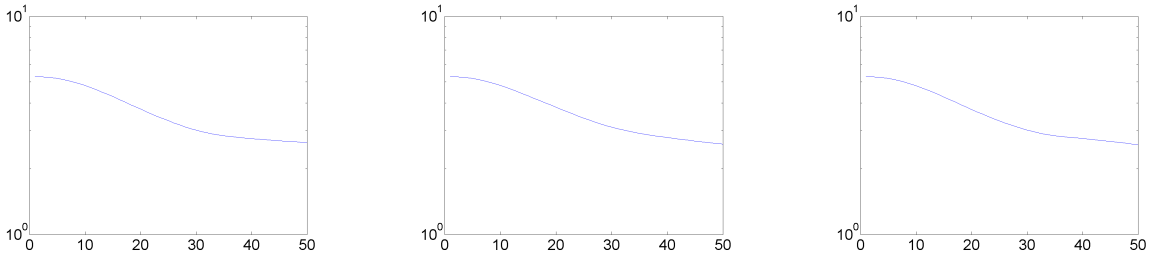

Figure 3.5: Convergence of the coarse-grid-optimization process for the synthetic shape using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

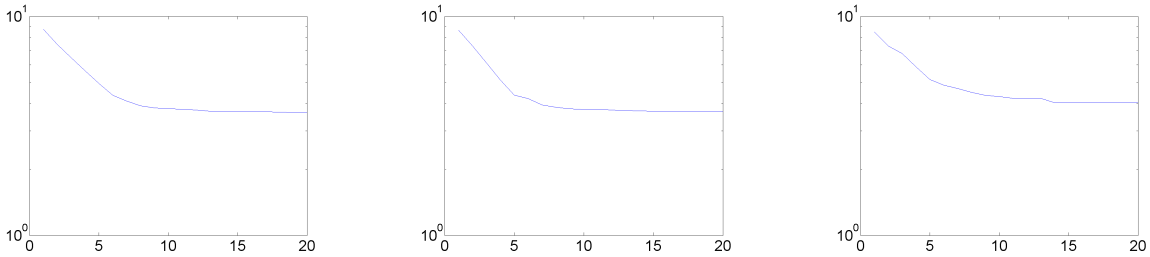

Figure 3.6: Convergence of the fine-grid-optimization process for the synthetic shape using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

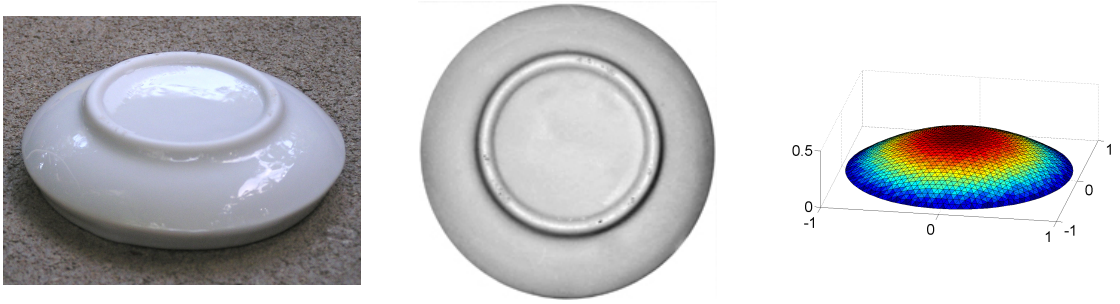

Figure 3.7: The bottom of the ceramic box, its shading image and the initial shape.

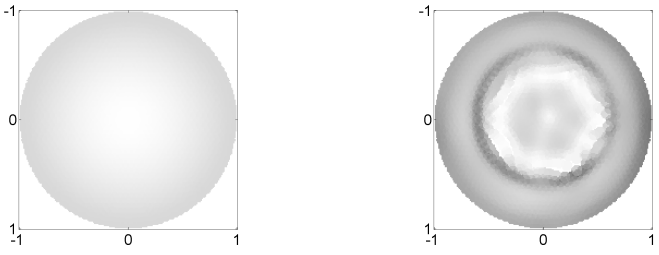

Figure 3.8: The shading images of the initial shape and of the reconstructed shape.

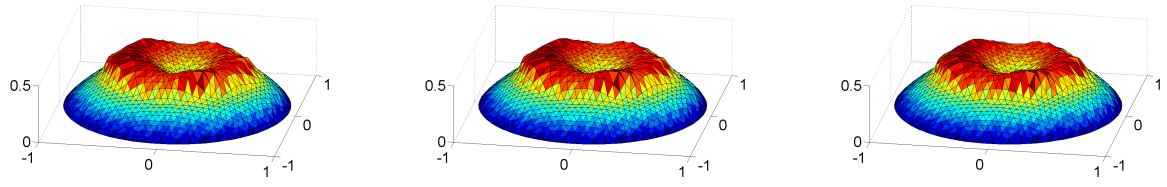

Figure 3.9: Reconstruction of the bottom of the ceramic box using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

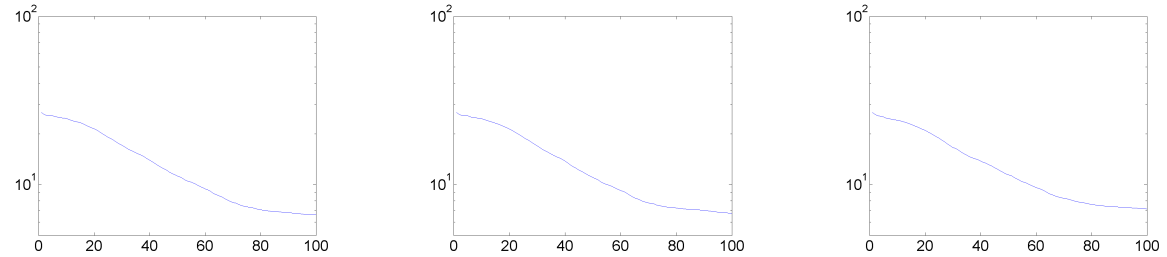

Figure 3.10: Convergence of the optimization process for the bottom of the ceramic box using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

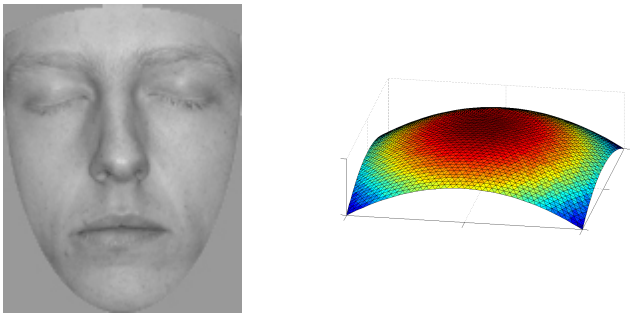

Figure 3.11: The shading image of the author's face and the initial shape.

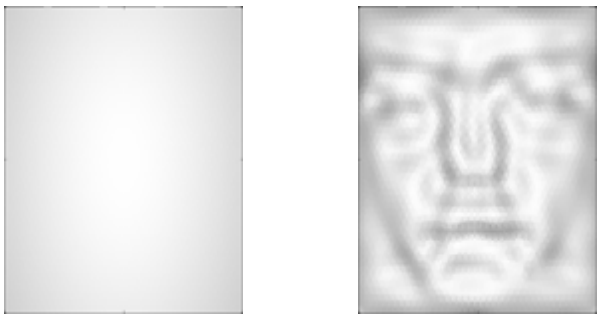

Figure 3.12: The shading images of the initial shape and of the reconstruction of the face.

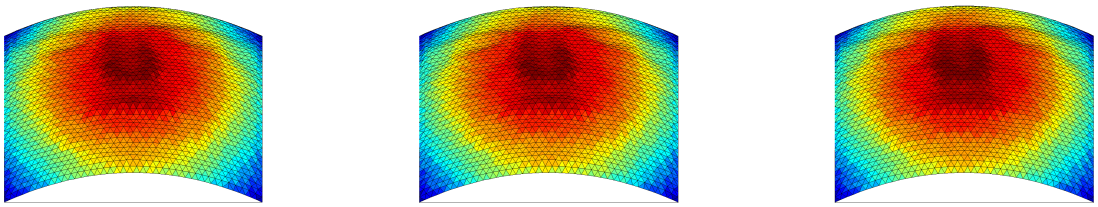

Figure 3.13: Reconstruction of the face after 10 iterations using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

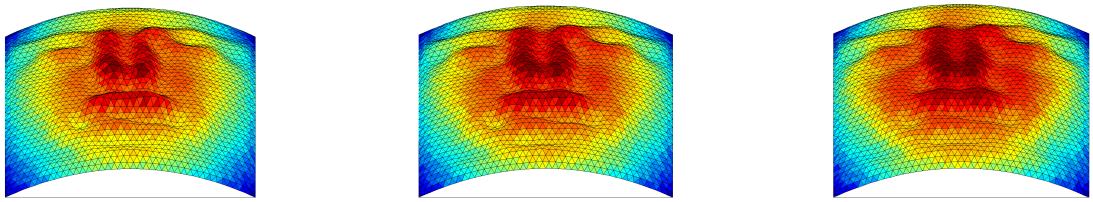

Figure 3.14: Reconstruction of the face after 20 iterations using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

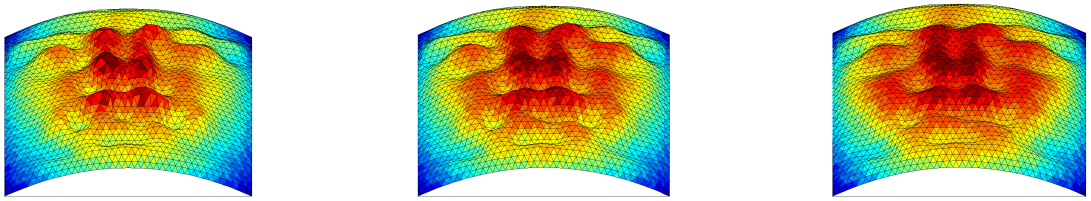

Figure 3.15: Reconstruction of the face after 30 iterations using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

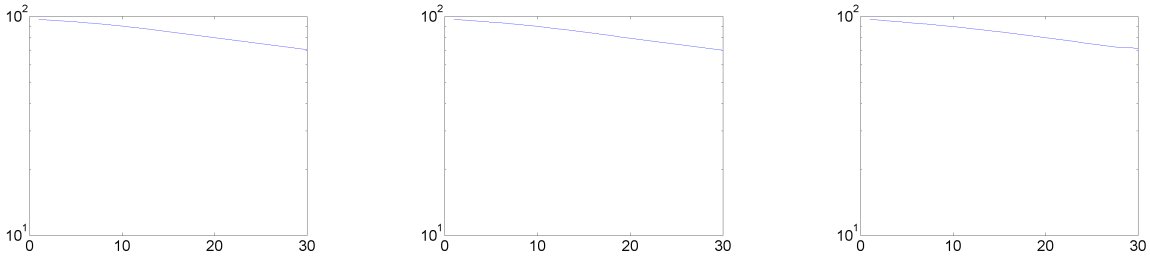

Figure 3.16: Convergence of the optimization process for the face using different Riemannian metrics. From left to right: Euclidean, H^0- and H^2-metric.

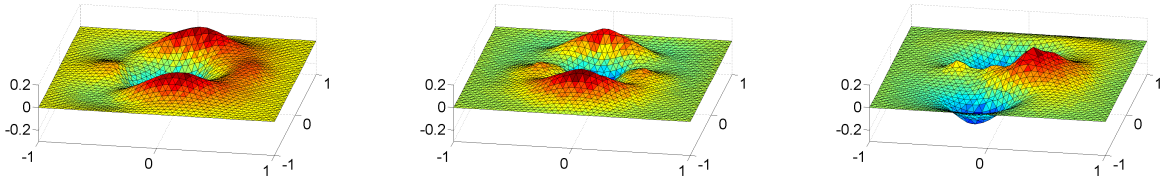

Figure 3.17: Reconstruction of the synthetic shape using the steepest descent method, the Euclidean metric and an oblique light source. From left to right: $l = (0.1, 0, 1)$, $l = (0, 0.1, 1)$, $l = (0.1, 0.1, 1)$.

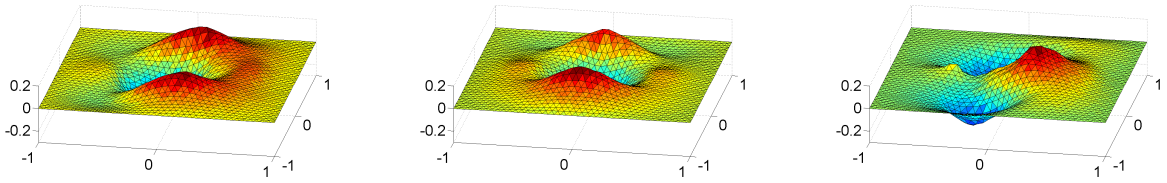

Figure 3.18: Reconstruction of the synthetic shape using the NCG-method, the Euclidean metric and an oblique light source. From left to right: $l = (0.1, 0, 1)$, $l = (0, 0.1, 1)$, $l = (0.1, 0.1, 1)$.

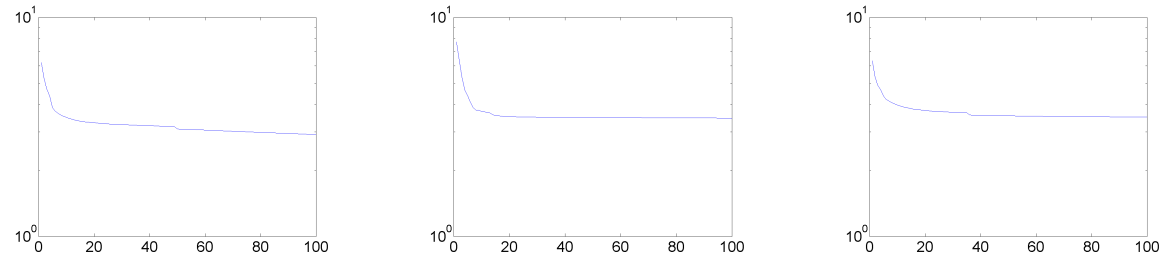

Figure 3.19: Convergence of the fine-grid-optimization process for the synthetic shape using the steepest descent method, the Euclidean metric and an oblique light source. From left to right: $l = (0.1, 0, 1)$, $l = (0, 0.1, 1)$, $l = (0.1, 0.1, 1)$.

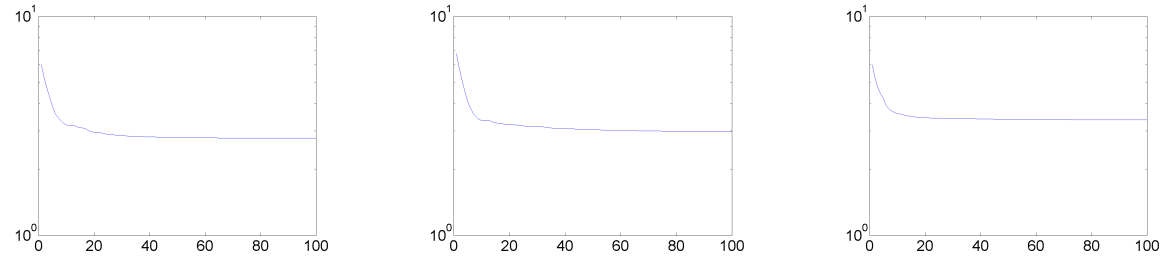

Figure 3.20: Convergence of the fine-grid-optimization process for the synthetic shape using the NCG-method, the Euclidean metric and an oblique light source. From left to right: $l = (0.1, 0, 1)$, $l = (0, 0.1, 1)$, $l = (0.1, 0.1, 1)$.

Table 3.1: The values of the function f for the initial and final shapes in the reconstruction of the synthetic surface. Left: Steepest descent method. Right: NCG-method

	Eu	H^0	H^2
$f^{(1)}_{init}$	5.29	5.29	5.29
$f^{(1)}_{final}$	2.54	2.69	2.52
$f^{(2)}_{init}$	10.24	10.97	9.99
$f^{(2)}_{final}$	3.63	3.92	3.84

	Eu	H^0	H^2
$f^{(1)}_{init}$	5.29	5.29	5.29
$f^{(1)}_{final}$	2.64	2.60	2.58
$f^{(2)}_{init}$	10.63	10.51	10.09
$f^{(2)}_{final}$	3.64	3.67	4.03

Table 3.2: The values of the function f for the initial and final shape in the reconstruction of the bottom of the ceramix box. Left: Steepest descent method. Right: NCG-method

	Eu	H^0	H^2
f_{init}	27.94	27.94	27.94
f_{final}	9.31	9.36	9.05

	Eu	H^0	H^2
f_{init}	27.94	27.94	27.94
f_{final}	6.63	6.77	7.18

Table 3.3: The values of the function f for the initial shape and the reconstruction of the face after 10, 20 and 30 iterations. Left: Steepest descent method. Right: NCG-method

	Eu	H^0	H^2
f_{init}	98.19	98.19	98.19
f_{10}	89.35	89.30	89.48
f_{20}	78.58	78.62	79.41
f_{30}	69.20	68.97	71.59

	Eu	H^0	H^2
f_{init}	98.19	98.19	98.19
f_{10}	90.27	89.91	89.72
f_{20}	79.84	79.48	79.87
f_{30}	70.53	70.04	71.66

Table 3.4: The values of the function f for the initial and final shapes in the reconstruction of the synthetic surface using the Euclidean metric and an oblique light source l. Left: Steepest descent method. Right: NCG-method

l	$(0.1, 0, 1)$	$(0, 0.1, 1)$	$(0.1, 0.1, 1)$
$f^{(1)}_{init}$	5.36	5.43	5.50
$f^{(1)}_{final}$	2.11	2.36	1.92
$f^{(2)}_{init}$	7.93	9.57	8.22
$f^{(2)}_{final}$	2.92	3.47	3.50

l	$(0.1, 0, 1)$	$(0, 0.1, 1)$	$(0.1, 0.1, 1)$
$f^{(1)}_{init}$	5.36	5.43	5.50
$f^{(1)}_{final}$	1.98	2.14	1.84
$f^{(2)}_{init}$	7.72	8.60	7.90
$f^{(2)}_{final}$	2.78	2.98	3.36

Chapter 4

Remarks and Outlook

In the process of writing the thesis and in discussions with Prof. Ring, some suggestions for further improvements or further studies came up. Some of these ideas would require to rewrite a large part of the algorithms and functions and could be part of future research.

One suggestion concerns the Riemannian metric which is used in the shape space $\mathcal{S}$. In the thesis, we considered the Euclidean metric and the H^n-metric. Since, the Euclidean metric is the standard inner product in $\mathcal{S} \cong \mathbb{R}^{3N}$, this inner product serves as a reference metric for more sophisticated metrics. For example, we may compare the advantages of each metric for a certain problem. However, the idea to introduce the H^n-metric was to adopt the as-isometric-as-possible-metric from [4] in such a way to penalize points that come too close to each other more effectively. And in principle, we could realize this idea. Nevertheless, Kilian et al. [4] wanted to *deform* meshes as-isometric-as-possible, hence, they looked for a metric which penalizes non-isometric deformations. But we are not interested in special deformations, we want to *minimize* a function in the shape space. Thus, it may be more advantageous to construct a metric such that the optimal descent directions for a function allow a faster convergence to the minimizer, in comparison to the Euclidean metric.

Another idea was introduced in the context of the geodesic equations. In order to obtain an explicit formula for $\dot{\kappa}_p$, we had to make an approximation (see equation (2.8)) and we mentionned that an exact solution would require to solve a band-structured linear system for $\dot{\vec{\kappa}}$. However, one may bother about the disadvantage of this simplification. The answer can be given if we rewrite the functions used in the algorithm and compare the performance. But again, we want to minimize a function and in addition this process should be as fast as possible. Therefore, we should avoid too much computational effort unless this is useful to obtain a faster convergence. If we replace all approximations similar to equation (2.8) by a matrix solve, then such efforts are necessary to calculate the optimal descent direction, a step along a geodesic and the parallel translate of a vector.

Furthermore, one may think about the advantage of the explicit Euler method which is used in two functions. For sure, one may apply a Runge-Kutta-method, which yields a more accurate approximation. But this is also more expensive since we have to evaluate the right-

hand-side of the geodesic equations, or of the equations of parallel translation, about four times as much as for the explicit Euler method. Moreover, the situation is even more complicated if we do not have an explicit formula for $\dot{\vec{\kappa}}$ which we can evaluate, but if we have to solve a linear system for each point in the mesh in order to calculate $\dot{\vec{\kappa}}$.

In addition, one may see a slight analogy between the step length used for the explicit Euler steps and the *Courant-Friedrichs-Lewy-condition* (CFL-condition). In the context of partial differential equations solved with a finite-difference-scheme, this condition relates the step length Δt in time to the size Δx of the spatial discretization. This condition is a necessary condition for the convergence of the finite-difference-scheme. In one dimension and for explicit Euler steps, this condition reads

$$\frac{u\Delta t}{\Delta x} \leq 1$$

where u is the velocity of the system associated with the equation. For sure, we do not solve a partial differential equation, but nevertheless, we also use a discretized surface in space and discrete time steps. The step length in time which we use in the algorithms is essentially

$$\epsilon = \frac{\delta}{\|v\|}$$

where $\|v\|$ is the norm of the first time derivative of $(p)_{p\in P}$ and $\delta \leq 0.05$. And this satisfies the CFL-condition since the size of our space discretization is always greater or equal 0.05.

Bibliography

[1] T. Frankel. *The Geometry of Physics. An Introduction.* Cambridge University Press, 2001. [cited at p. 3]

[2] B. Horn. *Obtaining Shape from Shading Information*, chapter 4, pages 115–155. The Psychology of Computer Vision. McGraw-Hill, 1975. [cited at p. 2, 27]

[3] J. Jost. *Riemannian Geometry and Geometric Analysis.* Springer, 2005. [cited at p. 3]

[4] M. Kilian, N. J. Mitra, and H. Pottmann. Geometric Modeling in Shape Space. *SIGGRAPH*, 2007. [cited at p. 2, 12, 13, 53]

[5] R. Kimmel. *Numerical Geometry of Images. Theory, Algorithms and Applications.* Springer, 2003. [cited at p. 2, 27]

[6] P. Michor and D. Mumford. An Overview of the Riemannian Metrics on Spaces of Curves using the Hamiltonian Approach. *Applied and Computational Harmonic Analysis*, 23:74–113, 2007. [cited at p. 2, 12]

[7] W. Ring and B. Wirth. Optimization Methods on Riemannian Manifolds and their Application to Shape Space. *SIAM J. Optim.*, 22(2):596–627, 2012. [cited at p. 1, 33]

[8] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999. [cited at p. 2, 27, 28]

List of Symbols and Abbreviations

Abbreviation	Description	Definition
C	set of all pairs of neighbouring points of a triangulated mesh	page 14
M	a triangulated mesh	page 14
N	number of nodes of a triangulated mesh	page 13
n_p	local normal vector to a mesh at the point p	page 14
$\vec{\vec{n}}$	concatenation of the normal vectors to a triangulated mesh	page 14
$\mathcal{N}(p)$	points of a mesh neighbouring to the point p including p	page 14
P	set of all points of a triangulated mesh	page 14
$\mathcal{S}$	shape space of triangulated meshes in $\mathbb{R}^3$ with fixed connectivity	page 13
$\mathcal{T}(p)$	triangles of a mesh adjacent to the point p	page 14

List of Figures

List of Tables

Index