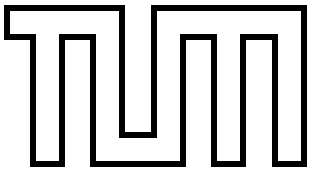


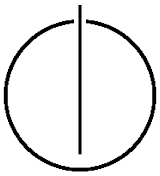
FAKULTÄT FÜR INFORMATIK

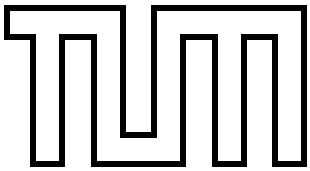
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor Thesis in Computer Science

A Comparison of Intrinsic Metrics on 2D-Manifolds

Frank Peter Schmidt





FAKULTÄT FÜR INFORMATIK

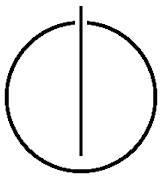
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor Thesis in Computer Science

A Comparison of Intrinsic Metrics on 2D-Manifolds

Ein Vergleich von Intrinsischen Metriken auf Zweidimensionalen Mannigfaltigkeiten

Author:	Frank Peter Schmidt
Supervisor:	Prof. Dr. Daniel Cremers
Advisor:	Dr. Emanuele Rodolà
Date:	September 15, 2014



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used. Munich, September 10, 2014 Frank Peter Schmidt

Acknowledgments

I thank Prof. Dr. Daniel Cremers from the TU Munich computer vision group for giving me the chance to write about this topic. Special thanks to Dr. Emanuele Rodolà and Matthias Vestner for their feedback, help and patience during the journey to the completion of this thesis. Finally, I thank Sara Heigl for her moral support, her efforts to review my thesis and her help with the illustrations.

Abstract

We compare the most common intrinsic metrics, namely the geodesic, the diffusion, the commute-time and the biharmonic distance. In order to do so, we first introduce the concept of metric spaces and apply it to the space of three-dimensional surfaces. After presenting the basic concepts of differential geometry on regular surfaces, we define the different metrics first in the continuous setting and then show ways to discretize them to triangular meshes. We further introduce our experiments to measure the qualitative performance of the intrinsic metrics on different meshes and give some insight into their implementation. Finally we find, that the biharmonic distance satisfies most of the preferable properties of an intrinsic metric, while the other distances have more specialized strenghts.

Zusammenfassung

The same in german/ Das selbe in Deutsch.

Contents

Outline of the Thesis

Part I: Introduction and Background Theory

CHAPTER 1: INTRODUCTION

This chapter presents an overview of the thesis and its purpose. Furthermore, it states the motivation of this thesis.

CHAPTER 2: SHAPES AS METRIC SPACES

The theoretic foundations of metric spaces are explained. After that, the Gromov-Hausdorff distance as a metric space on three-dimensional surfaces is introduced.

CHAPTER 3: DIFFERENTIAL GEOMETRY

The basic concepts of the differential geometry of three-dimensional surfaces are introduced. Then the notion of geodesic curves and the resulting geodesic metric are presented. This chapter finishes after introducing the Laplace operator on regular surfaces and the intrinsic metrics based on the Laplacians eigenfunctions and eigenvalues.

Part II: Implementation and Experiments

CHAPTER 4: TESTING PURPOSES AND PIPELINE

The experiments and ideas behind the tests of this paper are presented. Starting with the timing of the computation of the metrics, we present further experiments to measure their qualitative properties on triangulated meshes.

CHAPTER 5: IMPLEMENTATION DETAILS

This chapter gives an overview over the decisions and problems that occurred during the course of the implementation.

Part III: Results and Conclusion

CHAPTER 6: TESTING RESULTS

In this chapter, we describe the results of the experiments and state possible reasons for the performance of the different metrics.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This chapter finalizes the thesis by summarizing the results of this thesis and stating possible future research topics.

Part I.

Introduction and Background Theory

1. Introduction

Measuring the distances between pairs of points on a three-dimensional surface is one of the most classical problems in the field of computer graphics and shape analysis. Using this information, there is a wide field of applications to explore: From the segmentation of a surface into its basic components, the embedding of a shape into another space, to simplifying bigger problems to the deformation of a surface while retaining its properties through to the task to classify three-dimensional into different categories, possibly finding duplicate surfaces in different positions. Especially on the shape matching, there is a lot of work done up to today, some of it completely unrelated to distances but most of them are an instance of the minimum distortion correspondence problem. In other words, the problem searches for a correspondence between two surfaces which distorts their intrinsic properties (i.e. the distance between points) the least. In order to come up with a solution, the generally continuous three-dimensional shape is often given as a bounding surface which has to be discretized to a triangulated mesh so that we are able to run computations on it. To do the actual computations there are many different approaches, some examples are described in [20, 3, 16]. To provide some insight to this topic, we give a short introduction to the basics of metric spaces and how to compute the distortion of metrics between two shapes.

The main purpose of this paper is to compare the four most prevalent distance functions on three-dimensional shapes. The first one is the geodesic distance [24, 12], which measures the distance over the surface of the shape. Its contenders are the diffusion distance [23], the commute-time distance [10, 14] and the biharmonic distance [14] which are all based on the eigenfunctions of the Laplace-Beltrami operator on the respective surface. In practical applications, there are often similar shapes which have undergone rigid or isometric transformations, which means that the metric functions have to be at least invariant to those changes. Further desirable properties of the considered functions are to be:

1. locally isotropic: close to the source vertex, the metric should behave similar to the geodesic.
2. globally shape aware: reflect the overall shape of the surface when y is far from x.
3. insensitive to noise and topology: not changing significantly if noise or topological changes are added to the mesh.
4. parameter-free: the distance function does not depend on parameter to be set specifically for a given surface.
5. practical to compute: computation times of all distances between all points on common meshes should take at most a few minutes.
6. smooth: smooth with respect to perturbations of x and y; having no singularities except derivative discontinuity at the source point.

After introducing the different metrics, we present the testing pipeline and the specific information used. The objective of this paper is to show the properties and the effectiveness of the metrics on examples and possibly give a overview to their qualities, since often the functions are not rigorously explained in that aspect.

2. Shapes as Metric Spaces

The main question this chapter will be answering is, whether or not there exists such a thing as a "space of shapes" and if so, how to discern shapes from each other. To achieve this, we need the notion of a distance on a shape, in other words, we have to define a metric.

2.1. Metrics

We start by defining the most important point of this section, the metric space.

Definition 1 (metric space)**.** *A set M is called a metric space, if for each pair of points $x, y \in M$ there is a distance/metric function $d_M : M \times M \to \mathbb{R}_+ \cup \{0, \infty\}$ such that:*

- $d_M(x, y) = 0 \Leftrightarrow x = y$ *(identity of indiscernibles)*
- $d_M(x, y) = d_M(y, x)$ *(symmetry)*
- $d_M(x, y) \leq d_M(x, z) + d_M(z, y)\, \forall x, y, z \in M$ *(triangle inequality)*

In other words, if we can find a distance function which fulfills the given properties, every set can be a metric space. The ones most important to this thesis are the metric spaces defined on all three-dimensional shapes. But what are the interesting properties of such a metric space? For one, there then exists a way to measure how "close" one shape is to another by watching its individual points and comparing the behavior of the metric function over the shapes. One interesting term in this context is the isometry:

Definition 2. *A surjective, distance-preserving map f is called an isometry, where f being distance preserving means that for $f : X \to Y$ the equation $d_X(x, y) = d_Y(f(x), f(y))\, \forall x, y \in X$ holds. Two metric spaces (X, d_X) and (Y, d_Y) are isometric, if there exists such an isometry $f : X \to Y$.*

Thus, if a shape undergoes a isometric transformation, the distance retained. The more intuitive way of describing isometries is, that there are two instances of the same shape and one is changed a little, like for example by being rotated or scaled. There is also the notion of near-isometries for almost isometric shapes, like a human in two different poses (which are not isometric, since some bending and stretching of the surface occurs).

Another thing we are assuming from here on is that our metric spaces are compact. A metric space being compact implies that it is closed or complete and totally bounded, which is a reasonable call, since the surfaces we are interested in can mostly be approximated by 3D-meshes.

Definition 3 (sequentially compact)**.** *A subset $X \subset Y$ of a metric space Y is sequentially compact if every sequence in X has a convergent subsequence whose limit belongs to X. Explicitly this means that for each sequence (x_n) with $x_n \in X$, there exists a subsequence (x_{n_k}) such that for $k \to \infty$ the subsequence$(x_{n_k}) \to x$ converges to a point $x \in X$.*

Now there are a few interesting properties of metric spaces. If (X, d_X) is a metric space and $Y \subset X$, then a metric function for Y can be obtained by the restriction of d_X to the subset $d_Y = d_X|_Y$. This is a useful property if for example we wish to compare a part of a surface to the whole, the metric is unchanged on the partial surface. Moreover, X is called ambient space for Y and even though the restriction of d_X is the simplest, it is not the only way to define a metric on Y. In many cases, there is a more natural intrinsic metric, which means that it is not dependent on the ambient space. Now, to define the distance of a point $x \in X$ to a subset Y, the smallest distance between x and Y is used: $d_X(x, Y) = \inf_{y \in Y} d_X(x, y)$.

Example 2.1. *Let us consider the metric space $(\mathbb{R}^2, \|\cdot\|)$ with the euclidean metric and its subset $S \subset \mathbb{R}^2$ containing the points of the unit circle. The restriction of $\|\cdot\|$ to S would be a possible metric function, but a more intuitive, intrinsic metric is the minimal arc length between two points. It is easily to be seen, that the minimal arc length is in fact a metric, since it is symmetric, equal to zero if the points are equal and there is no shorter way over a third point and so the triangle inequality is also fulfilled. Additionally the arc length does not depend on the $\mathbb{R}^2$ coordinates of the points but on their position on the unit circle, what makes the minimal arc length an intrinsic metric. The two proposed metrics are also not isometric, as the distance between two opposing points $x, y \in S$ is different: $\|x, y\| = 1$ while $minarclength(x, y) = 2\pi$.*

So if we can define some sort of distance function between shapes, we could distinguish between shapes with small differences (e.g. two humans in different positions) and shapes with big differences like a human and a car.

2.2. Gromov-Hausdorff distance

The most commonly used metric to differentiate three-dimensional shapes is the Gromov-Hausdorff distance. But before we begin working on the Gromov-Hausdorff distance, we need to find a intuition of what it means for surfaces to be "close" to each other. Smooth surfaces in $\mathbb{R}^3$ can (at least locally) be parametrized by a domain $U \subset \mathbb{R}^2$ for example by an embedding $f : U \to \mathbb{R}^3$. Two parametrizations then specify a homeomorphism from one surface to the other. Those two surfaces are said to be "close" if the homeomorphism only slightly changes some properties, like distances or derivatives, of the surfaces.

To start off, we consider the problem of comparing two shapes in the same metric space. For that the Hausdorff distance as seen in figure 2.1 is used.

Definition 4 (Hausdorff distance)**.** *The Hausdorff distance between two compact subsets $X, Y \subset (Z, d_Z)$ is defined by*

$$d_{\mathcal{H}}^Z(X, Y) = \max \left\{ \sup_{x \in X} d_Z(x, Y), \sup_{y \in Y} d_Z(y, X) \right\}.$$

$d_{\mathcal{H}}^Z(X, Y)$ is a semi-metric on the space of compact subsets of a metric space.

Even though it is called distance, the Hausdorff distance is only a semi-metric since there are cases, where $d_{\mathcal{H}}^Z(X, Y) = 0 \Leftrightarrow X = Y$ does not hold. Take for example the set $X = (0, 1)$ and its closure $Y = [0, 1]$. So the biggest distance should be between either 0 or

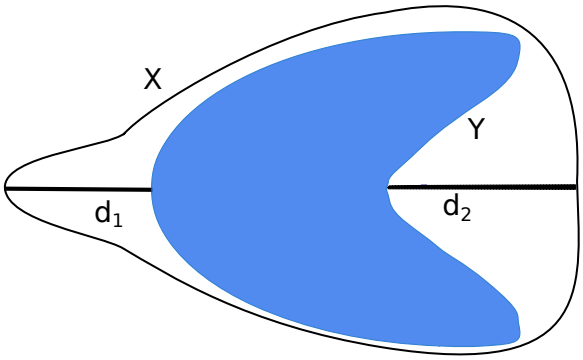


Figure 2.1.: A visualisation of the two relevant distances of the Hausdorff distance: $d_1 = \sup_{x \in X} d_Z(x, Y)$ while $d_2 = \sup_{y \in Y} d_Z(y, X)$. The Hausdorff distance is defined as the bigger one, d_2 in this case.

1 and X which in both cases is zero although $X \neq Y$. Also note, that even one stray point can make the Hausdorff distance arbitrarily large.

But with the Hausdorff distance, we can only find the distance between sets which are subsets of a common metric space. To be able to define a distance between different metric spaces X, Y, the basic idea is to use a isometric embedding into another metric space Z and use the Hausdorff-distance there. The Gromov-Hausdorff distance $d_{\mathcal{GH}}(X, Y)$ is defined as the minimum r for which such an embedding space Z and the isometric embeddings $X', Y' \subset Z$ exist with $d_{\mathcal{H}}^Z(X, Y) < r$.

Definition 5 (Gromov-Hausdorff). *The Gromov-Hausdorff distance between two metric spaces X and Y is defined by*

$$d_{\mathcal{GH}}(X, Y) = \inf_{Z, f, g} d_{\mathcal{H}}^Z((f(X), g(Y))$$

taken over all ambient spaces Z and isometric embeddings $f : X \to Z, g : Y \to Z$.

$d_{\mathcal{GH}}(X, Y)$ is a metric on the space of isometry classes of compact metric spaces thus we can say that there exists a "space of shapes". Note however, that from a practical perspective, the possibilities for choosing Z, f and g are far to huge to simply compute the Gromov-Hausdorff distance directly without further restrictions.

To begin with, we restrict the search space to possible correspondences between x and Y.

Definition 6 (correspondence). *A correspondence between two sets X and Y is a set $R \subset X \times Y$ which satisfies*

- *for all $x \in X$ there exists $y \in Y$ such that $(x, y) \in R$.*
- *for all $y \in Y$ there exists $x \in X$ such that $(x, y) \in R$.*

Note that a correspondence neither has to map one point to exactly one other nor does every point have to be a part of the correspondence. A correspondence R is associated to a map $f : X \to Y$ if $R = \{(x, f(x)) : x \in X\}$ contains the image of f. Our next target is

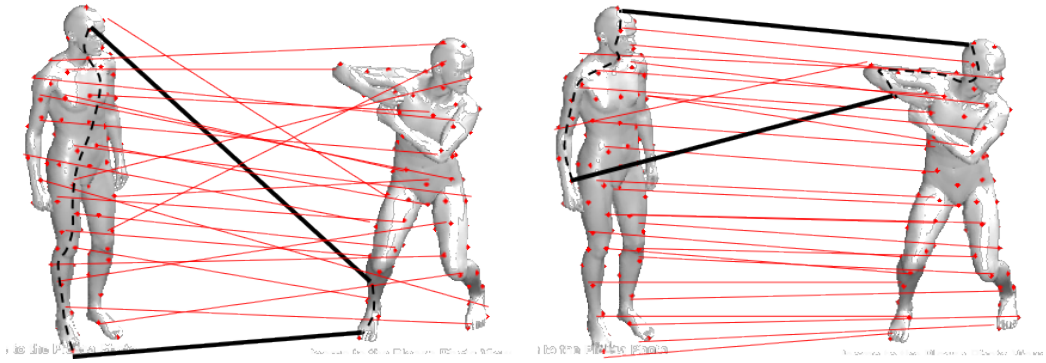


Figure 2.2.: The figure shows two different given correspondences. For the black correspondences, additionally the shortest path between them is given. Since the metric distortion measures the change of these distances by the correspondence, we can see that the distance on the left shape is not being preserved, while the distance on the right set of meshes is preserved almost perfectly. Therefore the correspondence on the left side results in great metric distortion, whereas the right correspondence will have a lower distortion, based on the given distances.

to show that the Gromov-Hausdorff distance can be alternatively defined by using correspondences between two metric spaces. Remember that the Gromov-Hausdorff distance was defined to measure the difference between two metric spaces. Now we can define the distortion between two given metric spaces by using a correspondence.

Definition 7 (metric distortion)**.** *Let (X, d_X) and (Y, d_Y) be compact metric spaces and $R \subset (X, d_X) \times (Y, d_Y)$ be a correspondence. Then the distortion of R is defined as*

$$dis\, R = \sup\{\, |d_X(x, x') - d_Y(y, y')| : ((x, y), (x', y') \in R)\}. \tag{2.1}$$

How the distortion of a metric shows on meshes can be seen in figure 2.2. By the definition of isometry, the distortion of R is zero if and only if the map associated with R is an isometry. This leads to another definition of the Gromov-Hausdorff distance.

Definition 8 (Gromov-Hausdorff distance (2))**.** *The Gromov-Hausdorff distance between two metric spaces X and Y is defined by*

$$d_{\mathcal{GH}}(X, Y) = \frac{1}{2} \inf_{R \subset X \times Y} dis\, R. \tag{2.2}$$

In this formulation, the Gromov-Hausdorff distance is defined as the infimum of $r > 0$ for which there is a correspondence with $dis\, R = 2r$. And unlike the first definition, we now minimize over a finite set of correspondences instead of all possible ambient spaces and their corresponding embeddings.

Another improvement is to use open ball coverings of the metric spaces. This means that instead of using all points within a metric space, we choose representatives such that they cover all points on the mesh.

Definition 9 (open ball covering). *Let $x \in (X, d_X)$ be a metric space. An open ball of radius $r > 0$ centered at x is defined by*

$$B_X(x, r) = \{z \in X : d_X(x, z) < r\}.$$

For a subset $A \subset X$, we define the open ball as

$$B_X(A, r) = \bigcup a \in A B_X(a, r).$$

A set $C \subset X$ is a r-covering of X if $B_X(C, r) = X$.

The interesting point of this is that the r-covering of a shape is close to the original shape in a Gromov-Hausdorff sense. Let $\{x_1, \ldots, x_n\} \subset X$ be a r-covering of a compact metric space (X, d_X). Then $d_{\mathcal{GH}}(X, \{x_1, \cdots, x_n\}) \leq r$ holds as stated in [5], which means that $d_{\mathcal{GH}}$ is consistent to sampling. This further leads to us only having to compute $d_{\mathcal{GH}}$ for a dense enough r-covering of our metric spaces X and Y to have a good approximation of its behavior in the originally continuous spaces. Also, this shows that the approximation of shapes by triangulated meshes, which more or less are point samples, is well posed in this context up to a certain error.

Since it is not easy to compute an optimal covering, the farthest point sampling was devised. It is efficient to compute, once the metric function is available and is at most worse than an optimal sampling by a factor of two. Its basic idea is to iteratively select points from the metric space, until the needed number of points is reached. The algorithm is initialized by either choosing a arbitrary point or by choosing two points which are the farthest away from each other and declaring them as our initial covering C_0. After that, the algorithm selects a point x fulfilling the equation $\operatorname{argmax}_x d_X(x, C)$ in each step, adding it to the previously selected $C_k = C_{k-1} \cup \{x\}$. The result is a progressively denser sampling of the shape which in general is not unique but "almost" optimal. This can then be used to create a Voronoi sampling of the shape by assigning all points to their closest sampled point $x \in C_k$.

Minimizing the Gromov-Hausdorff distance is a common approach followed in shape analysis. In order to reduce the problems complexity, the equation from (2.2) can be relaxed to a quadratic assignment problem [15], for which there exist a number of optimization algorithms.

3. Differential Geometry

3.1. Regular Surfaces and the First Fundamental Form

Since all of the metrics in this thesis are based on the fact that 3D shapes can be modeled as regular surfaces the $\mathbb{R}^3$, this section will give some insight into the basics of the math behind them. For all practical purposes these regular surfaces are approximated by piecewise linear triangular meshes, consisting of vertices and edges. To begin with, we narrow down our choice of meshes to those, which are 2-dimensional manifolds without a boundary, so that we do not have to worry about boundary conditions. The answer to the question, how a complex mesh can be represented mathematically, is given by the second chapter of [8]: The intuitive answer is to take pieces of a plane and to deform and arrange them, such that they cover the whole surface. The only constraints to ensure the ability to apply differential geometry are, that the resulting surface is not allowed to have sharp points/edges or self-intersections. A more rigorous definition is the following:

Definition 10 (regular surface)**.** *A subset $S \subset \mathbb{R}^3$ is a regular surface if, for each $p \in S$, there exists a neighborhood $V \in \mathbb{R}^3$ and a map $x : U \to V \cap S$ of an open set $U \subset \mathbb{R}^2$ onto $V \cap S \subset \mathbb{R}^3$ such that:*

1. *x is differentiable. That means that if we write*

$$x(u, v) = (\mathbf{x}(u, v), \mathbf{y}(u, v), \mathbf{z}(u, v)), \quad (u, v) \in U,$$

   the functions $\mathbf{x}(u, v), \mathbf{y}(u, v), \mathbf{z}(u, v)$ have continuous partial derivatives of all orders in U.

2. *x is a homeomorphism. Since x is continuous by condition 1, this condition requires that x has an inverse $x^{-1} : V \cap S \to U$ which is continuous.*

3. *For each $q \in U$, the differential $dx_q : \mathbb{R}^2 \to \mathbb{R}^3$ has full rank.*

This definition states, that for every point on the surface there exists a neighborhood with a map x which maps from the parameter domain $U \subset \mathbb{R}^2$ to S (in correspondence to the plane patch from earlier) and the collection of all those neighborhoods defines the regular parametrized surface as can be seen in figure 3.1.

Remark 3.1. *The first condition ensures that we can use differential geometry on the regular surface, while the second one restricts the surface, such that it can not have intersections with itself. Even though the differential of the map dx_p is covered later in this section, it can already be told that the third condition is responsible for the surface only having non-degenerated tangent planes.*

One of the important points of regular surfaces is, that its local properties do not change, if we use another parametrization. Thankfully, it has already been proven, that there always exists a diffeomorphism that can transfer points from one parameter domain to the

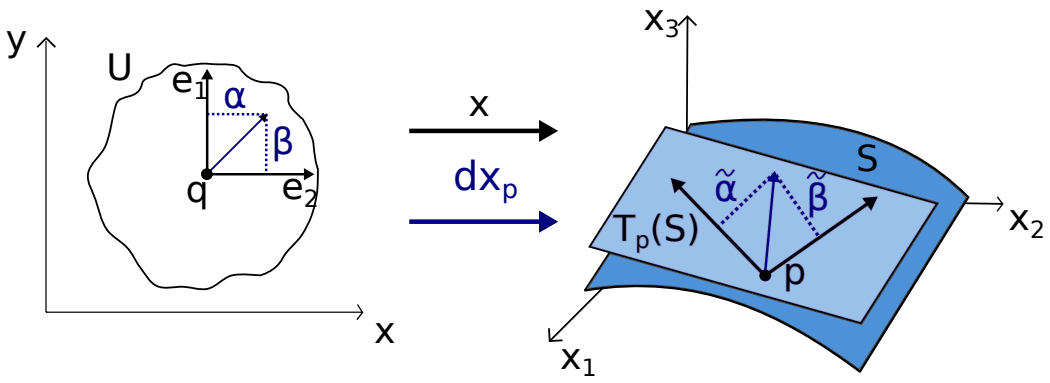


Figure 3.1.: A graphical example of a regular surface: The points of the parameter domain U are mapped to the surface S by the parametrization x with $p = x(q)$. The derivative of the map dx_p maps tangent vectors of U (in the basis e_1, e_2) to tangent vectors on S (in the basis $dx_p(e_1), dx_p(e_2)$.

other and therefore make different maps interchangeable: Let p be a point on a regular surface S and $x : U \to S$ and $y : V \to S$ be parametrizations of S with $p \in x(U) \cap y(V) = W$. Then the change of coordinates $h : y^{-1}(W) \to x^{-1}(W), h = x^{-1} \circ y$ is differentiable and has a differentiable inverse.

Now that we have defined regular surfaces, the next step is to find a way to express functions defined on the surface. Since we already know that a regular surface is just a image of a two dimensional parameter domain, we can use that knowledge to define our function there.

Definition 11 (differentiable function on surfaces)**.** *Let f be a function from an open subset $V \subset S$ of a surface S to $\mathbb{R}$. Then f is called differentiable at $p \in V$ if, for a parametrization $x : U \to S, p \in x(U) \subset V$, the composition $\tilde{f} = f \circ x$ is differentiable at $x^{-1}(p)$. f is differentiable in V, if it is differentiable at all points of V.*

This means we can now transfer functions from one domain into the other, given a parametrization.

After defining how to transcribe functions between the parameter domain and the actual surface, we will now continue with the behavior of vectors under the differential map x. Let us begin with the best visual concept of differentiating a map, the tangent plane. The concept of a tangent plane is that at a certain point on the surface, the tangent vectors of all differentiable parametrized curves through that point constitute a plane.

Definition 12 (tangent plane)**.** *Let $x : U \subset \mathbb{R}^2 \to S$ be a parametrization of a regular surface S and let $q \in U$. The vector subspace of dimension 2, $dx_q(\mathbb{R}^2) \subset \mathbb{R}^3$, coincides with the set of tangent vectors to S at $x(q)$ and is called tangent plane to S at $p = x(q)$ and will be denoted by $T_p(S)$.*

Notice, that the above definition of the tangent plane is not dependent on the parametrization x. The parametrization only determines the basis $(\partial x/\partial u)(q), (\partial x/\partial v)(q)$ of the tangent plane $T_p(S)$. From here on out, they are abbreviated as $(\partial x/\partial u) = x_u$ and $(\partial x/\partial v) = x_v$. Before we take a look at vectors specifically, we have to understand the concept of the differential of a map which was already used a few times in this thesis:

Definition 13 (differential of a map)**.** *Let $x : U \to S$ be a differential map and let $\alpha : (-\epsilon, \epsilon) \to U$ be a differential curve in the parameter domain with $\alpha(0) = q, \alpha'(0) = w$. The composition of them is called $\beta = x \circ \alpha$. Then the differential of x at q is*

$$dx_q(w) = \beta'(0).$$

Since β is the image of α on the surface S, dx_q maps tangent vectors from the parameter domain to tangent vectors of the surface by definition. Also, dx_q is not dependent on α but is solely a property of the differential map x and it is a linear map. This can be seen, if we apply the chain rule to the definition of dx_q where (u, v) are coordinates in U and (x, y, z) are coordinates in $\mathbb{R}^3$:

$$dx_q(w) = \beta'(0) = (x \circ \alpha)'(0) = x'(\alpha(0)) \cdot \alpha'(0) = \begin{pmatrix} \partial(x)/\partial(u) & \partial(x)/\partial(v) \\ \partial(y)/\partial(u) & \partial(y)/\partial(v) \\ \partial(z)/\partial(u) & \partial(z)/\partial(v) \end{pmatrix} \begin{pmatrix} \partial u/\partial t \\ \partial v/\partial t \end{pmatrix}$$

Notice that the first matrix indeed does not depend on a specific α.

The last point in this section is the introduction of the first fundamental form, which is a replacement for the natural inner product on a surface S.

Definition 14 (first fundamental form)**.** *The quadratic form $I_p : T_q(s) \to \mathbb{R}$ on $T_q(S)$, defined by*

$$I_p(w) = \langle w, w \rangle_p = |w|^2 > 0,$$

is called the first fundamental form of the regular surface $S \subset \mathbb{R}^3$ at $p \in S$.

It can be used to determine geometric values like lengths of curves and areas of regions on regular surfaces without referring back to the ambient space.

Given a certain basis x_u, x_v associated to a parametrization $x(u, v)$ at p, we can express the first fundamental form in that basis. Since a tangent vector $w \in T_p(S)$ is the tangent vector to a parametrized curve $\alpha(t) = x(u(t), v(t)), t \in (-\epsilon, \epsilon)$, with $p = \alpha(0) = x(u_0, v_0)$, we obtain

$$\begin{aligned} I_p(\alpha'(0)) &= \langle \alpha'(0), \alpha'(0) \rangle_p \\ &= \langle x_u u' + x_v v', x_u u' + x_v v' \rangle_p \\ &= \langle x_u, x_u \rangle_p (u')^2 + 2\langle x_u, x_v \rangle_p u'v' + \langle x_v, x_v \rangle_p (v')^2 \\ &= E(u')^2 + 2Fu'v' + G(v')^2 \\ &= \begin{pmatrix} u' & v' \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} \end{aligned}$$

where the values of the functions involved are computed for $t = 0$ and E,F and G are dependent on the basis x_u, x_v of $T_p(S)$. By letting p run in the coordinate neighborhood corresponding to $x(u, v)$ we obtain functions $E(u, v)$,$F(u, v)$ and $G(u, v)$ which are differentiable in that neighborhood. These coefficients play important roles in many intrinsic properties of the surface. Now we can even use two different vectors $(\alpha, \beta), (\delta, \gamma) \in T_p(S)$ to insert into the matrix notation to get the more general bilinear form of the first fundamental form:

$$I_p((\alpha, \beta), (\delta, \gamma)) = \begin{pmatrix} \alpha & \beta \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} \delta \\ \gamma \end{pmatrix}$$

One nice property of the first fundamental form is that it is invariant to isometries [8], which makes it a good candidate for a base of an intrinsic metric.

3.2. Minimal Geodesics on Surfaces

One of the most fundamental metrics is the idea of constructing the shortest path between two points and define the length of that path as the distance between those two points. In the Euclidean domain, this metric is defined by the Euclidean norm of the vector joining the two points. On a regular surface, defining this distance is not as easy, since a lower dimensional set is embedded into a higher dimension. The basic idea is, to find the shortest path along the surface from one point to the other and take its length as the distance. It is commonly known, that such a path is a straight line and its generalization to curved surfaces is called a minimal geodesic, which will be defined during the course of this section.

Since a geodesic is definitely some kind of curve, we start by introducing the core concepts needed for the definition of a geodesic.

Definition 15 (parametrized curve)**.** *A parametrized curve α is the restriction of a differentiable mapping $(0-\epsilon, l+\epsilon) \to S), \epsilon > 0$ to the interval $[0, l]$. The curve α is called to join two points $p, q \in S$ if and only if $\alpha(0) = p$ and $\alpha(l) = q$ and it is called regular, if its derivative $\alpha'(t)$ is nonzero for $t \in [0, l]$.*

Now let $w(t)$ be a vector field along the curve α. Then w is called differentiable, if for the parametrization $x(u, v)$ the vector field $w(t)$ can be written as $w(t) = a(t) \cdot x_u + b(t) \cdot x_v$, where a and b are differentiable functions. Another important part is the covariant derivative of a vector field.

Definition 16 (covariant derivative)**.** *Let $\alpha(t)$ be a parametrized curve on S with $\alpha(0) = p \in S, \alpha'(0) = y \in T_p(S)$ and a differential vector field $w(t)$ constricted to α. The normal projection of the derivative of w in respect to time $\frac{dw}{dt}(0)$ onto the tangent space $T_p(S)$ is called the covariant derivative at p of the vector field w relative to y:$\frac{Dw}{dt}(0)$*

The covariant derivative is well-defined for differentiable vector fields and is furthermore intrinsic. This can be seen, if we look at the expression for the covariant derivative:

$$\frac{Dw}{dt} = (a' + \Gamma^1_{11}au' + \Gamma^1_{12}av' + \Gamma^1_{12}bu' + \Gamma^1_{22}bv')x_u + (b' + \Gamma^2_{11}au' + \Gamma^2_{12}av' + \Gamma^2_{12}bu' + \Gamma^2_{22}bv')x_v$$

where $(u'v') = y$ and the Γ are the so called Christoffel symbols, which are only dependent on first fundamental form and therefore $\frac{Dw}{dt}$ is an intrinsic property of the surface as shown in chapter 4.3 of [8]. A geometric interpretation of the covariant derivative would be the second derivative of the vector field w as seen from the surface.

Definition 17 (parallel vectorfield)**.** *The vector field w along the parametrized curve α is called parallel if it satisfies*

$$\frac{Dw}{dt} = 0$$

for all points on α.

An example for parallel vector fields can be seen in figure 3.2.

Now every aspect of the following definition has been introduced:

Definition 18 (geodesic curve)**.** *A non-constant, parametrized curve $\gamma : I \to S$ is called geodesic at $t \in I$ if the field of its tangent vectors $\gamma'(t)$ is parallel along γ at t. Consequently, the curve γ is called geodesic, if $\frac{D\gamma'}{dt} = 0 \forall t \in I$. [8, 238-246]*

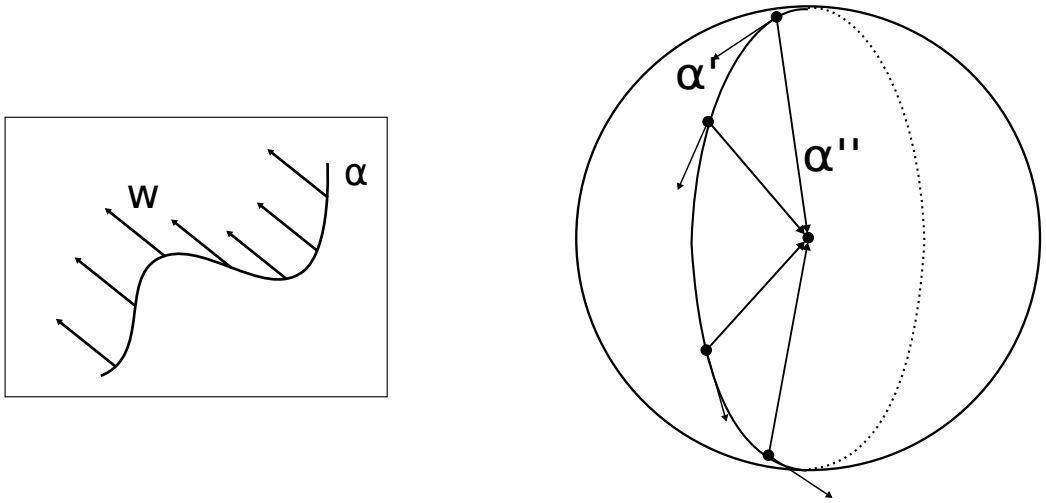


Figure 3.2.: Left: parallel vector field w of a curve α. Right: The second derivative of a curve α on the meridian of a sphere is normal to the surface of the sphere and therefore is a parallel vector field.

To be a minimal geodesic, the curves length has to be less or equal to the length of any other piecewise regular curve on the surface.

Example 3.2. *If we look at the sphere S^2, its geodesic curves are obtained by intersecting the sphere with a plane passing through the center point of the sphere. This results in circle-shaped curves whose tangential vector field is parallel, since it is always pointing into the same direction if seen from the surface (see figure 3.2). So there are at least two geodesics joining two points p_1 and p_2, just by following the intersection in different directions, starting from p_1. On the S^2 the minimal geodesic would be either the shorter arc joining p_1 and p_2 or, if they are antipodal points like the north and the south pole, there is an infinite number of minimal geodesics joining p_1 and p_2.*

On the other hand, the existence of a minimal geodesic is not granted: Let p be a point on the minimal geodesic joining the points p_1 and p_2 (which are not a pair of antipodal points/sufficiently close) on the sphere S^2 and let the surface be $S^2 - p$. Then there exists no minimal geodesic between p_1 and p_2, since the only other geodesic joining them goes the long way around and is therefore longer than a piecewise regular curve almost equal to the minimal geodesic on S^2 except for going around the hole at p. One way to ensure the existence of a minimal geodesic is to constrain the surfaces to have certain properties. In general, most of the surfaces examined are both closed and bounded which means, they are compact. As shown in [8, 331-332], compact surfaces are complete and we can use the theorem of Hopf and Rinow:

Theorem 3.3. *"Let S be a complete surface. Given two points p, q, there exists a minimal geodesic joining p and q." [8, 333]*

Hence we can safely assume that there exists a minimal geodesic on the regular surfaces considered. This approach is not very practical on actual meshes, since it needs a parametrization of the surface and comparing the length of all possible geodesic curves is not feasible.

Discretisation to triangulated meshes

There are two main ways to compute shortest geodesic paths on triangulated meshes: One is solving the Eikonal equation with the Fast Marching Method to approximate geodesics on the mesh as proposed by Kimmel and Sethian in [12]. The second way, which will be used later in this paper, has been proposed by Mitchel, Mount and Papadimitriou (MMP) in 1987. Their basic ideas are presented here, for further information, we refer to [24]. The fundamental idea is to use a simple parametrization of the geodesic distance on each edge and propagate the distance information starting from the source point over the whole surface. Take note, that geodesics on a triangular mesh need to have two properties:

- They need to be straight lines within each face.
- When crossing over an edge, the shortest paths need to correspond to a straight line if the faces are unfolded into a common plane.

Additionally, there are two kinds of vertices which need additional attention: boundary vertices and saddle vertices, which have a total angle greater than 2π. With this in mind, we can take a look at the MMP algorithm. The main element of this are the so called windows, which bundle multiple shortest paths that traverse an edge into the same direction, into one tuple of six parameters.

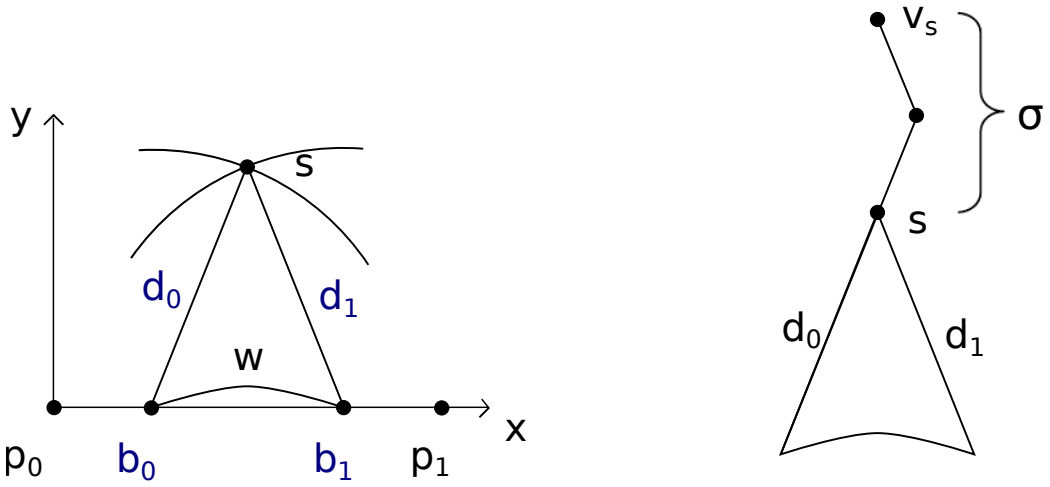


Figure 3.3.: Visualization of the window parameters

The window parameters Let's first assume, that the shortest path from the source vertex v_s to the point p does not pass through any boundary or saddle vertices. In that case, all traversed triangles can be unfolded into a common plane, so that the path is a straight

line in that plane. If we now consider some neighboring points of p whose shortest paths pass through the same sequence of faces, we get at set of straight lines emanating from v_s intersecting the same edges. So these paths are combined into one window and it is saved for the edge e by first defining its width through determining the beginning and the end point of the window by $b_0, b_1 \in [0, ||e||]$. Additionally, the relative position of the source vertex to the window is encoded by the distance d_0, d_1 to the two window endpoints and a binary direction τ. In the case, that the path passes through one or more saddle or boundary vertices, let s be the one closest to p. All paths of p and its neighboring points pass through s, which therefore is a (pseudo-)source for them. The window now stores the distance information to s as its source and an additional parameter that contains the distance of s from v_s: $\sigma = D(s)$. So the distance field D over the window is described by the tuple $(b_0, b_1, d_0, d_1, \sigma, \tau)$ which are visualized in figure 3.3.

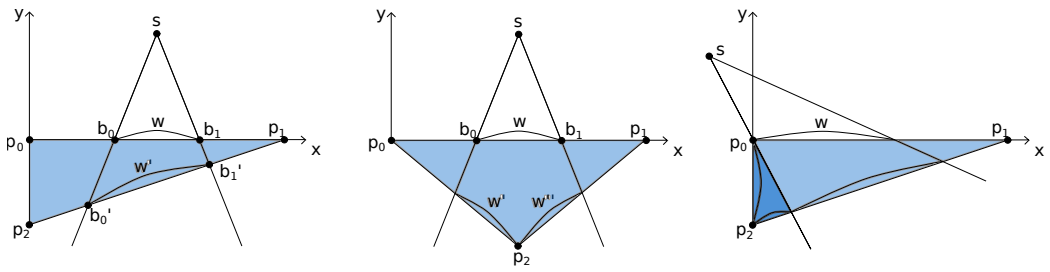


Figure 3.4.: Visualization of the three possible situations during the propagation step.

Window propagation To compute the distance function over the whole mesh, the windows are propagated over it. Given a window w on the edge e_1, the distance field will be propagated over a adjacent face f, resulting in new windows on the opposing edges e_2, e_3 of that face. Since there are possibly already existing windows on those edges, we later need to intersect them with the existing ones and only keep the information of the shortest distances. After again unfolding the mesh into a common plane, we extend the connections of the endpoints of w, b_0 and b_1, and the source s until they intersect with one of the opposing edges. This results in either one or two new windows, depending on whether or not there is an vertex in between the two intersection points. Now we already have the values of b_0', b_1' of the new window w' and only need to compute the new distances between s and the new endpoints to obtain d_0' and d_1'. The pseudosource distance $\sigma' = \sigma$ stays the same and the direction τ is assigned to point into the face f.

The special case of w being adjacent to a saddle or boundary vertex v results in a few additional windows. Since shortest paths may pass through v, we need to add windows to the parts of the face, which can be reached through v and are not already taken care of by the preceding steps. As seen in figure 3.4, if p_0 is a saddle or boundary vertex and part of the window, the algorithm first generates the regular window on the edge $\overline{p_1 p_2}$. But additionally it adds windows to the darker blue part of the triangle, since those are the parts of the face, which can probably only be reached by passing through p_0. These windows will have the pseudosource p_0 and $\sigma = D(p_0)$ is set accordingly. This scenario is treated in a symmetric manner for the mirrored case where p_1 is part of the window.

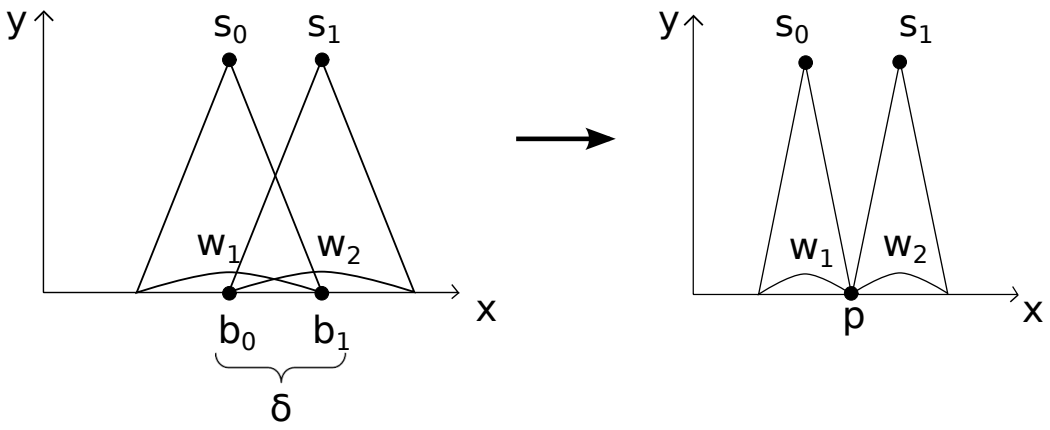


Figure 3.5.: Left: initial situation before the intersection. Right: the starting and ending points of the two windows have been adjusted to the point p

Intersection of windows Once all new windows were created, we can intersect them with the existing ones, resulting in windows with the minimal distance. Let δ be the non-empty intersection interval of two windows w_1 and w_2. In the simple case, that one of the windows has a larger distance all over δ, then δ is just cut away from its interval. This leaves only intersections, were one window has a shorter distance on one end, while the other window is nearer to the source on the other end (figure 3.5. To intersect those two windows, we need to find the point $p \in \delta$ in between them, which is exactly as far away from the source in one window, as it is in the other. This means, that p has to fulfill the following condition:

$$||s_0 - p|| + \sigma_0 = ||s_1 - p|| + \sigma_1$$

This equation can be reduced to a quadric which has only one solution with the constraint $p \in \delta$. Finally, we only need to adjust the boundaries b_1 of w_1 and b_0 of w_2 to match the point p and recompute the endpoint distances.

Propagation order The described algorithm propagates the distance information from the source point outwards in a Dijkstra-like manner. This means, that every time, a window is created or modified, it will be placed in a priority queue from which then the first window is taken and propagated across a face next. Even though the algorithm computes the same result even without an specific ordering of the queue, the performance is increased, if the queue is ordered by the windows distance from the source. Notice, that windows can be added, modified and removed in one step and the queue has to be updated accordingly.

During initialization, on each edge adjacent to the source vertex v_s a window is created. The distance fields on those windows are trivial, e.g. the distance every vertex adjacent to v_s is equal to the edge length. After that, one window is popped of the priority queue in each step and processed.

Construction of geodesic paths After the distance information was propagated throughout the whole mesh, it is easy to find the shortest or geodesic path from any given point p

to the source s. In the most general case of p lying on the interior of a face, we first collect all the windows on its edges and minimize the length of the path going from p through the window to the source. So in other words, we minimize $||p - p'|| + D(p')$ for all p' within those windows.

The next step is to trace back the path from p' to s. To do this, we take the direction τ of the window containing p' and locate its pseudo-source v_s using the distance information provided. The next point in the backtracing process is obtained by intersecting the line $\overline{v_s p'}$ with the opposite edges of the face. Now the algorithm is repeated with the window containing the intersection.

A special case again are pseudosources, which are boundary or saddle vertices: To trace them back, we look at the adjacent windows and iteratively go through them, until we find one with a pseudosource different from the boundary or saddle vertex we are currently looking at. Using that knowledge, we can compute the minimal geodesics and their lengths on triangulated surfaces like it can be seen in figure 3.6.

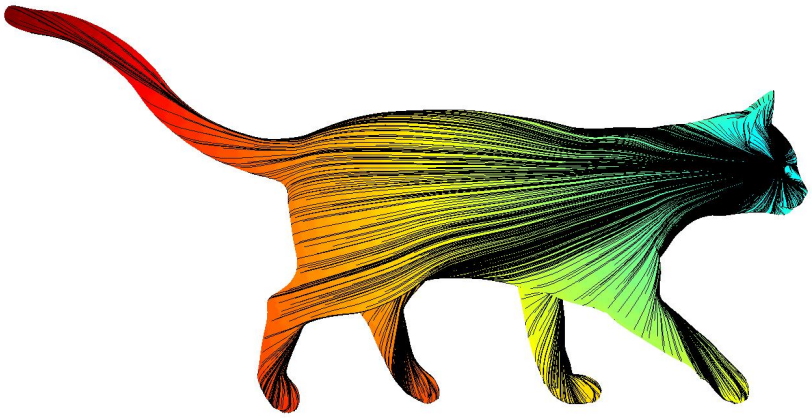


Figure 3.6.: A visualization of geodesic paths on the mesh of a cat. The plotted paths are the minimal geodesics connecting one point on the head to the other points.

3.3. The Laplacian and Metrics based on its Eigenfunctions

3.3.1. The Laplacian on regular surfaces

The Laplace-Beltrami operator (or Laplacian for short) on a regular surface is the restriction of the general Laplace operator onto the surface. Geometrically, the Laplacian can therefore be seen as the second derivative of a function on a surface, in analogy to the one-dimensional case of the Laplace operator in Euclidean space $\Delta f = f''$. But the point we are interested in is that its eigenfunctions are orthogonal to each other and, even more interesting, the Laplace-Beltrami Operator only depends on the first fundamental form and therefore is invariant to isometries.

Definition

Since the Laplace operator is defined as the divergence of the gradient of a function f,

$$\Delta f = div(\nabla(f))$$

we first need to prove that those two functions are well defined on a regular surface.

The Gradient In Euclidean space $\mathbb{R}^n$, the gradient is defined as

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right).$$

The resulting vector points into the direction of the greatest increase of the function and its magnitude corresponds to the slope of the function. By the Riesz Representation Theorem, it is also used to define the directional derivative $df_p(v)$ of f at the point p in the direction of the vector $v \in \mathbb{R}^n$:

$$\langle \nabla f(p), v \rangle = df_p(v) \tag{3.1}$$

Another way to define the directional derivative $df_p(v)$ is to watch its behavior over time:

$$df_p(v) = \frac{d}{dt}_{|t=t_0} f(p + tv)$$

or, in a more general way, by using $\gamma : (-\epsilon, \epsilon) \to \mathbb{R}^n$ fulfilling $\gamma(0) = p, \dot{\gamma}(0) = v$

$$df_p(v) = \frac{d}{dt}_{|t=t_0} f(\gamma(t)).$$

Now the second equation can be used on a regular surface S, since it is not necessary to use $f(p + tv)$ which could possibly go out of S but any γ. If the directional derivative $df_p(v)$ of $f : S \to \mathbb{R}^n$ and $v \in T_p(S)$ is calculated, a γ satisfying the conditions exists by the construction of $T_p(S)$, so that $df_p(v)$ exists and can be used to define the gradient on the manifold. After converting (3.1) into the arithmetics of regular surfaces, it looks like this:

$$I_p(\nabla f(p), v) = df_p(v) \text{ for } \forall v \in T_p(S)$$

This equation implies that exactly one ∇f exists for $v \in T_p(S)$ to satisfy the equation above.

The Divergence The divergence operator div converts a vector field $V : S \to T_p(S)$ into a scalar field. Even though the formula for doing so in the $\mathbb{R}^n$

$$div\, V = \sum_{i=1}^{n} \frac{\partial V_i}{\partial x_i} \tag{3.2}$$

seems complex, the idea behind it is simple: It basically interprets the vector field as the definition of a flow, whose direction and volume in a certain point is defined by the corresponding vector. The operator will then compute the current volume change under the

influence of V. It can be proven, that for a function f with a compact support and a vector field V, the divergence is the negative adjoint operator to the gradient:[11]

$$\langle V, \nabla f \rangle = \langle -div\, V, f \rangle \tag{3.3}$$

Since we already know that the first fundamental form of our surface is the equivalent of the inner product in Euclidean space and there exists a unique gradient for every f on S, there also exists a unique function fulfilling the above equation.

Definition 19 (Laplace-Beltrami operator)**.** *Let S be a regular Surface with $x : U \subset \mathbb{R}^2 \to S$ as its parametrization and $T_p(S)$ being its tangent plane. Then the Laplace-Beltrami operator is defined as*

$$\Delta f = div(\nabla(f))$$

Combining our knowledge about the gradient and the divergence together, we come to the conclusion that the Laplace-Beltrami-Operator is well-defined on regular surfaces. It can even be expressed in the parameter domain of S, but again, since we are working on actual meshes, we will now take a look at the discretization of our new defined operator on triangular meshes.

Discretization of the Laplacian and its eigenfunctions

Since the Laplacian maps functions to functions, we first take a look at the way functions are represented on a mesh based on the approach in [6]. Generally, a function f is represented by its values at each vertex of the mesh. The missing information about the function values inside the faces can be filled in by interpolation, for example by assuming that the function behaves linearly inside of every triangle. To achieve this, we use the finite element method, which approximates a function f by breaking it into a finite set of basis functions $\{\phi_i\}$. The resulting representation of f is a linear combination of the basis functions $\tilde{f} = \sum_i f_i \phi_i$ with weights $f_i \in \mathbb{R}$. For triangulated meshes, the most natural choice of basis functions are the piecewise linear hat functions ϕ_i which equal one at their associated vertex and zero at all other vertices.

Computation of the Laplacian Next, we define the function $h = \Delta f$ and get the discretized representation $h = \sum_i h_i \phi_i$. One could think that the Laplacian should vanish, since it is a second derivative and all hat functions ϕ_i are linear. But due to Green's identity, by breaking up the integral into a sum over all triangles σ, it holds that

$$\langle \Delta f, \phi_j \rangle = \sum_k \langle \Delta f, \phi_j \rangle_{\sigma_k} \tag{3.4}$$

$$= \sum_k \langle \nabla f, \nabla \phi_j \rangle_{\sigma_k} + \sum_k \langle N \cdot \nabla f, \phi_j \rangle_{\partial \sigma_k}. \tag{3.5}$$

As long as the mesh has no boundary, the second sum vanishes since the normals N of each edge of the mesh cancel each other out as adjacent triangles have mirrored normals. Consequently, we are left with the term $\langle \nabla f, \nabla \phi_j \rangle$ for each triangle and as long as that term

does not vanish, the resulting Laplacian is not zero. Because f is a linear combination, we can further simplify:

$$\langle \nabla f, \nabla \phi_j \rangle = \sum_i f_i \underbrace{\langle \nabla \phi_i, \nabla \phi_j \rangle}_{C_{ij}} \tag{3.6}$$

$$= \frac{1}{2} \sum_{i \in neighbor(j)} (\cot \alpha_{ij} + \cot \beta_{ij})(f_i - f_j) \tag{3.7}$$

$$= (Cf)_j \tag{3.8}$$

where f_i is the value of f at the vertex v_i and α_{ij}, β_{ij} are the angles opposite of the edge between the vertices v_i and v_j on the adjacent faces. Since we are searching for the coefficients h_i of the Laplacian and h is a linear combination, we can reformulate:

$$\langle h, \phi_j \rangle = \sum_i h_i \underbrace{\int \phi_i(x)\phi_j(x)dx}_{M_{ij}} = (Mh)_j \tag{3.9}$$

If we now combine (3.8) and (3.9), the result is a matrix representing the Laplace operator:

$$\langle h, \phi_j \rangle = \langle \Delta f, \phi_j \rangle Mh = Cf \Rightarrow h = M^{-1}Cf = Lf \tag{3.10}$$

Here the matrix M is a mass matrix, containing area elements derived from the triangular faces of the mesh and C is called the stiffness matrix, which contains the cotangents. The resulting matrix L describes the discrete Laplacian and maps the coefficients of f to the coefficients of $h = \Delta f$ and has the dimension of the number of vertices to the power of two. For further information on this method, see [19].

Eigenfunctions of the Laplacian Because the Laplace-Beltrami operator is linear, it has to have eigenfunctions, which are used in the metrics later described. Since we already showed that the Laplacian can be represented by a matrix L to map functions to functions, the eigenvectors of L represent functions on the mesh. Using the Helmholtz equation and the knowledge obtained until now, we can use the matrices M and C to solve a generalized eigenvalue problem.

$$\Delta f = \lambda f \Rightarrow M^{-1}Cf = \lambda f \rightarrow Cf = \lambda Mf \tag{3.11}$$

As C and M are symmetric, we can use the QZ-algorithm as described in [17] to solve for the eigenvalues and eigenfunctions of the Laplacian. Note that the QZ-algorithm returns a orthonormal (with respect to the M-inner product) basis of eigenvectors which in general is not unique, since eigenvectors can be scaled and still belong to the same eigenvalue. Therefore the computed eigenfunctions can have switched signs due to the algorithm, but apart from that they are invariant to isometries as is the Laplacian.

3.3.2. The Diffusion Distance

The first metric to be discussed is the diffusion distance. Specifically we consider heat diffusion over a surface, which is fully described by the heat kernel and associated with

the Laplacian. But instead of using the heat kernel itself, we use the so called Heat Kernel Signature (HKS) which was first proposed in [23]. The heat kernel takes into account the heat transfer between all points at all time steps, whereas the HKS restricts the heat kernel to the temporal domain. By preserving certain properties of the heat kernel, the HKS is a stable and isometry invariant metric on a mesh.

Heat Operator and Heat Kernel We start by introducing the basic facts about heat diffusion and the heat kernel needed to define the Heat Kernel Signature. Let S be a compact regular surface possibly with a boundary. The heat diffusion process over S is dominated by the heat equation

$$\Delta_S u(x,t) = -\frac{\partial u(x,t)}{\partial t}$$

where Δ_S is the Laplacian of S and, if S has a boundary, u is required to satisfy the Dirichlet boundary condition $u(x,t) = 0 \forall x \in \partial S, t \in \mathbb{R}^+$. The heat operator $H_t(f)$ describes the heat distribution after time t given the initial heat distribution $f : S \to \mathbb{R}$, specifically fulfilling equation (3.3.2) at all T and converging to f for $t \to 0$. Both the heat operator and Δ_S are mapping real-valued functions defined on S to another and it can easily be verified that they are related by $H_t = e^{-\lambda \Delta_S}$. Hence both operators have the same eigenfunctions and if λ is an eigenvalue of Δ_S, then $e^{-\lambda t}$ is an eigenvalue of H_t corresponding to the same eigenfunction.

Definition 20 (heat kernel)**.** *The heat kernel is the minimum function $k_t(x,y) : \mathbb{R}^+ \times S \times S \to \mathbb{R}$ satisfying*

$$H_t f(x) = \int_S k_t(x,y) f(y) dy. \tag{3.12}$$

For compact S the heat kernel has the eigen-decomposition

$$k_t(x,y) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(x) \phi_i(y), \tag{3.13}$$

where λ_i and ϕ_i are the i^{th} eigenvalue and eigenfunction of the Laplace-Beltrami operator.

From a intuitive perspective, the heat kernel $k_t(x,y)$ is the amount of heat transfered from x to y in time t from the starting heat distribution of one heat unit at x. Another interpretation is that the heat kernel is the transition density function of Brownian motion on the regular surface. This means that for $x, y \in S$, the heat kernel describes the probability of a random walker arriving at y in time t after starting at x.

According to [23] the most interesting properties of the heat kernel are that it is symmetric, isometry invariant, multi-scale and stable. Its symmetry is obvious, since it does not make a difference for the heat transfer from which point the heat originates, the amount of heat transferred stays the same. That the heat kernel is isometry invariant is a consequence of the relation to the Laplacian which is invariant to isometries and only depends on intrinsic properties of the shape. Due to this property, the heat kernel can be used to analyze shapes undergoing isometric deformations, for example matching articulated shapes. The multi-scale property comes from the heat kernels dependency on the time parameter. It means that for small values of t, the function $k_t(x,\cdot)$ is mainly determined

by small neighborhoods of x and these neighborhoods grow bigger as t increases. This implies that $k_t(x, \cdot)$ reflects the local properties of the shape around x for small values of t, while it captures mainly the global structure of the shape from the point of view of x for large values of t. Lastly, the heat kernel being stable means that it is only slightly influenced by perturbations of the underlying regular surface. The majority of deformable shapes in practice are not isometric and therefore a reliable metric should not be sensitive to small perturbations. To explain the stability, we refer back to the heat kernels interpretation as the transition probability of the Brownian motion on the surface. Intuitively, this means that $k_t(x, y)$ is a weighted average of over all paths between x to y in time t, which should not be greatly affected by local perturbations.

Connection to Diffusion Distances The Heat Kernel Signature is closely related to diffusion maps and diffusion distances proposed by Lafon [13] for data representation and dimensionality reduction. The diffusion distance between $x, y \in S$ at time scale t is defined as

$$\begin{aligned} d_t^2(x, y) &= k_t(x, x) + k_t(y, y) - 2k_t(x, y) \\ &= \sum_{i=0}^{\infty} e^{-\lambda_i t}(\phi_i(x) - \phi_i(y))^2. \end{aligned} \tag{3.14}$$

This fact is pretty interesting, since we will see later on, that the Biharmonic distance as well as the Commute-time distance can be obtained by just replacing the heat kernel k_t by their respective "kernels". Notice, that the Heat Kernel Signature and its corresponding distance function is not scale invariant. If we use the truncated sum from (3.14), we get a good approximation of the distance between two points. Since the eigenvectors and eigenfunctions of the Laplace-Beltrami operator can be computed on a discretized mesh, we will use the truncated sum to approximate the distances between heat kernel signatures on a given mesh.

3.3.3. Commute-time Distance

A problem with The diffusion distance is that for one it is not scale invariant and second that it depends on a time parameter, which has to be set for each shape specifically. To solve the second problem, we can integrate the heat kernel over time, so that the result is no longer dependent on the time parameter:

$$\begin{aligned} \int_0^{\infty} k_t(x, y)dt &= \int_0^{\infty} \sum_i e^{\lambda_i t}\phi(x)\phi(y)dt \\ &= \sum_i \phi(x)\phi(y) \int_0^{\infty} e^{\lambda_i t}dt \\ &= \sum_i \frac{1}{-\lambda_i}\phi_i(x)\phi(y) = g(x, y) \end{aligned} \tag{3.15}$$

The result is the commute-time kernel, which corresponds to the probability density function of a random walk of any length transitioning from point x to y. It can be used to define

the commute-time distance in a similar fashion as the diffusion distance:

$$d_C^2(x,y) = g_C(x,x) + g_C(y,y) - 2g_C(x,y) \tag{3.16}$$
$$= \sum_{i=0}^{\infty} \frac{1}{-\lambda_i}(\phi(x) - \phi(y))^2. \tag{3.17}$$

Furthermore, the commute-time distance is scale-invariant. Note that the commute-time kernel is the harmonic Green's function of the Laplacian, which means that it fulfills the equation $\Delta \int g_C(x,y)f(y)dy = f(x)$ for a given f. The disadvantage of this is that the harmonic Green's function has a singularity at the diagonal, which means that (3.16) is not defined in the continuous case.

3.3.4. The Biharmonic Distance

The biharmonic distance was proposed by Lipman, Rustamov and Funkhouser after reviewing the above described metrics on shapes in their paper [14]. Their new distance operator is quite similar to the diffusion distance and the commute-time distance, but they exchanged the diffusion kernel with the Green's function of the biharmonic differential equation. In the continuous case, the biharmonic distance can be defined using the eigenvectors and eigenvalues of the Laplace-Beltrami operator

$$d_B(x,y)^2 = \sum_{i=0}^{\infty} \frac{(\phi_i(x) - \phi_i(y))^2}{\lambda_i^2}. \tag{3.18}$$

This definition is only slightly different from the other two distances, the only part changing is the factor in font of the difference between the two eigenvectors. While the biharmonic distance has a factor of $1/\lambda_i^2$, the power of the λ in the definition of the commute-time distance is one and the factor of the diffusion distance is $e^{-\lambda_i t}$. So even though it is a seemingly minor change, the behaviour and the properties of the distance functions differ largely. The basic idea behind this is that it is related to how fast the normalized λ_i in the factors decay: if the decay is to slow it will produce a logarithmic singularity at the diagonal of the harmonic Green's function (like $\sum_i 1/\lambda_i \ \sum_i 1/i$). On the other hand, if the decay is to fast, the eigenvectors with high frequencies are basically ignored and the distance becomes "too global". The paper states, that the factor of $1/\lambda_i^2$ provides a good balance as it decays fast enough to be shape-aware over the whole mesh and slow enough to get good local properties around the source point.

The biharmonic distance as defined in (3.18) can further be written to fit the diffusion distance formulation with a different kernel:

$$\begin{aligned} d_B^2(x,y) &= \sum_{i=0}^{\infty} \frac{|\phi_i(x)|^2}{\lambda_i^2} + \sum_{i=0}^{\infty} \frac{|\phi_i(x)|^2}{\lambda_i^2} - 2\sum_{i=0}^{\infty} \frac{\phi_i(x)\phi_i(y)}{\lambda_i^2} \\ &= g_B(x,x) + g_B(y,y) - 2g_B(x,y). \end{aligned} \tag{3.19}$$

using the Green's function $g_B(x,y)$ of the biharmonic operator Δ^2, which means that it satisfies the relation

$$\Delta_{(x)}^2 \int g_B(x,y)f(y)dy = f(x) \tag{3.20}$$

for smooth enough f. This information can used to discretize the biharmonic distance by obtaining the discretized g_B from (3.20) and applying it to (3.19) to obtain the biharmonic distance on the mesh. Another way to practically compute it is to approximate the distance by using the truncated sum, as it is the standard methodology for approximating the diffusion distance. We compute the first N eigenfunctions of the discrete Laplacian and computing $d_B(x, y)$ as follows:

$$\tilde{d}_B(x, y)^2 = \sum_{i=0}^{N} \frac{(\phi_i(x) - \phi_i(y))^2}{\lambda_i^2}. \tag{3.21}$$

Although the error bound is (in general) linear in $1/K$ this approximation provides considerable speedup in trade-off of accuracy over the exact computation. Additionally, since the approximate distance uses a fixed set of eigenvectors, it is smooth.

Part II.

Implementation and Experiments

4. Testing Purposes and Pipeline

In order to get a grasp of the properties and the behavior of the different distances, we ran a set of experiments on a set of meshes from the datasets TOSCA [4], SHREC 2010 [2] and SHREC 2011 [9]. These datasets contain triangulated meshes, undergoing different transformations, ranging from scaling through holes in the mesh to noise and topology changes. These shapes will allow us to get a understanding of the performance of the four different metrics if subjected to these kinds of transformations.

4.1. Timing

The first experiment we ran was to compare how long it took the different metrics to compute the distances between points, a task not uncommon in shape analysis applications. In order to do so, we used Matlab on a 2.40GHz Intel Core 2 Duo processor. We timed the computation of distances on meshes with varying amounts of vertices from one vertex to all other vertices. Additionally, in this experiment we take a deeper look at the computations which can be done beforehand. The computation of the distances based on the Laplacians eigenfunctions/-vectors can be sped up by precalculating them and then using the saved eigenfunctions and eigenvectors for the remaining computation steps, making the overall computation-time shorter. Notice however that there cannot be any time saving steps taken for the geodesic distance, at least not with the method described in chapter 3.2. As a side note, an alternative approach, which approximates the geodesic distance by using the heat diffusion process based on the Laplacian, was proposed in [7]. It takes advantage of the fact, that heat distributes over a mesh based on the distance between the origin of the heat and the point of question.

4.2. Sensitivity to noise, tessellation and deformation

To test the robustness of the metric functions, we compute the distance function for different kinds of changes of the original surface. We then compare the result visually by correlating the color and the shape of the isolines of the original mesh and the changed one. As the SHREC datasets provide different strengths of modifications in a multitude of areas, we will depict those showing the clearest result to make a point in the resulting figure. To be specific, the different deformations we will take into account are:

- isometry: The surface undergoes a isometric transformation, for example if the mesh of a person is changed to have a different pose.
- changes in local/global scale: These changes include the shrinking and growing in size of parts of the mesh or the whole mesh.

- micro holes/holes: Not watertight meshes with holes of varying sizes.
- shot noise/noise: Here, different amounts of noise is added to the vertex coordinates of parts of the mesh or the whole mesh.
- topology: Summarizes changes which change the topology of a mesh like a different tessellation of the surface or additional edges or faces.

Farthest point sampling In the next experiment we watched the behavior of the different metrics, if they are used to obtain a farthest point sampling, FPS for short, of a given mesh. The idea behind the FPS is that meshes can get quite complex with huge amounts of vertices, slowing down the computations on the mesh. The FPS provides a fast and practical way to obtain an almost optimal approximation of an open ball covering if it is given a metric on the mesh. The way it works is to start from a random vertex or with the two vertices farthest away from each other and then, with each iteration step, add the vertex to the set, which is farthest away from the currently selected. The resulting set in general is not unique but can then be used to assign all vertices to vertices of the set, representing them and resulting in a so called Voronoi sampling. The obtained FPS and the Voronoi sampling can then be used to save computation time on tasks like shape matching, so the usage of the FPS is a common procedure in shape analysis. During the experiments, we compared the FPS of 150 points of the original shape with the FPS of shapes with the above mentioned changes and how good the set covers the mesh. In this experiment one additional metric was used: The Euclidean distance is not isometry invariant, which makes it not fitting to use for shape matching in general, but it is known to produce good results if used in the FPS. And to keep the computation times within a reasonable range, we approximated the exact geodesic distance by the Dijkstra algorithm on the larger meshes.

Error measurement As a last experiment, we wanted to get a quality measure of the different metrics. We did so by comparing the relative errors which resulted from the deformations mentioned above. The ideal result for a metric was, if it did not change under isometric deformations and only had slight changes under the other deformations. To test the completion of this task, we first computed the distances from one point to all other points on the mesh. After selecting one mesh as the reference mesh, also called the null shape, we subtracted the distances of corresponding points. To make the resulting error more informative, we divided it by the maximum distance on the reference mesh and only considered its mean and the maximum error.

Example 4.1. *Let p_0 be the origin point of the distances on the null shape M and $p_1, \cdots, p_4$ other points on the same mesh. Further, let $d(p)$ be the distance from p_0 to p on the original shape and $p'_0, \cdots, p'_3$ are the corresponding points on the deformed mesh M' with the distance function d'. An important thing to notice is that not every point on M has to have a corresponding point on M', like p_4 has no correspondence in this example. To calculate the errors, we repeat for all corresponding points $p_i \in M, p'_i \in M'$:*

$$e_i = \frac{d(p_i) - d'(p'_i)}{\max_{p \in M} d(p)}$$

To finish the error computation, we compute the mean of the errors in all the points and the save it together with the maximum error.

5. Implementation Details

This chapter concentrates on the specific implementation choices during the testing process. The main software used to implement and run the computations is Matlab R2014a under GNU/Linux 3.14. In addition to that, we used a collection of functions for graphs and meshes called "Toolbox Graph" from the official Matlab file exchange[18] and an implementation of the geodesic distance as described in 3.2 taken from Google Code[1]. The whole project has been made available online to check the actual source code under [21].

5.1. The metrics

The Geodesic Distance The implementation of the geodesic distance is written for Windows in C++, to be compiled into a shared library and then called with a Matlab API. This resulted in a small amount of problems, most of which were small differences due to the change of the underlying operating system. Apart from that, the general idea behind the computation is to first input the mesh through the API and to specify the Algorithm type, which has to be either the exact geodesic distance, the Dijkstra algorithm on the original shape or Dijkstra performed on the mesh with a certain number of subdivision of the edges. It is obvious, that the exact algorithm is the slowest one, while Dijkstra is the fastest of these three algorithms, trading accuracy with speed.

We wrote our own wrapper function around the Matlab API to increase the comfort of using them, by only having to input the mesh information, consisting of the data of the vertex positions and the faces together with the origin point of the geodesic distance to be computed. Further we chose to return the distance from the origin point to all other vertices, since after the propagation of the distances over the whole mesh, these can be retrieved easily. As stated before, no computations could be carried out beforehand and be reused later to speed up the computation. Also, each origin point has to be handled separately, since the computed data does not coincide between different source points, leaving only the initiation of the mesh and the algorithm to be performed once of all following computations.

One peculiarity of the chosen implementation is that it checks the input meshes, whether or not they are sane in different aspects. This led to problems with a few of the deformed meshes, leading to the program to crash since the input mesh did not fulfill the requirements for continuation. In particular, the algorithm used by the SHREC datasets to decrease the amount of vertices and create the *sampled* meshes of the dataset generated meshes not fulfilling this requirement. While not all of those meshes resulted in a crash, the whole set of meshes modeled to have holes and some of the meshes with micro holes in the original mesh also were rejected by the mesh check of the geodesic distance implementation. To avoid this problem, we removed those meshes from our testing set.

The Laplacian eigenfunctions and eigenvalues To compute the normalized Laplacian matrix L, we used the function `compute_mesh_laplacian()` from the Toolbox Graph. The eigenfunctions ϕ_i and eigenvalues λ_i were then obtained by using Matlabs internal function `eigs()` to compute the first n eigenvalues. The algorithm returned the positive eigenvalues, sorted in an ascending order, and their orthogonal eigenvectors. An important note is that the first eigenvalue was either zero or at least very close to zero and should therefore be discarded to grant the numerical stability of the resulting distance functions. We chose to use the first 200 eigenfunctions to get a reasonably good approximation of the distance functions.

The distances based on the Laplacian The distance functions based on the Laplacian eigenfunction are approximated by using the general formula:

$$d(x,y) = \sum_{i=1}^{200} (\phi_i(x) - \phi_i(y))^2 \cdot f \tag{5.1}$$

were the factor f is a function that changes depending on which distance function we are using. To compute the diffusion distance we used $f(\lambda_i, t) = e^{-2t\lambda_i}$, for the commute-time distance $f(\lambda_i) = \frac{1}{\lambda_i}$ was used and for the biharmonic distance, the factor was set to be $f(\lambda_i) = \frac{1}{\lambda_i^2}$. Since the input and the computations were almost the same for the different functions, we decided to combine them into one function with a parameter to choose the wanted one through a function handle. The most efficient way we found to implement this is to compute this sum for one x and all other y one by one within one function to minimize the overhead and maintain the described flexibility.

One special adjustment was made on the computation of the diffusion distance. Since it is known that it is not scale invariant, we used method described in [14]: By rescaling the time parameter to $t \leftarrow t/\lambda_1$ where λ_1 is the smallest eigenvalue not equal to zero, we can use the resulting t to select an equal scaling of the diffusion distance over different meshes.

5.2. Testing

Timing of the computation We chose different meshes with vertex counts of approximately $1k, 5k, 10k, 20k$ and $50k$ to measure the computation time of the different metrics. To do that, we used the `tic()` and `toc()` functions of Matlab to time the pure computation times, starting from the function call until the result is returned. The experiment computed the distance from one vertex to all other vertices and averages the computation time over ten independent repetitions, writing the results to a text file.

Plotting of the distance functions To plot the metrics, we plotted an approximation of equidistant isolines on the mesh. The isolines divide the surface into level sets, regions with similar values and therefore visualize the change of the metric over a mesh. In order to compute the shape of the isolines, an algorithm first searches for border vertices between two of a given set of equally spaced level sets. Then it constructs line segments between two neighboring border vertices, resulting in a set of line segments, which are combined to obtain isolines. Since the isolines are bound to vertices, it only approximates the real

isolines, which in general do not need to cross a vertex. These isolines are then plotted onto the mesh and exported as a TIFF-file for the following visual processing.

In order to speed up the computation, we precomputed the Laplacian eigenfunctions and -values. This results in computation times within a few minutes instead of an hour. Additionally, we use correspondence files so that we do not need to find corresponding vertices by hand, but can just look them up.

Error measurement In order to measure the maximum error and the mean error, we use the same tricks to speed up the computation as in the previous paragraph. Using precomputed eigenfunctions and correspondence files, we basically proceed just as stated in the theoretical chapter of this paper. After computing all distances from one point to all the other points on all meshes (with their corresponding points), we compute the relative error by subtracting the distance of the base mesh from the distance on the current shape and divide the result by the maximum distance on the base mesh. Then the mean and the maximum error are computed and saved for all metrics. This process is repeated over meshes with different deformations and different starting points to get an idea of the metrics properties.

Farthest point sampling The last experiment we implemented is the farthest point sampling based on the different metrics. We used function handles to switch between the different distance functions and built a generic implementation of the farthest point sampling around that. As the basic idea behind the farthest point sampling is to start with one point and then add the points to the set, which are farthest away from the set, we basically do the following: We save the minimal distances from all points to the points of the set in a vector d. After a point p_i was added to the set, we update d by computing the distance from p_i to all other vertices and then taking the minimum between the resulting vector and d. Then we choose the next point p_{i+1} to be added as the point corresponding to the minimal distance in d.

Additionally, we are using the standard euclidean distance in this experiment. To obtain the distance from one point to the other points, we simply use the standard formula for the euclidean distance as:

$$d^2_{euclidean}(v,u) = (v_1 - u_1)^2 + (v_2 - u_2)^2 + (v_3 - u_3)^2,$$

where $u, v \in \mathbb{R}^3$ are two vertices of the mesh.

Part III.

Results and Conclusion

6. Testing Results

6.1. Timing

Table 6.1.: The table shows the time in seconds to compute the different distances. Take note, that all metrics except the geodesic distance need to have the Laplacian eigenfunctions computed.

$\|Vertices\|$	Laplacian eigenfunctions /-values	diffusion	commute time	biharmonic	geodesic exact	geodesic Dijkstra
1k	2.98	0.033	0.016	0.017	0.057	0.006
5k	12.46	0.122	0.064	0.070	0.481	0.014
10k	30.35	0.283	0.157	0.169	1.449	0.036
20k	65.08	0.487	0.268	0.288	3.632	0.063
50k	206.58	1.193	0.655	0.705	12.123	0.149

The average computation times to compute the distance from one point to all other points are condensed into table 6.1. We can see that the computation of the Laplacian eigenfunctions and eigenvalues takes the most time. But after obtaining them, the computation time of the diffusion, the commute-time and the biharmonic distance are really short. Since they all have the same basic structure, the small differences in speed are results of the different factors: While the factor $\frac{1}{\lambda_i}$ of the commute-time distance is the fastest to compute, the additional square of the eigenvalue for the biharmonic distance makes it slightly slower. The fact that the factor of the diffusion distance consists of an exponentiation and a product makes it the slowest one of those three.

One important thing to note here is that even though the computation of the Laplacian eigenfunctions and eigenvalues takes quite a bit of time, the result can be saved and reused every time the distance has to be computed. So if we were to compute the distance between all points of the mesh (even with respecting the symmetry of the metric and therefore half the computations), the computation of the Laplacian based metrics is already faster then the exact geodesic distance on the mesh with 5000 vertices by an approximate factor of 4.

Even though the implementation of the geodesic is highly optimised, the exact geodesic distance is definitely the slowest metric to compute. Here the propagation of the windows over all edges of the mesh comes into play. In the paper [24] it is stated, that the windows per edge increase exponentially and, since each new vertex is connected to the rest of the mesh by at least one edge, this means that the computation time has to increase at least at the same speed. If you approximate the geodesic distance by the Dijkstra algorithm, the computation gets faster than every other metric but its quality decreases significantly.

6.2. Sensitivity to noise, tessellation and deformation

6.2.1. Isolines

Our first set of experiment results provides a visualisation of the different metrics on shapes from the SHREC 2010 dataset. Similar to the experiments in [14], we computed the distance from a single source vertex to all other vertices on the mesh, color coding it onto the mesh by using Phong shading, which results in a dark blue color for small distances and bright red colors for bigger values. In addition to that we plotted white, equally spaced isolines onto the mesh to give a better visualisation of the properties of the chosen metric.

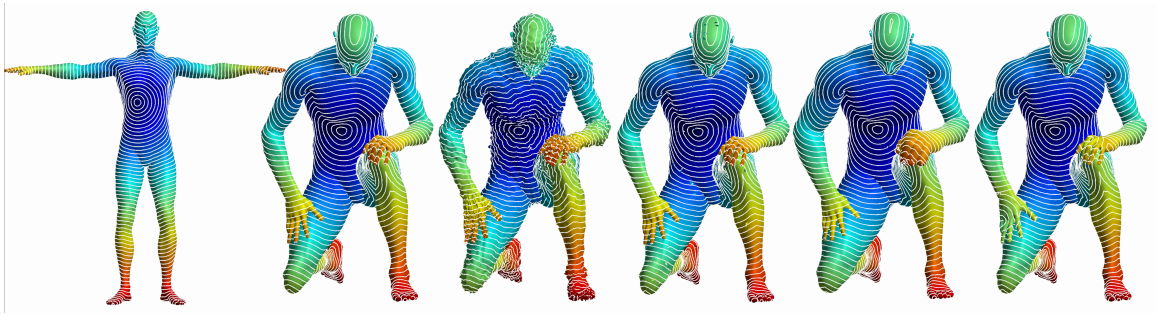


Figure 6.1.: Comparison of the geodesic distance under different deformations of the mesh; from left to right: the null shape, isometry, noise, microholes, local scaling and topology changes.

The geodesic distance It is commonly known that the local properties of the geodesic distance are desirable. As we can see on all meshes in figure 6.1, the geodesic distance is isotropic and increases in a circular fashion, extending over the whole mesh. There also lies its biggest flaw: The geodesic distance is not globally shape-aware, which can be seen on the meshes as that the isolines run diagonal along the arms and legs. This also means, that the shape of the isolines far from the source vertex depends on the exact placement of the starting point. For example, would the source point lie somewhere on the left arm of the mesh, then the isolines would run perpendicular to the direction of the arm, forming small circles, instead of being distorted as they are in figure 6.1. Another point is that the geodesic distance is not smooth, as can be best seen on the locally scaled shapes left knee, where there is a big cusp. In general, the geodesic distance results in cusps and ridges on the mesh, especially at the opposite side of the source vertex. Furthermore, it is susceptible to topological changes or bigger holes in the mesh (which could not be tested): On the corresponding mesh, the right hand is connected to the right thigh through a face, resulting in a significant change of the isolines (and therefore also the metric), since the shortest path through the upper leg is shorter than the geodesic distance before the change. The isometric transformation from the null shape resulted in changed isolines on the legs, which again are a result of the missing global-shape awareness of the geodesic distance. Apart from that, the geodesic distance is largely invariant to noise, small holes and local scalings.

The diffusion distance To begin with, the main difficulty with the diffusion distance is that it is not parameter-free. It is possible to adjust the amount of global and local shape-awareness by choosing bigger or smaller t, but it is not possible to have both at the same time. We start by looking at the results of the diffusion distance with $t = 1$. The global

Figure 6.2.: Comparison of the diffusion distance with $t = 1$ under different deformations of the mesh; from left to right: the null shape, isometry, noise, holes, local scaling and topology changes.

shape-awareness of the diffusion distance is really dominant, resulting in isolines perpendicular to the central axis of protrusions as the arms or legs. What can be seen close to the source point is that the global shape has an unwanted influence on the local distances, resulting in elliptic isolines close to the source vertex instead of circular ones. This shows also, as with increasing t the isolines close to the source vertex get closer and closer to being parallel instead of being circular. As we already presumed the time parameter in this case favors the global properties, resulting in a not isotropic propagation close to the source. Apart from that, the diffusion distance seem so almost invariant to the shown deformations of the shape.

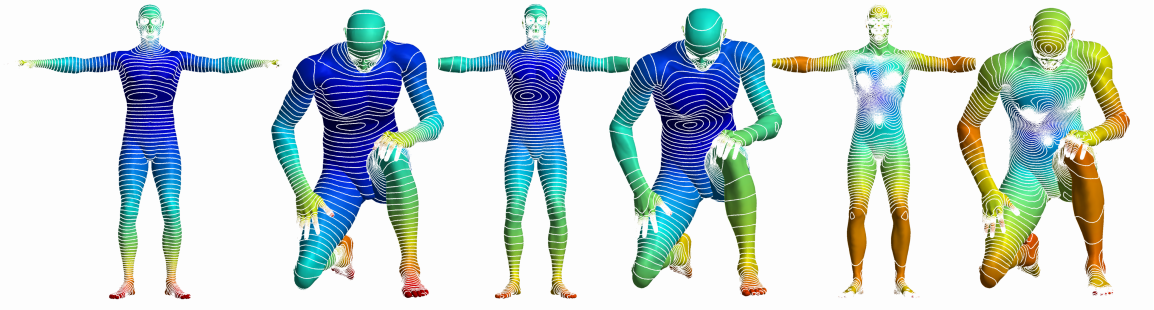


Figure 6.3.: Comparison of the diffusion distance with $t = 0.1, 0.05, 0.01$ on the null shape and an isometric deformation; the value of t decreases from left to right.

In the second figure 6.3 we decided to show the behaviour of the diffusion distance with smaller t. Even though the parameter with $t = 0.1$ is set to a relatively small value, it still has some good global properties, such as the isolines perpendicular to the central axis of protrusions as the arms or legs. But on the other hand, the isolines close to the source vertex return to a more circular appearance. This effect get more and more prominent,

the smaller t gets, revealing more and more detail close to the source vertex and loosing its global properties up to a degree, that the vertices far away from the source vertex are assigned near constant values, resulting in level sets with large areas.

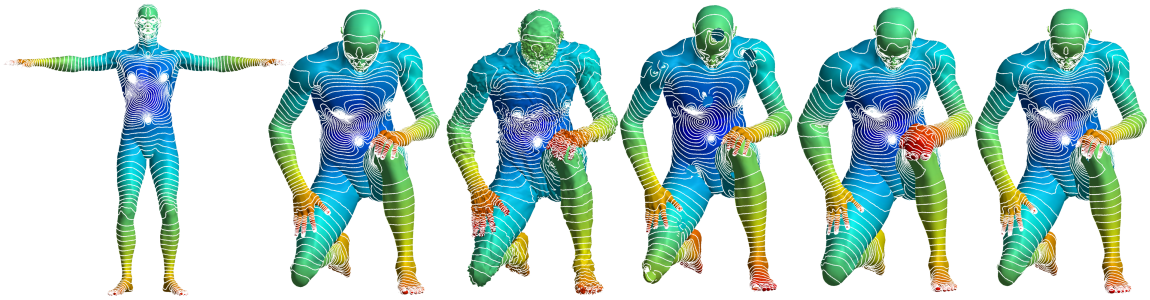


Figure 6.4.: Comparison of the commute-time distance under different deformations of the mesh; from left to right: the null shape, isometry, noise, holes, local scaling and topology changes.

The commute-time distance The commute-time distance is, as mentioned in the subsection 3.3.3, the integral of the diffusion distance over all t and therefore being multi-scale without depending on a parameter. As we can see in figure 6.4, it obtained some of the diffusion distances properties: The global shape-awareness has been preserved, as the isolines on the limbs are still vertical to their central axis, while the isolines close to the source point are close to circular. But some of the influences of the diffusion distances result in unwanted behaviour, such as the local maxima at the belly button and around the belly. These extrema in particular can be seen in figure 6.3 with really small t. In addition to that the commute-time distance is not smooth, which can be seen by the isolines having crooks and cusps. Concerning the different deformations, the commute-time distance seems to be only affected lightly by them, resulting in the isolines wandering a small distance up or down the limbs, but generally retaining their shape. The only one which has large effect is the appearance of holes in the mesh. For one, there are multiple local maxima at the edge of a hole, which is supplemented by the fact that we had to manually change the distance of the 60 (out of around 50k) vertices, which had the largest distance from the source point, to zero. This minority of points had distance values up to double the maximal distance plotted in figure 6.4. So even though the isolines on the mesh with holes still resemble the others, the behaviour of the commute-time distance in this experiment is not desirable.

The biharmonic distance The last set of results, which shows the performance of the biharmonic distance, can be seen in excerpts in figure 6.5. As it was designed to be, it is shape-aware and at the same time close to the geodesic distance in proximity of the source point. This can be seen in the circular shape of the isolines close to the source, while the isolines follow the shape more and more, the farther they are from the source point. Under the different deformations, the isolines and therefore the behaviour of the biharmonic metric is preserved in general. Only on the arms undergoing local scaling

Figure 6.5.: Comparison of the biharmonic distance under different deformations of the mesh; from left to right: the null shape, isometry, noise, holes, local scaling and topology changes.

and topology changes the distance changes slightly, increasing the distance to one hand while decreasing the distance to the other. But altogether, the biharmonic distance keeps its properties across all deformations.

6.2.2. Farthest point sampling

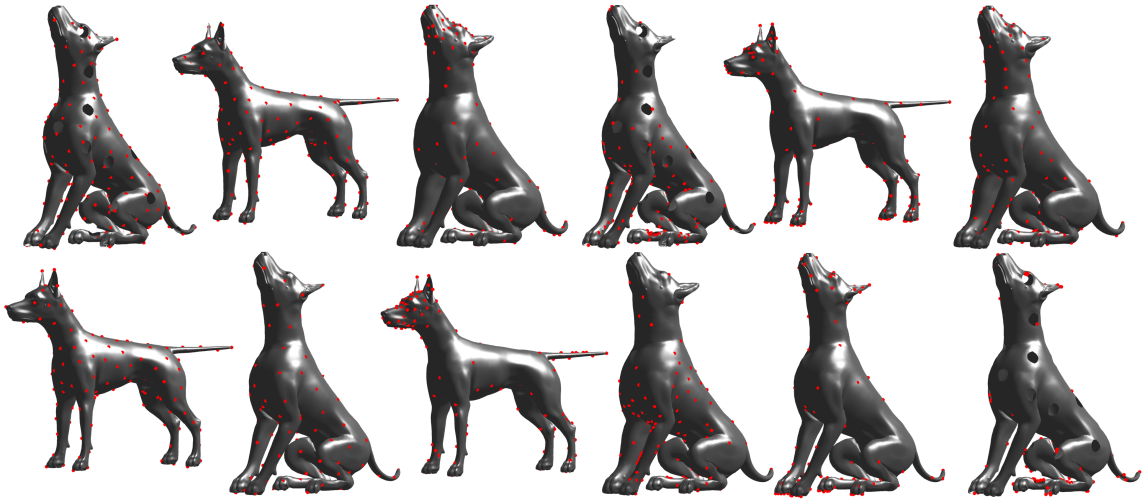


Figure 6.6.: A selection of farthest point samplings of the different intrinsic metrics on shapes with holes, topology changes, local scale changes and the null shape; from left to right: Euclidean, geodesic, diffusion with $t = 0.1$, diffusion with $t = 1$, commute-time and biharmonic distance.

The results of this experiment are quite straight forward: Using the Euclidean distance results in the best, equally distributed set of vertices over the mesh, closely followed by the geodesic distance as can be seen in figure 6.6. The diffusion distance has an interesting effect, that depending on the parameter t the points either cluster around the head of the dog or around the torso, leaving only a small number of vertices on the remaining body parts.

As has already happened during the plotting experiments, the commute-time distance intensifies the effects of small t, resulting in almost no vertices on the torso and clusters of points on the head and the legs of the dog. While the FPS based on the biharmonic distance seems to be almost equally distributed on the upper part of the figure, the vertices of the lower part, showing the mesh with a number of holes, are concentrated on the legs and the head while covering almost nothing of the rest of the mesh. This is probably for one a result of the difficulties of the metric to handle a mesh with holes and it can also be seen on the upper picture, that the biharmonic distance has a tendency to distribute the points with a bigger focus on the head and legs then on the torso. Therefore the best results are being produced by the fastest and simplest metric of the set: the Euclidean distance. Even though it is unusable in any of the other use cases of intrinsic metrics, like for example shape matching because of its missing invariance isometries, for the task of computing a farthest point sampling the Euclidean distance is the best.

6.2.3. Error measurement

We now review the results of our error computation with the different metrics. As a reminder, the calculated values are the mean and maximal error between a mesh under a certain deformation and the null shape, divided by the maximum distance on the null mesh. As can be seen in table 6.2 there is no clear answer as to which metric is superior in regards

Table 6.2.: The mean error of the distances, grouped by the type of deformation. The minimum value is underlined.

metric	isometry	local scale	scale	topology	noise	shot noise	micro holes	holes
geodesic	.0232	.0407	.1580	.0508	.0218	.0419	.0417	-
diffusion t=0.1	.0043	.0310	.0058	.0408	.0241	.0044	.0059	.0434
diffusion t=1	.0022	.0320	.0029	.0223	.0180	.0018	.0031	.0528
commute-time	.0034	.0092	.0037	.0138	.0309	.0031	.0035	.0331
biharmonic	.0049	.0220	.0016	.0420	.0625	.0024	.0017	.0388

to the metric distortion of the distance from one source point to all other points. While it is obvious, that the geodesic and the diffusion distance with a small t are no competition, the other metrics compete for the best mean errors. The diffusion distance with $t = 1$ is clearly the best for the categories isometry and noise, whereas the commute-time has the best mean errors for topology changes, changes in local scale and meshes with holes. This can be tied to the alternate definition as the average time a random walker takes to go from one point to the other and return, which is not as strongly influenced by those deformations as the other metrics. Finally, the biharmonic distance provides the smallest mean errors for scaled meshes and micro holes. The bad performance of the geodesic and the more local version of the diffusion distance probably based on their bad shape-awareness. Additionally, one can see that the geodesic distance is not scale-invariant, as the corresponding mean error is disproportionally high.

In our experiments measuring the maximum (relative) error, which can be seen in table 6.3, the biharmonic distance had the best performance. In five out of eight categories

Table 6.3.: The maximum error of the distances, grouped by the type of deformation. The minimum value is underlined.

metric	isometry	local scale	scale	topology	noise	shot noise	micro holes	holes
geodesic	.0984	.1554	.4064	.2227	.1311	.1567	.1566	-
diffusion t=0.1	.0258	.2745	.0347	<u>.0774</u>	.0728	.0365	.0346	<u>.2209</u>
diffusion t=1	.0217	.1473	.0301	.2042	<u>.0685</u>	.0330	.0302	.4938
commute-time	.0288	.0899	.0493	.1598	.1381	.0487	.0469	.6374
biharmonic	<u>.0188</u>	<u>.0648</u>	<u>.0285</u>	.3226	.1113	<u>.0275</u>	<u>.0276</u>	.5229

the biharmonic metric resulted in the smallest maximum error. Only on meshes under topology changes, noise and with holes the diffusion distance was superior to the biharmonic distance. Apart from that, note that the commute-time distance again showed inconsistent behaviour on meshes with holes, as it resulted in the smallest mean error but, similar to the isolines experiment, the biggest maximum error.

Altogether, the deformations can be separated into two classes, based on the performance of the metrics in regards to the maximum error and independent of the specific metric: The deformations resulting in relatively small maximum errors approximately ranging from two to 8 percent, namely isometry, local scale, scale, micro holes and shot noise and the more "challenging" deformations with maximum errors ranging from ten to 40 percent, namely noise, holes and topology. The exact same partitioning of deformations can be extracted from table 6.2 by separating the deformations by the question, whether or not the majority of the metrics have a mean error below or above one percent, which means we can safely conclude that noise, holes and topology changes are the most challenging deformations of a mesh for all considered intrinsic metrics.

7. Conclusions and Future Work

This work compares the most commonly known intrinsic metrics, the geodesic distance, the diffusion distance, the commute-time distance and the biharmonic distance based on their local and global properties as well as on their performance and their invariance to different common deformations of three-dimensional surfaces. We found, that the geodesic distance has good local properties and results in a decent FPS, but becomes almost unusable on a global scale, having generally the biggest mean and maximum errors and being not shape-aware.

In comparison, the diffusion distance can have either good local or good global properties, based on the time parameter t. Its problem lies in the fact that the diffusion distance can not be good at both at the same time, but can only have good global shape-awareness for large t or a solid local behavior using small t. If the diffusion distance is used for a farthest point sampling, then the points either cluster around the source point for small t or have an disproportionally small amount of vertices near the source point for larger t, not completing the basic idea behind the farthest point sampling. Concerning the error calculations, the diffusion distance gives a solid performance for noisy surfaces and performs moderately for the other deformations.

The commute-time distance is the integral over the diffusion distances over all t, resulting in decent local and global properties, coming at the cost of having local maxima. During our experiments, the commute-time distances had a small edge in the categories local scale and topology, where it had the smallest mean error and also had the second smallest maximum error. On the other hand, the commute-time distance had difficulties with handling meshes with holes, resulting in additional local maxima and therefore big maximum errors while having a small mean error. Apart from that, the commute-time metric had no special traits, clearly differentiating it from the rest.

Finally, the newest of the intrinsic metrics, the biharmonic distance, excelled in most of our experiments. Since it is tuned to be smooth and have equally good local and global properties, being shape-aware far from the source while being isotropic close to it, the biharmonic distance performed best in the visual comparison. And even though it is not appropriate to use for the farthest point sampling, it performed best in the metric distortion tests out of all the tested metrics.

As the usage of intrinsic metrics is one of the fundamental approaches in shape analysis, there is a wide range of future work possible. Possibly the first one would be to study the influence of different intrinsic metrics on different applications like the matching and segmentation of three-dimensional surfaces. Another point would be to include newly developed distances like the earth mover's distance, which was only presented this year in [22], into this comparison.

Appendix

A. Detailed Results

A.1. Timing of the computations

Here we give the exact output of the experiments for the timing and the error measurement. To start of, the following output gives the time in seconds it took to compute the distance from one vertex to all other vertices with the respective metrics on the different meshes.

raw data

```
----------------------30-Aug-2014----------------------------
        times: laplacian geodesic_dijkstra diffusion commute_time biharmoinc

shrec2010_0003.sampling.5:  964 verts
        times: 3.344817 0.005798 0.038892 0.030821 0.032886

shrec2010_0003.sampling.4:  4573 verts
        times: 12.515630 0.014021 0.166007 0.132712 0.137609

shrec2010_0002.sampling.3:  10762 verts
        times: 30.154200 0.035964 0.374983 0.290331 0.303356

shrec2010_0002.sampling.1:  20237 verts
        times: 64.855514 0.063136 0.683523 0.530818 0.551047

shrec2011_0001.null.0:  52565 verts
        times: 206.082749 0.149082 2.292431 1.739161 1.777857


----------------------31-Aug-2014----------------------------
        times: 0 geodesic_exact 0 0 0

shrec2010_0003.sampling.5:  964 verts
        times: 0.000003 0.064268 0.000000 0.000000 0.000000

shrec2010_0003.sampling.4:  4573 verts
        times: 0.000002 0.482627 0.000000 0.000000 0.000000

shrec2010_0002.sampling.3:  10762 verts
        times: 0.000002 1.465620 0.000000 0.000000 0.000000

shrec2010_0002.sampling.1:  20237 verts
        times: 0.000003 3.653875 0.000000 0.000000 0.000000

shrec2011_0001.null.0:  52565 verts
        times: 0.000002 12.020188 0.000000 0.000000 0.000000
```

```
----------------------02-Sep-2014----------------------------
        times: laplacian geodesic_exact diffusion commute_time biharmoinc

shrec2010_0003.sampling.5:  964 verts
        times: 2.979829 0.057016 0.032513 0.015851 0.017184

shrec2010_0003.sampling.4:  4573 verts
        times: 12.459126 0.480894 0.121599 0.064065 0.069511

shrec2010_0002.sampling.3:  10762 verts
        times: 30.346754 1.449161 0.283143 0.157000 0.169491

shrec2010_0002.sampling.1:  20237 verts
        times: 65.081098 3.632062 0.486950 0.267707 0.287883

shrec2011_0001.null.0:  52565 verts
        times: 206.580039 12.122674 1.192548 0.654706 0.705106
```

A.2. Error computations

The following is the output of the error computation. To obtain the tables 6.2 and 6.3, first the average of the absolute values of the different points on one mesh was computed and then the results of meshes from the same deformation type were averaged again. The values are relative errors, meaning that it is given in relation to the calculated maximum distance on the original shape.

raw data

```
----------------------06-Sep-2014----------------------------
1 geodesic, 2 diffusion t=0.1, 3 diffusion t=1, 4 commute-time, 5 biharmonic
        max error, avg error for distances from point 910

shrec2010_0001.isometry.1:
        1: mean 0.052910, max error 0.155468
        2: mean -0.007574, max error 0.053344
        3: mean -0.004927, max error 0.044472
        4: mean -0.005253, max error 0.065969
        5: mean -0.001054, max error 0.037862

shrec2010_0001.isometry.2:
        1: mean 0.004197, max error 0.048823
        2: mean 0.002476, max error 0.017797
        3: mean 0.001947, max error 0.015873
        4: mean -0.004235, max error 0.012502
        5: mean -0.007491, max error 0.012036

shrec2010_0001.microholes.1:
        1: mean 0.052837, max error 0.155468
        2: mean -0.007467, max error 0.053428
        3: mean -0.004751, max error 0.044403
```

```
        4: mean -0.004871, max error 0.067332
        5: mean -0.002006, max error 0.037556

shrec2010_0001.microholes.2:
        1: mean 0.052800, max error 0.155389
        2: mean -0.008136, max error 0.052877
        3: mean -0.005573, max error 0.044370
        4: mean -0.006292, max error 0.067473
        5: mean -0.002547, max error 0.037528

shrec2010_0001.localscale.1:
        1: mean 0.053049, max error 0.155473
        2: mean 0.023298, max error 0.273737
        3: mean 0.023835, max error 0.098575
        4: mean -0.007644, max error 0.106460
        5: mean -0.020654, max error 0.043352

shrec2010_0001.localscale.2:
        1: mean 0.050533, max error 0.153274
        2: mean 0.032746, max error 0.288551
        3: mean 0.043470, max error 0.209265
        4: mean -0.014668, max error 0.092477
        5: mean -0.029063, max error 0.096429

shrec2010_0001.noise.1:
        1: mean 0.044233, max error 0.148110
        2: mean -0.021256, max error 0.071856
        3: mean -0.033226, max error 0.029755
        4: mean 0.026016, max error 0.154957
        5: mean 0.062665, max error 0.189193

shrec2010_0001.noise.2:
        1: mean 0.017823, max error 0.122251
        2: mean 0.008985, max error 0.088037
        3: mean 0.001334, max error 0.141139
        4: mean -0.013420, max error 0.077807
        5: mean -0.053732, max error 0.060876

shrec2010_0001.scale.1:
        1: mean 0.273772, max error 0.566791
        2: mean -0.007603, max error 0.053150
        3: mean -0.004925, max error 0.044477
        4: mean -0.005765, max error 0.066461
        5: mean -0.001069, max error 0.037811

shrec2010_0001.scale.2:
        1: mean 0.108125, max error 0.241885
        2: mean -0.007578, max error 0.053298
        3: mean -0.004928, max error 0.044469
        4: mean -0.004977, max error 0.066284
        5: mean -0.001052, max error 0.037868

shrec2010_0001.topology.1:
        1: mean 0.061550, max error 0.286327
        2: mean -0.022882, max error 0.147108
        3: mean -0.004827, max error 0.318274
```

```
        4: mean 0.008132, max error 0.206154
        5: mean 0.033896, max error 0.391453

shrec2010_0001.topology.2:
        1: mean 0.079992, max error 0.288813
        2: mean -0.055920, max error 0.033784
        3: mean -0.037315, max error 0.210891
        4: mean 0.018111, max error 0.154698
        5: mean 0.060428, max error 0.345446

shrec2010_0001.shotnoise.1:
        1: mean 0.052903, max error 0.155468
        2: mean -0.006892, max error 0.054179
        3: mean -0.003770, max error 0.045084
        4: mean -0.005322, max error 0.063821
        5: mean -0.002492, max error 0.037817

shrec2010_0001.shotnoise.2:
        1: mean 0.052894, max error 0.155468
        2: mean -0.006003, max error 0.055461
        3: mean -0.002306, max error 0.045791
        4: mean -0.006528, max error 0.067352
        5: mean -0.004952, max error 0.037461
time needed: 274.735655


----------------------06-Sep-2014----------------------------
1 geodesic, 2 diffusion t=0.1, 3 diffusion t=1, 4 commute-time, 5 biharmonic
        max error, avg error for distances from point 10000

shrec2010_0001.isometry.1:
        1: mean 0.030846, max error 0.157851
        2: mean -0.003945, max error 0.016395
        3: mean -0.000907, max error 0.015667
        4: mean 0.002150, max error 0.021229
        5: mean 0.002161, max error 0.019082

shrec2010_0001.isometry.2:
        1: mean -0.005000, max error 0.031417
        2: mean 0.003238, max error 0.015826
        3: mean -0.001032, max error 0.010824
        4: mean -0.001842, max error 0.015633
        5: mean -0.008730, max error 0.006179

shrec2010_0001.microholes.1:
        1: mean 0.030677, max error 0.157851
        2: mean -0.003814, max error 0.016308
        3: mean -0.000781, max error 0.016549
        4: mean 0.001767, max error 0.024561
        5: mean 0.001257, max error 0.017667

shrec2010_0001.microholes.2:
        1: mean 0.030651, max error 0.157734
        2: mean -0.004340, max error 0.015931
        3: mean -0.001382, max error 0.015639
        4: mean 0.001234, max error 0.028072
```

```
        5: mean 0.000844, max error 0.017819

shrec2010_0001.localscale.1:
        1: mean 0.030772, max error 0.157796
        2: mean 0.030146, max error 0.264235
        3: mean 0.022919, max error 0.096145
        4: mean -0.002447, max error 0.091133
        5: mean -0.014621, max error 0.036541

shrec2010_0001.localscale.2:
        1: mean 0.028290, max error 0.155058
        2: mean 0.038001, max error 0.271383
        3: mean 0.037767, max error 0.185036
        4: mean -0.012095, max error 0.069622
        5: mean -0.023803, max error 0.082776

shrec2010_0001.noise.1:
        1: mean 0.020647, max error 0.144976
        2: mean -0.031932, max error 0.069149
        3: mean -0.024941, max error 0.006676
        4: mean 0.053672, max error 0.158807
        5: mean 0.056814, max error 0.178996

shrec2010_0001.noise.2:
        1: mean -0.004636, max error 0.108967
        2: mean -0.034027, max error 0.062236
        3: mean -0.012449, max error 0.096400
        4: mean -0.030644, max error 0.160857
        5: mean -0.076979, max error 0.016322

shrec2010_0001.scale.1:
        1: mean 0.181619, max error 0.569552
        2: mean -0.003931, max error 0.016258
        3: mean -0.000904, max error 0.015688
        4: mean 0.001645, max error 0.021032
        5: mean 0.002154, max error 0.019061

shrec2010_0001.scale.2:
        1: mean 0.068539, max error 0.247571
        2: mean -0.003964, max error 0.016288
        3: mean -0.000908, max error 0.015667
        4: mean 0.002320, max error 0.043452
        5: mean 0.002144, max error 0.019074

shrec2010_0001.topology.1:
        1: mean 0.030877, max error 0.157851
        2: mean -0.022931, max error 0.118553
        3: mean -0.008179, max error 0.198272
        4: mean 0.010914, max error 0.161556
        5: mean 0.027027, max error 0.297007

shrec2010_0001.topology.2:
        1: mean 0.030892, max error 0.157851
        2: mean -0.061486, max error 0.010259
        3: mean -0.039021, max error 0.089416
        4: mean 0.018065, max error 0.116740
```

```
        5: mean 0.046524, max error 0.256570

shrec2010_0001.shotnoise.1:
        1: mean 0.030841, max error 0.157851
        2: mean -0.002941, max error 0.017609
        3: mean 0.000041, max error 0.018642
        4: mean 0.000327, max error 0.043370
        5: mean 0.000913, max error 0.018118

shrec2010_0001.shotnoise.2:
        1: mean 0.030833, max error 0.157851
        2: mean -0.001835, max error 0.018692
        3: mean 0.001178, max error 0.022301
        4: mean 0.000137, max error 0.020188
        5: mean -0.001249, max error 0.016793
time needed: 285.138423


----------------------07-Sep-2014----------------------------
1 geodesic, 2 diffusion t=0.1, 3 diffusion t=1, 4 commute-time, 5 biharmonic
        max error, avg error for distances from point 10000

shrec2010_0001.holes.2:
        2: mean -0.015809, max error 0.226532
        3: mean -0.019862, max error 0.393604
        4: mean 0.000475, max error 0.153215
        5: mean 0.006196, max error 0.463801

shrec2010_0001.holes.3:
        2: mean -0.053096, max error 0.174638
        3: mean -0.069995, max error 0.299244
        4: mean -0.046189, max error 0.114623
        5: mean -0.062616, max error 0.340264
time needed: 339.136533


----------------------07-Sep-2014----------------------------
1 geodesic, 2 diffusion t=0.1, 3 diffusion t=1, 4 commute-time, 5 biharmonic
        max error, avg error for distances from point 910

shrec2010_0001.holes.2:
        2: mean -0.020886, max error 0.235264
        3: mean -0.031136, max error 0.649599
        4: mean -0.012485, max error 0.264872
        5: mean 0.000593, max error 0.660772

shrec2010_0001.holes.3:
        2: mean -0.083803, max error 0.247058
        3: mean -0.090159, max error 0.632709
        4: mean -0.073147, max error -2.016727
        5: mean -0.085645, max error 0.626859
time needed: 347.140476
```

Bibliography

[1] Multiple source/target exact geodesic algorithm for triangular meshes, 2008.

[2] Alexander Bronstein, Michael Bronstein, Umberto Castellani, Anastasia Dubrovina, Leonidas Guibas, Radu Horaud, Ron Kimmel, David Knossow, Etienne Von Lavante, Diana Mateus, et al. Shrec 2010: robust correspondence benchmark. In *Eurographics Workshop on 3D Object Retrieval (3DOR'10)*, 2010.

[3] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences of the United States of America*, 103(5):1168–1172, 2006.

[4] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer, 2008.

[5] Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33. American Mathematical Society Providence, 2001.

[6] Keenan Crane. Discrete differential geometry: The discrete laplacian, fall 2011.

[7] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat: a new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):152, 2013.

[8] Manfredo Perdigao Do Carmo and Manfredo Perdigao Do Carmo. *Differential geometry of curves and surfaces*, volume 2. Prentice-Hall Englewood Cliffs, 1976.

[9] Helin Dutagaci, Afzal Godil, Petros Daras, Apostolos Axenopoulos, G Litos, Stavroula Manolopoulou, Keita Goto, Tomohiro Yanagimachi, Yukinori Kurita, Shun Kawamura, et al. Shrec'11 track: generic shape retrieval. In *Proceedings of the 4th Eurographics conference on 3D Object Retrieval*, pages 65–69. Eurographics Association, 2011.

[10] Francois Fouss, Alain Pirotte, J-M Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):355–369, 2007.

[11] Philipp Herholz. General discrete laplace operators on polygonal meshes. diploma thesis, Humbold-Universität zu Berlin, 2012.

[12] Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, 1998.

[13] Stéphane S Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, 2004.

[14] Yaron Lipman, Raif M Rustamov, and Thomas A Funkhouser. Biharmonic distance. *ACM Transactions on Graphics (TOG)*, 29(3):27, 2010.

[15] Facundo Memoli. On the use of gromov-hausdorff distances for shape comparison. In *Eurographics symposium on point-based graphics*, pages 81–90. The Eurographics Association, 2007.

[16] Facundo Mémoli. Spectral gromov-wasserstein distances for shape matching. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 256–263. IEEE, 2009.

[17] Cleve B Moler and Gilbert W Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241–256, 1973.

[18] Gabriel Peyre. Toolbox graph, 2008.

[19] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[20] Emanuele Rodola, Alexander M Bronstein, Andrea Albarelli, Filippo Bergamasco, and Andrea Torsello. A game-theoretic approach to deformable shape matching. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 182–189. IEEE, 2012.

[21] Frank Schmidt, 2014.

[22] Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (TOG)*, 33(4):67, 2014.

[23] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[24] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J Gortler, and Hugues Hoppe. Fast exact and approximate geodesics on meshes. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 553–560. ACM, 2005.