CA' FOSCARI UNIVERSITY – VENICE
Department of Environmental Sciences, Informatics and Statistics
Second Cycle Degree Programme in Computer Science

Dissertation

Student: Andrea Gasparetto
Matriculation number: 812882

A Statistical Model of Riemmannian Metric Variation for Deformable Shape Analysis

Supervisor: Prof. Andrea Torsello

Academic Year 2011-2012

Contents

Abstract

Non-rigid transformations problems have recently been largely addressed by the researchers community due to their importance in various areas, such as medical research and automatic information retrieval systems. In this dissertation we use a novel technique to learn a statistical model based on Riemmannian metric variation on deformable shapes. The variations learned over different datasets are then used to build a statistical model of a certain shape that is independent from the pose of the shape itself. The statistical model can then be used to classify shapes that do not belong to the original dataset.

Chapter 1

Introduction

Shape analysis is a discipline which analyses and elaborates geometric objects. It includes several approaches used in automatic geometric shape analysis of digital objects in order to identify similarities or partial correspondences between different objects. It usually involves the elaboration of a discretization of the shape that has to be analysed. In fact, given a surface, we take its boundary representation to extrapolate information about the underlying surface. The boundary representation of a surface is usually a 3D model, often called mesh. Depending on the grain of the discretization (a mesh can be a dense representation of the underlying surface which means it is composed of an high number of elements, namely vertices, or sparse if the number of vertices is low), the object must be simplify before a comparison can be achieved. This simplification is usually called a shape descriptor (equivalently signature or fingerprint). The main goal of this process of simplification is to try to preserve in the shape descriptor most of the information of the surface. Shape analysis is applied to several different fields. For example, in archaeology (find

similar objects or missing parts), in security applications (faces reconnaissance), in medicine (recognize shape changes related to illness) or in virtual environment (on the 3D model market to identify objects for copyright purpose). Shape descriptors can be classify by their invariance with respect to certain transformations, like isometric transformations. In this case we speak of intrinsic shape descriptors. These descriptors do not change with different isometric embeddings of the shape. This type of descriptors comes in hand when applied to deformable objects (like a person in different poses) as these deformations do not involve much stretching but are nearly isometric. Such descriptors are commonly based on geodesic distances measures along the surface of an object or on other isometry invariant characteristics, such as the Laplace-Beltrami spectrum. Hence, different shape descriptors target different aspects of a shape and can be used for specific applications. The properties that we are looking for in a shape descriptor can be summarized as:

- Isometry: congruent solids (or isometric surfaces) should have the same shape descriptor being independent of the solid's given representation.
- Similarity: similar shaped solids should have similar descriptors. The shape descriptors should depend continuously on the shape deformation.
- Efficiency: the effort needed to compute those descriptors should be reasonable.
- Completeness: ideally, those fingerprints should give a complete characterization of the shape. One step further it would be desir-

able that those fingerprints could be used to reconstruct the solid.

- Compression: in addition it would also be desirable that the descriptor data should not be redundant.

- Physicality: finally, it would be nice if an intuitive geometric or physical interpretation of the meaning of the descriptors would be available.

As proved in the work of Reuter *et al.* [23], the spectrum of the Laplacian satisfies most of these properties. In particular, only the completeness property does not holds for this shape descriptor. The Reuter *et al.* work differs from the one we are presenting because it uses the Laplace-Beltrami operator directly on the underlying surface, while we use the mesh Laplacian based on a discretization of the surface (namely, the mesh). Our method start from these considerations and aim to build a statistical model, that allows the comparison between different shapes. It is a data-driven method in which the first phase consists of a learning phase whose input is a set of mesh representing the same shape in different poses. Once the descriptor of each mesh is computed, we use them to define a descriptor of the shape itself, *i.e.* we compute the descriptor of the dataset. This new descriptor can then be used to make comparisons between different shapes, namely meshes that do not belong to the dataset used for the learning phase. The comparison is done through the definition of probabilistic distributions, that can be easily compared with each other through a distribution dissimilarity criteria. This dissertation has the following structure:

- Chapter 2: in the second chapter we introduce briefly part of the theory involved in the definition of our method, like basic notions

of differentiable geometry, Riemannian geometry, differential operators and spectral theory.

- Chapter 3: in the third chapter we explain in deep our model, how the statistical method is defined and how we compute the distributions of the shapes.

- Chapter 4: in the fourth chapter we introduce the dataset used in the tests and propose a brief introduction to the SCAPE method, a data-driven method used to generate human shapes.

- Chapter 5: in the fifth chapter we present a schematic overview of the architecture of the project (namely, the implementation of the described model) where we explain the two main phase of the project and the inputs and the outputs of both.

- Chapter 6: in the sixth chapter we present the experimental results obtained using our model.

Chapter 2

Mathematical background

Differential geometry is a field of mathematics that uses the differential and integral calculus techniques in order to study geometric problems. Planes, curved spaces and surfaces theories on the three-dimensional Euclidean space represent the starting point for the development of this discipline. Differential geometry is probably as old as the others mathematical disciplines. Its diffusion is recorded in the period following the definition of fundamentals of calculus by Newton and Leibnitz. In the first half of the Nineteenth century, Gauss obtains a lot of results in the field of three-dimensional surfaces. In 1845 Riemann, a German mathematician, lays the foundation for a more abstract approach. In the end of that century, Levi-Civitae and Ricci developed the parallel translation concept in the classical language of tensors. This approach received an important contribution from Einstein and his studies on the relativity. During the first years of this century, E. Cartan introduced the first methods independent from a particular coordinate system. Chevalley's book *"The Theory of Lie Groups"* dated 1946, continued the work started by

Cartan about the basic concepts and notations which still have an impact on the modern study of the discipline. Surfaces differential geometry studies smooth surfaces with several additional structures, for example a Riemann metric. Surfaces have been extensively studied from different perspectives, both in their embedding in a Euclidean space and studying their properties determined uniquely from the inner distance of the surface measured on the curves which the surface is made of.

2.1 Differentiable geometry

A manifold is a structure that allows to extend in a natural way the notions of differential calculus. Intuitively, a manifold is an object that appears locally as an Euclidean space. In order to define a manifold in a more formal way, it is important to introduce firstly the topological space.

2.1.1 Topological space

A topological space is defined as a tuple containing a non-empty set X and a collection V of subsets of X, which are the open sets of X. For each subset contained in the collection V, the following properties must be satisfied:

1. X and $\emptyset$ belongs to V;

2. the union of each collection of elements in V belongs to V;

3. the intersection of a finite collection of elements in V belongs to V.

The neighbourhood of a point $p \in X$ is the subset U which contains an open set A such that $p \in A$. A function $f : X \to Y$ where X and Y are topological spaces is continuous if for each open set $A \subset Y$, $f^{-1}(A) \subset X$ is open. A topological space X is separable (or of Hausdorff) if for each pair of distinct points $p \in X$, $q \in X$, $p \neq q$, exist neighbourhoods U_p of p and U_q of q such that

$$U_p \cap U_q = \emptyset \tag{2.1}$$

In this case all the spaces will be separated. The cover of a set X is a collection of non-empty subsets

$$\mathcal{U} = \{U_i\}_{i \in I} \tag{2.2}$$

of X such that

$$X = \bigcup_{i \in I} U_i \tag{2.3}$$

A cover $\{U_i\}_{i \in I}$ is open if all the U_i are open.

2.1.2 Topological manifold

We are now able to formally define a topological manifold.

Definition 1. *A separated topological space X is a topological manifold of dimension n if it admits an open cover $U = \{U_i\}_{i \in I}$ such that each U_i is homomorphous to an open set of $\mathbb{R}^n$.*

In other words, each point of a manifold has a neighbourhood which is homomorphous to an open set of the Euclidean space. Let $X = \{X, U\}$ be

a manifold, for each $U_i \in U$ is possible to define a continuous application which takes the name of coordinate application

$$\varphi : U_i \to \mathbb{R}^n \tag{2.4}$$

The image $B_i = \varphi_i(U_i)$ is an open set of $\mathbb{R}^n$ and the application $\varphi_i : U_i \to B_i$ is an homomorphism. Let $q \in U_i$ we can define:

$$\varphi_i(q) = (x_1(q), ..., x_n(q)) \tag{2.5}$$

$x_k(q)$ is the k-th coordinate of the point q with respect to the chart (U_i, φ_i). The open sets U_i are then called coordinate open sets. The function φ_i coordinates U_i, namely it assigns unequivocally to each point of the open set a tuple of n scalars which determines the point. The map $\varphi_i : U_i \to B_i$ is called coordinate chart and the collection of charts U_i, φ_i is called atlas.

2.1.3 Differentiable manifold

In order to define formally what is a differentiable manifold, it is useful to introduce firstly the transition map concept. Given a topological manifold of dimension n and two charts U_i and U_j, let:

$$U_{ij} = U_i \cap U_j = U_j \tag{2.6}$$

Let φ_i and φ_j be two functions which can be used to map U_{ij} (namely, get the coordinates). Let:

$$B_{ij} = \varphi_i(U_{ij}) \subset \mathbb{R}^n, B_{ji} = \varphi_j(U_j i) \subset \mathbb{R}^n and \varphi_{ij} = \varphi_i \varphi_j^{-1} : B_{ji} \to \mathbb{R}^n \tag{2.7}$$

Then φ_{ij} is an homomorphism on the image which is the open set B_{ij} of $\mathbb{R}^n$. Furthermore, the function $\varphi_{ij} = \varphi_i \varphi_j^{-1}$ is defined on an open set of $\mathbb{R}^n$ with values in $\mathbb{R}^n$. It makes sense then require that this function is differentiable. Two atlases of X may not be compatible from the differentiable point of view, that is why is necessary to explicitly define the atlas.

Definition 2. *A differentiable manifold of class k, C^k (C^∞), is defined as a topological manifold X and an atlas V, $\{X, V\} = \{X, U_i, \varphi_i\}$ such that the change of the chart φ_{ij} are differentiable function of class C^k (C^∞). Topological manifolds can be considered as C^0 manifolds.*

It might be useful to add new charts to an atlas, but in order to do so we have to check that the differential structure is maintained. Let $\varphi : V \to \mathbb{R}^n$ be a chart, which means that V is an open set of X, $\varphi(V)$ is an open set of $\mathbb{R}^n$ and φ is an homomorphism on the image. $\{V, \varphi\}$ is compatible with the atlas V if for each $U_i \in V$ such that $U_i \cap V$ is non-empty, the applications:

$$\varphi_i \varphi^{-1} : \varphi(U_i \cap V) \to \mathbb{R}^n and \varphi \varphi_i^{-1} : \varphi(U_i \cap V) \to \mathbb{R}^n \tag{2.8}$$

are of class $C^k(C^\infty)$. Hence, given an atlas $\mathcal{V}$ we can make use of the new atlas $\mathcal{V} \cup \{\mathcal{V}, \varphi\}$. More generally, we can add to $\mathcal{V}$ all the charts that are compatible. In this way we obtain a maximal atlas.

Definition 3. *A differentiable structure $C^k(C^\infty)$ is constituted by a differential manifold and a maximal atlas $\{X, U\} = \{X, U_i, \varphi_i\}$ such that the transition map are still in class C^k (C^∞).*

2.1.4 Tangent space

The tangent space of a point of a manifold is a generalization of the concept of tangent plane of a surface and the extension of the definition vector from affine spaces to general manifold. We can see the concept of tangent space in differential topology as the space containing all the possible direction of the curves that belongs to a certain point of a differentiable manifold. The dimension of the tangent space is the same of the dimension of the associated manifold. The definition of tangent space can also be generalized to structures as algebraic manifold, where the dimension of the tangent space is at least equal to the manifold one. The points where these dimensions are equal are called non-singular, the others are singular. Tangent spaces of a manifold can be collapsed together to form the tangent fibre bundle, a new manifold with doubled dimension with respect to the original manifold. There are several possible way to define the tangent space to a point of a manifold. In this paragraph we will introduce the geometric one. Let M be a differentiable manifold, defined as a topological space coupled with a collection of charts in $\mathbb{R}^n$ and differentiable transition maps of class C^k with $k \geq 1$. Let x be a point of the manifold and

$$\varphi : U \to \mathbb{R}^n \tag{2.9}$$

a chart defined in an open set U which contains x. Let

$$\gamma_1 : (-\varepsilon, \varepsilon) \to M \tag{2.10}$$

$$\gamma_2 : (-\varepsilon, \varepsilon) \to M \tag{2.11}$$

two differentiable curves, that is $\varphi \cdot \gamma_1$ and $\varphi \cdot \gamma_2$ are derivable in zero. The curves γ_2 and γ_2 are called tangent in 0 if they are the same in 0 and even the derivative of the chart are equal.

$$\gamma_1(0) = \gamma_2(0) = x \tag{2.12}$$

$$(\varphi \cdot \gamma_1)'(0) = (\varphi \cdot \gamma_2)'(0) \tag{2.13}$$

The tangency between curves defines an equivalence relation. The equivalences class is called the tangent vectors of the manifold M in the point x and are denoted as $\gamma'(0)$. The set which contains all the tangent vectors are independent from the chart ϕ and take the name tangent space of M in the point x. Each tangent vector represents the direction of a curve that pass through the point x.

2.2 Riemannian geometry

Riemannian geometry studies the Riemannian manifold, smooth manifold with a Riemannian metric associated to it. A Riemannian metric generalizes the concept of distance in a manifold in a similar way to the distance that we can compute on an Euclidean space. Namely, Riemannian geometry generalizes the Euclidean geometry on spaces which are not necessarily flat, even though these spaces can be considered Euclidean spaces locally. Various concepts based on lengths, like the length of the arc of a curve, the area of a plane region and the volume of a solid, have an counterpart in the Riemannian geometry. The notion of directional derivative of a function of the multi-variable calculus has been extended in the Riemannian geometry to the notion of covariant derivative of a ten-

sor. A lot of concepts and analysis technique and differential equations have been generalized to adapt to Riemannian manifolds.

2.2.1 Tensors

The notion of tensor generalizes all the structures defined usually in linear algebra. An example of tensor is the vector. In differential geometry the tensors are used on a differentiable manifold to define geometric notion of distance, angle and volume. This results is reached through the use of a metric tensor, namely an inner product on the tangent space of each point. Let V be a vector scape of dimension n over the field K. The dual space V^* is the vector space made of all the linear functional

$$f : V \to K \tag{2.14}$$

The space V^* has the same dimension of V, namely n. The elements of V and V^* are called respectively vectors and covectors. A tensor is a multi-linear application

$$T : \underbrace{V \times ... \times V}_{h} \times \underbrace{V^* \times ... \times V^*}_{k} \to K \tag{2.15}$$

Hence, a tensor T associated to h vectors $v_1, ..., v_h$ and k covectors $w_1, ..., w_k$ a scalar

$$T(v_1, ..., v_h, w_1, ..., w_k) \tag{2.16}$$

The multi-linearity ensures the function to be linear in each component. The order (or type) of the tensor is the pair (h, k). The space T^k_h is provided with a natural vector space whose dimension is n^{h+k}.

After this brief introduction to tensors, we can introduce tensor fields. A tensor field is obtained by associating to each point of a differentiable manifold, like an open set of the Euclidean space $\mathbb{R}^n$, a tensor defined in the tangent space of the point. It is required that this tensor changes with continuity (or in a differentiable way) while changing the point on the manifold. This condition can be forced by asking that the coordinates of the tensors expressed in a chart, namely a local reference system, to change with continuity (*i.e.* in a differentiable way) when the point changes, and this condition does not depend on the chart used. A metric tensor is a tensor field which characterizes the geometry of a manifold. Thanks to the metric tensor it is possible to define the notions of distance, angle, length of a curve, geodesic, curvature. It is a tensor of order $(0, 2)$ which measure the inner product of two vectors in the tangent space of a point.

2.2.2 Connection

In mathematics, a connection is a central tool of differential geometry. It is a mathematical object which connects tangent spaces that belongs to different points of a differentiable manifold. The connection between these two tangent spaces is obtained through a curve which links them together. Intuitively, the connection defines a way to glide the tangent space along the curve. This operation is called parallel transport. Formally, a connection over a differential manifold is defined through the introduction of a differentiable object, which is called covariant derivative. Conceptually, connection and covariant derivative are the same thing. Furthermore, a connection can be defined in an analogous way for every

vector bundle on the manifold, other than the tangent bundle. In differential geometry, the *Levi-Civita connection* is the canonical connection used on Riemannian manifolds, and it is the only connection without torsion that preserves the metric. Thanks to the Levi-Civita connection, the metric tensor of the Riemannian manifold is enough to define uniquely more elaborated concepts, like covariant derivative, geodesic and parallel transport. Let (M, g) be a Riemannian manifold. A connection $\bigtriangledown$ is of Levi-Civita if the following properties holds.

1. There is no torsion, which means that $\bigtriangledown_x Y - \bigtriangledown_y X = [X, Y]$.

2. The metric is preserved, that is

$$\bigtriangledown_x(g(Y, Z)) = g(\bigtriangledown_x Y, Z) + g(Y, \bigtriangledown_x Z) \tag{2.17}$$

   or, equivalently

$$\bigtriangledown_x g = 0 \tag{2.18}$$

Finally, we will introduce some notations that will be used in the following paragraphs. Let $\bigtriangledown$ be a Levi-Civita connection of (M, g). We can express the metric and the Levi-Civita connection in terms of local coordinates. So let (U, x) be a chart and $\partial_1, ..., \partial_n$ be the corresponding vector fields on U. We now define $g_{ij} \in C^\infty(U)$ by

$$g_{ij} = g(\partial_i, \partial_j) \tag{2.19}$$

And Christoffel symbols $\Gamma_{ij}^k \in C^\infty(U)$ by

$$\bigtriangledown_{\partial_i} \partial_j = \sum_k \Gamma_{ij}^k \partial k \tag{2.20}$$

2.2.3 Riemannian manifold

A Riemannian manifold is a differentiable manifold where the tangent space of each point is endowed of an inner product which varies whit continuity when the point changes (more precisely, it varies smoothly). Thanks to the inner product we can compute distances between points, curves lengths, angles, areas and volumes, in a similar way we could have done in an Euclidean space.

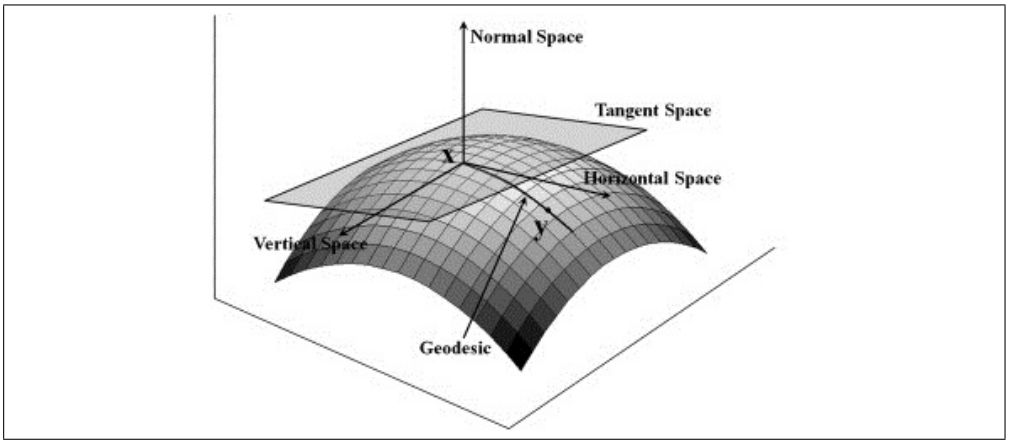

Figure 2.1: A Riemannian manifold and the spaces associated to it.

In particular, a Riemannian manifold is a metric space in which the concepts of geodesic is defined as the curve that realizes locally the distance between two points. In other words, a Riemannian manifold includes all classic geometric operators that we can find in the Euclidean geometry, even though their behaviour can be hugely different compared to the behaviour of the same operator in the plane. Recall what has been introduced in 2.1.1. We can now define a Riemannian metric. Let M be a manifold with $m \in M$, and let M_m denote the vector space of all tangent vectors at m.

Definition 4. *A Riemannian metric g on M is an inner product g_m such that, for all vector fields X and Y, the function*

$$m \mapsto g_m(X, Y) \tag{2.21}$$

is smooth.

We are now able to define a Riemannian manifold from a formally point of view.

Definition 5. *A Riemannian manifold can be defined as a differentiable manifold M endowed with a metric tensor g positive definite. The metric tensor g defines the inner product positive definite on the tangent space in each point of M. The Riemannian manifold is usually defined as the pair (M, g).*

For example, let $(,)$ denote the inner product on $\mathbb{R}^n$. An open $U \subset \mathbb{R}^n$ gets a Riemannian metric via $U_m \cong R^n$:

$$g_m(v, w) = (v, w) \tag{2.22}$$

2.2.4 Geodesic

A geodesic, in differential geometry, is a particular curve which describes locally the shortest trajectory between two points of a certain space. The space can be a surface, a more generic Riemannian manifold (a curve space) or a metric space. For example, in the Euclidean plane the geodesics are simple straight line, while on a sphere the geodesics are the great circle arcs. The concept of geodesic is strictly correlated to the Riemannian metric one, which is connected to the concept of

distance and of acceleration. Indeed, it can be thought as the path that a non-accelerated particle could do. The term "geodesic" comes from *geodesy*, the science which measure the dimensions and the shape of the terrestrial globe. In its original meaning, a geodesic was the shortest path between two points onto the Earth surface, namely a great circle arc. Hence, meridians and the Equator are geodesics, while the others circles of latitude are not. Let X be a generic metric space, a geodesic is a curve

$$\gamma : I \to X \tag{2.23}$$

defined on an interval I of the real straight line R, which realize locally the distance between points. More precisely, each point t that belongs to the interval I has a neighbourhood J in I such that for each pair of points t_1 and t_2 that belongs to J the following equivalence holds

$$d(\gamma(t_1), \gamma(t_2)) = \mathsf{t\_1 - t\_2} \tag{2.24}$$

If this equivalence is valid for each pair of points t_1 and t_2 in I, the geodesic is called minimizing geodesic or shortest path. In this case, the geodesic realizes the minimum distance not only locally, but also globally. A close geodesic is a curve

$$\gamma : S^1 \to X \tag{2.25}$$

defined on the circumference S^1, which is a geodesic if restricted to any arc contained in S^1. The shortest trajectory between two points on a curve space can be computed minimizing the curve length equation

through the standard technique of the variation calculus. The length of a curve

$$\gamma : [a, b] \to M \tag{2.26}$$

over a Riemannian manifold M can be computed using the equation

$$L(\gamma) = \int |\gamma'(t)| \, \mathrm{d}t \tag{2.27}$$

Hence the geodesic is a curve which, inside the space of all curves with fixed bounds, is a infimum, or more generally a stationary point, for the function length

$$L : \gamma \to L(\gamma) \tag{2.28}$$

It is possible to define an equivalent quantity, the so called energy of the curve:

$$E(\gamma) = \frac{1}{2} \int |\gamma'(t)|^2 \, \mathrm{d}t \tag{2.29}$$

Two different concepts that yields the same result: the geodesics are stationary point both for the length and for the energy. A useful example that explain this idea is that a rubber band stretched between two points contracts its length and minimize its potential energy. The resulting shape of the rubber band is however still a geodesic.

Let us now introduce the concept of existence and uniqueness of geodesic. For each point p which belongs to a Riemannian manifold M and for each non-null vector v of the tangent space T_p in p, it exists exactly one complete geodesic which pass through p and which is tangent to v. Which means that it exists $a, b > 0$ and a geodesic

$$\gamma : (-a, b) \to M \tag{2.30}$$

with $\gamma(0) = p$ and $\gamma'(0) = v$ such that every other geodesics with these two properties are still γ defined on a sub-interval of (a, b). The values a and b can also been infinite. The existence and uniqueness properties are due to the fact that a geodesic is a solution to a particular *second order Cauchy problem*. If the vector v is multiplied by a scalar, the corresponding geodesic results scaled. We can state then that, like in the plane geometry, for each point and for each direction exist an unique complete geodesic which passes through that point and which is oriented along that direction.

2.3 Laplace-Beltrami contribution

Before study in deep the arguments of this paragraph, a brief historical introduction is proposed about two mathematicians, whose contributions are essential to achieve the main goal of this dissertation. Those mathematicians were Pierre-Simon Laplace and Eugenio Beltrami. Pierre-Simon, marquis of Laplace, was a French mathematician, a physicist, an astronomer and a French noble who lived between the Eighteenth and Nineteenth century. He was one of the major scientists of the Napoleonic period. Its contributions in the field of mathematics, astronomy and probability theory are very important. For example, his work on the mathematical astronomy, resulted in five volumes that took about 26 years of his life and whose title is *"Mécanique Celeste"*, which changed the geometric study of mechanics developed by Newton in the one based on mathematical analysis. Eugenio Beltrami was an Italian mathemati-

cian who lived in the Mid-Nineteenth century, notable for his work concerning differential geometry and mathematical physics. His work was noted especially for clarity of exposition. He was the first to prove consistency of non-Euclidean geometry by modelling it on a surface of constant curvature, the pseudo-sphere, and in the interior of an n-dimensional unit sphere, the so-called Beltrami–Klein model. He also developed singular value decomposition for matrices, which has been subsequently rediscovered several times. Beltrami's use of differential calculus for problems of mathematical physics indirectly influenced development of tensor calculus by Gregorio Ricci-Curbastro and Tullio Levi-Civita.

2.3.1 Laplace operator

The Laplace operator, also known as Laplacian, is a differential operator given by the divergence of the gradient of a function in the Euclidean space. It is usually denoted as $\bigtriangledown \cdot \bigtriangledown$, $\bigtriangledown^2$ or $\bigtriangleup$. The Laplacian $\bigtriangleup f(p)$ of a function f in a point p is the rate at which the average value of f in a sphere centred on p deviate from $f(p)$ when the sphere radius grows. In a Cartesian coordinate system, the Laplacian is the sum of the second partial derivative of the function with respect to each independent variable. The Laplace operator is used in differential equations that describe numerous physical phenomena, like electric and gravitational potential, the heat diffusion and fluid propagation equation, or in the quantum mechanics. The Laplacian represents the flow density of the gradient flux of a function. For example, the net rate at which a chemical dissolved in a fluid move toward or away from a certain point is proportional to the Laplacian of the chemical concentration in that point, that is, symbol-

ically, the diffusion equation. For this reasons, the Laplace operator is widely used in science to model any kind of physical phenomena. In the field of image processing and computer vision, the Laplacian is used for example for blob and edges detection. After this short introduction, we can formally define this operator. As aforementioned, the Laplace operator is a second-order differential operator defined in an n-dimensional Euclidean space defined as the divergence $(\bigtriangledown\cdot)$ of the gradient $(\triangle f)$. Thus if f is a real-valued second-order differentiable function, the Laplacian of f is defined as:

$$\triangle f = \bigtriangledown^2 f = \bigtriangledown \cdot \bigtriangledown f \tag{2.31}$$

Alternatively, the Laplacian of f can be defined as the sum of each partial derivative of the second order in the Cartesian coordinates x_i

$$\triangle = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{2.32}$$

Thanks to the fact that the Laplace operator is a second-order differential operator, the Laplacian maps function of class C^k to function of class C^{k-2} when $k \geq 2$. Both the expression reported above define an operator whose type is

$$\triangle : C^k(\mathbb{R}^k) \to C^k(\mathbb{R}^\kappa) \tag{2.33}$$

or more generally an operator

$$\triangle : C^k(\Omega) \to C^k(\Omega) \tag{2.34}$$

for each open set Ω.

2.3.2 Laplace-Beltrami operator

Eugenio Beltrami, one of the mathematicians introduced at the beginning of this paragraph, has extended the results reached by Laplace in order to allow the use of the Laplace operator even in the field of differential geometry. During his researches, the Italian mathematicians has achieved an important goal generalizing the Laplace operator so it can be used on functions defined on Euclidean space surfaces and, more generically, on Riemannian manifolds. Like the Laplacian, the Laplace-Beltrami operator is defined as the divergence of the gradient. It is a linear operator taking functions into functions. The operator can be also extended in order to work with tensors, becoming the divergence of the covariant derivative. In our instance, the aforementioned covariant derivative is nothing but the Levi-Civita connection, which has been discussed in 2.2.2. The metric and Levi-Civita connection of a Riemannian manifold are all the components needed to generalize the familiar operators of vector calculus. A couple of definitions follows. Let TM denote the tangent bundle on M (namely, the disjoint union of the tangent spaces of M). Let $\Gamma(TM)$ be the vector space of all vector fields on M. Finally, let E be a tangent vector.

The gradient of $f \in C^\infty(M)$ is the vector field grad f such that, for $Y \in \Gamma(TM)$,

$$g(\text{grad } f, Y) = Yf \tag{2.35}$$

The divergence of $X \in \Gamma(TM)$ is the function div $f \in C^\infty(M)$ defined by

$$(\text{div } f)(m) = trace(\xi \to \bigtriangledown_{\xi} X) \tag{2.36}$$

We can define the Laplacian of $f \in C^{\infty}(M)$ as the function

$$\triangle f = \text{div grad } f \tag{2.37}$$

In a chart (U, x), set $g = det(g_{ij})$. Let (g^{ij}) be the matrix inverse to (g_{ij}). Then

$$\text{grad } f = \sum_{k} g^{ij}(\partial_i f)\partial_j \tag{2.38}$$

and, for $X = \sum_i X_i \partial_i$,

$$\text{div } X = \sum_i (\partial_i X_i + \sum_j \Gamma^i_{ij} X_j) = \frac{1}{\sqrt{\mathbf{g}}} \sum_j \partial_j (\sqrt{\mathbf{g}} X_j) \tag{2.39}$$

In particular

$$\bigtriangledown f = \frac{1}{\sqrt{\mathbf{g}}} \sum_{i,j} \partial_i (\sqrt{\mathbf{g}} g^{ij} \partial_j f) = \sum_{i,j} g^{ij}(\partial_i \partial_j f - \Gamma^k_{ij} \partial_k f) \tag{2.40}$$

2.3.3 Discrete Laplace operator

The Laplace-Beltrami operator, introduced in the previous paragraph, has been used extensively in the last years in the fields of graphic and geometric optimization. These applications include the manipulation of meshes, surface smoothing and shapes interpolation. To accomplish this purpose, several discretization and approximation schemas of the Laplace-Beltrami have been studied. The most common approaches were

based on the so called cotangent formula. The main problem of these approaches is that these schemas do not allow a point-wise convergence, which is a widely requested characteristic of several applications. *Belkin et al.* [3] proposed an algorithm for the Laplace operator approximation of a surface starting from a mesh which guarantees a point-wise convergence for an arbitrary mesh. In their work they show how the computed Laplacian is very close to the Laplace-Beltrami operator in each point of a surface. This result is really important because there are several arguments of computer graphics and geometric modelling which deal with bi-dimensional surfaces processing on tri-dimensional environments. These surfaces are typically represented as meshes, that is a set of coordinates represented by vertices and the information about how they are connected (vertices triangulation). The class of methods based on surface discrete differential geometry, and thus the use of the discrete Laplace operator, is used extensively for various tasks.

The Laplace-Beltrami operator, as mentioned in paragraph 2.3.2, is an object associated to a Riemannian manifold and which possess several interesting properties. One of the most interesting for this work is the capability of the eigenfunctions to form a base which reflect the surface geometry. Furthermore, the Laplace operator is strictly connected to the diffusion theory and to the heat equation over a surface. For these reasons the studies about the discretization of this operator are of particular interest. The objective of these methods is to obtain a discretization from a mesh which reflects narrowly the Laplacian of the surface. The principal contributions of the *Belkin et al.* work include a method to compute integrals on a surface using the associated mesh which allows the approximation of the Laplace-Beltrami operator on a surface granting

a point-wise convergence for an arbitrary mesh. Let us start from the algorithm used to compute the Laplace operator on a meshed surface. Let K be a mesh in $\mathbb{R}^3$. We denote with V the set of vertices of the mesh K. Given a face, a mesh or a surface X, let $Area(X)$ be the area of X. For a face $t \in K$, the number of vertices in t is denoted by $\#t$, while $V(t)$ is the set of vertices of t. The algorithm takes as input a function $f : V \in R$ and yield another function $L_K^h : V \to R$. L_K^h is thus the Laplace operator of the mesh K and for each vertex $w \in V$ it is computed as

$$L_K^h f(w) = \frac{1}{4\pi h^2} \sum_{t \in K} \frac{Area(t)}{\#t} \sum_{p \in V(t)} e^{-\frac{\|p-w\|^2}{4h})} (f(p) - f(w)) \tag{2.41}$$

The parameter h is positive and correspond to the size of the neighbourhood considered at each point. Assuming that K is quietly a dense mesh of the underlying surface S, L_K^h is close to the Laplacian of the surface ($\triangle_S$). Before continuing with the other obtained results, we have to introduce a couple of objects which will be used later. With functional Laplacian F_S^h we denote the intermediate object which allows to link the mesh Laplace operator L_K^h with the surface Laplacian $\triangle_S$. Given a point $w \in S$ and a function $f : S \to R$, this object is defined as

$$F_S^h f(w) = \frac{1}{4\pi h^2} \int_{x \in S} e^{-\frac{\|p-w\|^2}{4h})} (f(x) - f(w)) dv(x) \tag{2.42}$$

Furthermore, we need a measure which indicate how well a mesh approximate the underlying surface. Let thus M be the medial axis of S. For each $w \in S$, the local feature size at w (denoted with $lfs(w)$) is the distance from w to the medial axis M. The conditioning number $\rho(S)$

of S is the infimum of the lfs for any point of S. For each point $p \in S$, let n_p be the unitary normal of p and for each face $t \in K$ let n_t be the unitary normal of t (namely, the unitary normal of the plane that passes through t). We say that K is an (ξ, η)-approximation of S if for each face $t \in K$ the maximum distance between two points of t is $\xi\rho$ and if for each face $t \in K$ and vertex $p \in t$ the angle between n_t and n_p is at most η. Now we are able to proceed with the other obtained results. Intuitively, as the mesh approximate more closely the surface S, the mesh Laplace operator on K converges towards the Laplace-Beltrami operator of S. Let thus $K_{\xi,\eta}$ be a (ξ, η)-approximation of S. Let $h(\xi, \eta) = \xi^{\frac{1}{2.5+\alpha}} + \eta^{\frac{1}{1+\alpha}}$ for a fixed positive α. Then we have that for each $f \in C^2(S)$

$$\lim_{\xi,\eta\to 0} \sup_{K_{(\xi,\eta)}} \|L^{h(\xi,\eta)}_{K_{(\xi,\eta)}} f - \triangle_S f\|_\infty = 0 \tag{2.43}$$

where the supremum is taken from all the (ξ, η)-approximation of S. We conclude this paragraph introducing the approximation of integral on a surface. Let $d_S(x, y)$ be the geodesic distance between two point $x, y \in S$. Given a Lipschitz function, namely a function f such that $|f(x) - f(y)| \leq C|x - y|$ for each x and y and where C is a constant independent from both x and y, $g : S \to R$, let $L = Lip(g)$ be the Lipschitz constant of the function g. We have that $|g(x) - g(y)| \leq L d_S(x, y)$. Let $\|g\|_\infty = \sup_{x\in S} |g(x)|$. We can approximate $\int_S g \, \mathrm{d}v$ with the discrete sum

$$I_K g = \sum_{t\in K} \frac{Area(t)}{\#t} \sum_{v\in V(t)} g(v) \tag{2.44}$$

so that the above inequality holds:

$$\left| \int_S g \, \mathrm{d}v - I_K g \right| \leq 3(\rho L\xi + \|g\|_\infty (2\xi + \eta)^2) Area(S) \tag{2.45}$$

Moreover, let us suppose that g is twice differentiable, with the Hessian norm of g bounded by H. Then, for some constant C which depends only on S, the following inequality holds:

$$\left| \int_S g \, \mathrm{d}v - I_K g \right| \leq (CH\xi^2 + 3\|g\|_\infty (2\xi + \eta)^2) Area(S) \tag{2.46}$$

As a final note, let us introduce a different terminology that is often used while speaking about the discrete Laplace operator, which is the Laplacian matrix. For the case of a finite-dimensional graph (having a finite number of edges and vertices, *i.e.* a mesh), the discrete Laplace operator is more commonly called the Laplacian matrix.

2.4 Spectral theory

The spectral theory is a field of mathematics which extends the eigenvalues and eigenvectors theory of a single square matrix to a bigger variety of mathematical spaces. The researchers community has paid special attention to descriptors obtained through spectral decomposition of the Laplacian associated to a shape. The Laplacian-based descriptors represents indeed the state-of-art in terms of performance in several analysis operations on shapes. They are not only computationally efficient, but they are also isometry invariant for construction and they can be associated to a variety of transformations. Recently, a family of intrinsic geometric properties taking the name of diffusion geometry have grown in popularity. The studies on diffusion geometry are based on the theoretical work of *Berard et al.* [4] followed by the researches of *Coifman and Lafon* [10]. They suggest the adoption of the eigenvectors and eigen-

values of the Laplace-Beltrami operator associated to a shape in order to build the so called diffusion distances, which are invariant metrics. These distances revealed to be much more robust whit respect to geodesics since the latter are highly sensitive to topological noise. Recent advances in the discretization of the Laplace-Beltrami operator, that have been introduced in paragraph 2.3.3 led to a sets of robust and efficient computation tools.

2.4.1 Diffusion geometry

As aforementioned, diffusion geometry received a lot of attention from the researchers community in these years. One of its most diffuse uses has the objective to build pattern recognition applications. The analysis of non-rigid shapes and surfaces belongs to this set of applications. In particular, the diffusion geometry that derives from the heat propagation on the surface allows to define several similarity criteria on the shape which are intrinsic, that is invariant to inelastic deformation over the surface. Furthermore, the diffusion geometry has proved to be robust to topological noise, where other types of geometries, namely geodesic, have not. The first topic that will be addressed in this paragraph is the diffusion kernel. Let X be a shape, modelled as a compact Riemannian manifold. We denote with μ the standard measure of the are on X. The norm L^2 of a function f of X with respect to the measure μ is defined as

$$\|f\|_{L^2(X)} = \int_X f(x)\,\mathrm{d}\mu(x) \tag{2.47}$$

Given a metric $d : X \times X \to R_+$, the pair (X, d) forms a metric space while the triple (X, d, μ) a metric measure. A function $k : X \times X \to R$

is called diffusion kernel if it satisfies the following properties

- Non-negativity: $k(x, x,) \geq 0$
- Symmetry: $k(x, y) = k(y, x)$
- Positive-semidefiniteness: for each bounded f,

$$\int\int k(x, y)f(x)f(y)\,\mathrm{d}\mu(x)\,\mathrm{d}\mu(y) \geq 0 \tag{2.48}$$

- Square integrability:

$$\int\int k^2(x, y)\,\mathrm{d}\mu(x)\,\mathrm{d}\mu(y) \leq \infty \tag{2.49}$$

- Conservation:

$$\int k(x, y)\,\mathrm{d}\mu(y) = 1 \tag{2.50}$$

The diffusion kernel defines a linear operator

$$Kf = \int k(x, y)f(y)\,\mathrm{d}\mu(y) \tag{2.51}$$

which is known to be self-adjoint. A self-adjoint operator is an operator whose matrix is Hermitian, where an Hermitian matrix is one which is equal to its own conjugate transpose. Thanks to properties (4), K has a finite Hilbert norm, hence it is compact. This imply that this operator admits a discrete eigendecomposition $K\psi_i = \alpha\psi_i$ with eigenfunctions $\{\psi_i\}_{i=0}^{\infty}$ and eigenvalues $\{\alpha_i\}_{i=0}^{\infty}$. Because of (3) $\alpha_i \geq 0$, while thanks to (5) we have that $\alpha_i \leq 1$. The diffusion kernel admits then the following spectral decomposition property

$$k(x, y) = \sum_{i=0}^{\infty} \alpha_i\psi_i(x)\psi_i(y) \tag{2.52}$$

Since ψ_i forms an orthonormal base of $L^2(X)$,

$$\int\int k^2(x,y)\,\mathrm{d}\mu(x)\,\mathrm{d}\mu(y) = \sum \liminf_{i=0}^{\infty} \alpha_i^2 \tag{2.53}$$

Using these results, we can rework the properties (3-5) in the spectral form obtaining $0 \leq \alpha_i \leq 1$ and $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$. An important property of diffusion operators is that for each $t \geq 0$, the operator K^t is itself a diffusion operator with eigenbasis K and correspondent eigenvalues $\{\alpha_i^t\}_{i=0}^{\infty}$. There exist several ways to define a diffusion kernel and the linked diffusion operator. In this paragraph we will take into exam only operators that describe the heat propagation. The heat diffusion over a surface is governed by the heat equation

$$\left(\triangle_x + \frac{\partial}{\partial t}\right) u = 0 \tag{2.54}$$

where u is the heat distribution on the surface and $\triangle_x$ is the Laplace-Beltrami operator (see 2.3.2). For compact surfaces, the Laplace-Beltrami operator has discrete eigendecomposition of the form $\triangle_x \phi_i = \lambda_i \phi_i$. The diffusion operators associated to heat propagation processes are diagonalized by the eigenbasis of the Laplace-Beltrami operator, that is the K's having $\psi_i = \phi_i$. The correspondent diffusion kernel can be expressed as

$$k(x,y) = \sum_{i=0}^{\infty} K(\lambda_i)\phi_i(x)\phi_i(y) \tag{2.55}$$

where $K(\lambda)$ is a function such that $\alpha_i = K(\lambda_i)$. A particular case of a diffusion operator is represented by H^t defined by the family of transfer functions $H^t(\lambda) = e^{-t\lambda}$ and the associated heat kernel $h_t(x,y) =$

$\sum_{i=0}^{\infty} e^{-t\lambda}\phi_i(x)\phi_i(y)$. The heat kernel $h_t(x,y)$ is the solution of the heat equation with heat source on x at the time $t = 0$. The heat kernel will be introduced extensively in the paragraph 2.4.2. Since the diffusion kernel $k(x,y)$ measures the degree of proximity between x and y, it can be used to define a metric on X.

$$d^2(x,y) = \|k(x,\cdot) - k(y,\cdot)\|^2_{L^2(X)} \tag{2.56}$$

Such measure take the name of diffusion distance [10].

2.4.2 Heat kernel

The heat kernel can be defined in several ways. The classical definition of the heat kernel define it as the fundamental solution of the heat diffusion equation, which is

$$(\partial_t - \triangle)u = 0 \tag{2.57}$$

What we are seeking for is a smooth function $(t,x,y) \mapsto p(t,x,y)$ defined on $(0,\infty) \times M \times M$ such that, for each $y \in M$, $p(t,x,y)$ is the solution to the heat equation and for any $\phi \in C^\infty(M)$,

$$u(t,x) = \int_M p(t,x,y)\phi(y)\,\mathrm{d}y \tag{2.58}$$

tends to $\phi(x)$ when t tends to 0. In other words, the heat kernel allows to solve the Cauchy initial value problem defined above. In order to underline the link between the Laplace-Beltrami operator and the heat kernel with the work done in this study, it is necessary to firstly introduce the so called eigenvalues problem. Let M be a compact and

connected Riemannian manifold without boundary. The problem can be expressed as to find all real numbers λ for which exists a non-trivial solution $\phi \in L^2(M)$ to the equation

$$\triangle\phi + \lambda\phi = 0 \tag{2.59}$$

The real numbers λ take the name of eigenvalues, and the solutions ϕ are the correspondent eigenfunctions. Furthermore, for each λ, the eigenfunctions form a vector space known as eigenspace. The set of all these eigenvalues is called the spectrum of the problem. There exist two type of eigenvalues problem: the direct problem and the inverse problem. In the former we are interested in the geometric properties of M which allow us to snatch information about the eigenvalues and eigenvectors of the Laplacian. In the latter, we assume the knowledge about the spectrum of the problem and we try to extract information about the geometry of the manifold. This last approach is the one we are more interested in. To better comprehend the aforementioned problem we have firstly to adapt some of the analysis and equations partial differentiation techniques in order to make them work in the structures of the Riemannian geometry. Let us start with the introduction of the well-known partial differential equation that take the name of heat equation. Let M be an homogeneous medium and let Ω be a domain contained in M. Let $u(x,t)$ be a function which represents the temperature in the domain and whose parameters corresponds to a position and a time value. Let us suppose that there is neither heat nor refrigeration sources. In this case, the only way the heat can diffuse itself over the surface is through the border $\partial\Omega$. Symbolically, this concept can be expressed as

$$\frac{d}{dt}\int_{\Omega} u \, \mathrm{d}V = -\int_{\partial\Omega} F \cdot v \, \mathrm{d}A \tag{2.60}$$

where F is the flow density of heat and v is the outward unit normal vector field on M. The divergence theorem states that, given a smooth vector field X of M and v the outward unit normal vector field on ∂M, we have that

$$\int_{M} \mathrm{div}(x) \, \mathrm{d}V = \int_{\partial M} < X, v > \, \mathrm{d}A \tag{2.61}$$

Where $<,>$ is the metric associated with M. Assuming suitable smoothness and thanks to the divergence theorem, we obtain

$$\int_{\Omega} (u_1 - \triangle u) \, \mathrm{d}V = 0 \tag{2.62}$$

Considering that Ω is arbitrary, we can conclude that

$$u_t = \triangle u \tag{2.63}$$

This equation is also known as the heat equation. Let then M be a Riemannian manifold and let $\triangle$ be the corresponding Laplacian. We can define a differential operator L as $L = \frac{\partial}{\partial t - \triangle}$. Furthermore, we can add Dirichlet boundary condition and an initial temperature distribution $u_0(x)$, which leads to the final problem

$$\begin{cases} Lu = 0 & \text{for } x \in M \\ \partial u(x,t) = 0 & \text{for } x \in \partial M \\ \partial u(x,0) = u_0 & \text{for } x \in M \end{cases} \tag{2.64}$$

Now that we have laid the foundations, we can show the connection between spectrum, $\triangle$ and the geometry M, namely what is meant with

fundamental heat equation solution or, equivalently, what is the heat kernel. A function $p(x, y, t)$, continuous on $M \times M \times (0, \infty)$ and of class C^2 in the spatial variable and C^1 on the temporal variable, is the fundamental solution to the heat equation (or heat kernel) of the problem 2.64 if

- $L_x p = 0$ on $M \times M \times (0, \infty)$;
- $p(x, y, t) = 0$ if $x \in \partial M$;
- $\lim_{t \to 0+} \int_M p(x, y, t) u_0(y) \, \mathrm{d}V(y) = u_0(x)$ uniformly for each function u_0 which is continuous on M and that vanish in ∂M.

Thus $p(\cdot, y, \cdot)$ is the solution to the problem that we have seen above corresponding to an initial unitary temperature distribution positioned on y. We expect that $u(x, t) = \int_M p(x, y, t) u_0(y) \, \mathrm{d}V(y)$ will be the solution to the same problem with initial data u_0. To conclude this paragraph, let us show an example of a heat kernel of a Riemannian manifold. Let $(M, <, >)$ be a Riemannian manifold where $M = \mathbb{R}^n$ while $<, >$ is the canonical metric. Thus, the heat kernel is given by

$$p(x, y, t) = (4\pi t)^{-n/2)} e^{-\frac{\|x-y\|^2}{4t}} \tag{2.65}$$

Chapter 3

Statistical model definition

Now that all the theoretic pieces are in place, we can introduce the contributions that this study carry on and all the intermediate steps which allow us to obtain our experimental results. We can summarize these contributions as:

1. an efficient method to compute the manifold centroid of a dataset composed of several meshes, starting from the spectral decomposition of the models;

2. using part of the results obtained in (1), we define an alternative way to compute the geodesic distance between two matrices, *i.e.* the eigenvectors matrices that come from the spectral decomposition;

3. a method to compute the probabilistic distributions of each mesh and the relative statistical model.

Since the main goal of this work is to create a model which allows to compare a mesh which has not been used in the learning phase of

that model, we presents a method which allows to compare the spectral decomposition of an arbitrary mesh with the model created for a certain dataset.

3.1 Laplacian and eigendecomposition

In this paragraph we introduce the inputs and the outputs of the first two steps of our learning phase. For a better comprehension of the following steps, let us first introduce the Laplacian matrix from a different point of view with respect to the one explained in 2.3.3. The discrete representations of a surface can be defined as a graph G with n vertices V and a set of edges E which represents the mesh triangulation. The Laplacian matrix associated with the graph G is $L = (l_{i,j})_{n \times n}$ is defined as

$$L = D - A \tag{3.1}$$

where D is the degree matrix and A is the adjacency matrix of the graph G. A degree matrix is a diagonal matrix which contains information about the degree of each vertex. More formally, given a graph $G = (V, E)$, let $\|V\| = n$ the degree matrix D for G is a $n \times n$ square matrix defined as

$$d_{i,j} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where $deg(v_i)$ is the number of edges incident to the vertex v_i. Hence, the Laplacian matrix can be defined as

$$l_{i,j} = \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The Laplacian matrix has many interesting properties which have been exploited in this study. For a graph G and its Laplacian matrix L with eigenvalues $\lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{n-1}$ the following properties hold:

- L is always positive-semidefinite: $\forall i, \lambda_i \geq 0;\ \lambda_0 = 0$).
- The number of times 0 appears as an eigenvalue in the Laplacian matrix correspond to the number of connected components in the graph.
- L's eigenvalues real parts are positive.
- λ_0 is always 0 because every Laplacian matrix has an eigenvector $v_0 = [1, 1, ..., 1]$ that, for each row, adds the corresponding node's degree (from the diagonal) to a "-1" for each neighbour so that $Lv_0 = 0$.

In our study we assume that the meshes have only one connected component. Furthermore, for algorithmic reasons, we impose for each mesh that $\lambda_0 = 0$ and $\phi_0 = [1, 1, ..., 1]$, where λ_0 and ϕ_0 are respectively the first eigenvalue and the first eigenvector computed on the mesh. Once the Laplacian matrix has been computed for a mesh p, we can proceed with the spectral decomposition (also known as eigendecomposition, see 2.4.1). Intuitively, given a square matrix A with dimensions n, we are looking for a set of ϕs and for a set of λs which, for each $i = 1, ..., N$ satisfy the linear equation:

$$A\phi_i = \lambda_i \phi_i \tag{3.4}$$

where λ_i is an eigenvalue (a scalar) while ϕ_i is an eigenvector (a vector). We can thus factorize A as

$$A = \Phi \Lambda \Phi^{-1} \tag{3.5}$$

where Φ is the matrix whose columns are the eigenvectors of A, while *Lambda* is the diagonal matrix which contains all the eigenvalues ($\Lambda_{ii} = \lambda_i$). The optimization method used in 3.2 requires that all the eigenvectors are normalized. A vector is normalized when its length is 1. Hence, given ϕ a non-normalized eigenvector of dimension n, what we are looking for is

$$\hat{\phi} = \frac{\phi}{\|\phi\|} = \frac{\phi}{\sqrt{\sum_i^n \phi_i^2}} \tag{3.6}$$

Even if we have to compute the full Laplacian matrix in order to consequently compute its eigendecomposition, we are not interested in all the eigenvectors and all the eigenvalues. In fact we assume the greatest 200 eigenvalues (and the corresponding eigenvectors) are all we need to define our statistical model. Formally, let $M_j^i = (V, E)$ be the j-th mesh of the dataset i composed of n meshes, whit V the set of vertices and E the set of edges (the triangulation of the mesh). The first step of our method produce the Laplacian matrix L_j^i as output. The second step take as input the matrix L_j^i in order to produce the matrices ϕ_j^i and λ_j^i such that

$$L_j^i = \phi_j^i \lambda_j^i (\phi_j^i)^{-1} \tag{3.7}$$

The sets of all the matrices produced by the spectral decomposition form the output of the second step.

$$\Phi^i = \{\phi_j^i, \quad \text{with } j = 1, ..., n\} \tag{3.8}$$

$$\Lambda^i = \{\lambda_j^i, \quad \text{with } j = 1, ..., n\} \tag{3.9}$$

where ϕ_j^i and λ_j^i belong to $\mathbb{R}^{|V| \times 200}$. In the following paragraphs we will refer to the spectral decomposition of a mesh j that belongs to a dataset i with n vertices as the pair $(\phi_j^i, \lambda_j^i) \in \mathbb{R}^{n \times 200} \times \mathbb{R}^{n \times 200}$.

3.2 Manifold centroid

The goal of the second step of the learning phase is to compute the manifold centroid with respect to all the spectral decompositions of each mesh of a dataset i. In our method we approach this task as two separated optimization problems since we want to find the spectral decomposition of the manifold centroid (and thus not the manifold itself) starting from the sets of eigenvectors matrices and eigenvalues vectors of the meshes of the dataset. In other words we want to compute the spectral decomposition as defined in 3.1 starting from the spectral decomposition of the meshes of the dataset.

3.2.1 Centroid eigenvectors matrix

Intuitively, the eigenvectors centroid of a Riemannian manifold can be expressed as the matrix that minimize the squared geodesic distances

between all the N eigenvectors matrices of the meshes of a dataset. Let n be the number of vertices of a mesh of the dataset. The corresponding Laplacian matrix L is thus a $n \times n$ symmetric matrix. Remind the spectral decomposition of L

$$L = \Phi \Lambda \Phi^T \tag{3.10}$$

The spectral decomposition of a $n \times n$ real symmetric matrix allows to obtain a corresponding eigenvectors matrix whose eigenvectors are real, orthogonal to each other and with unitary norm. Hence, the matrix Φ is composed of orthogonal vectors. Furthermore, in the spectral decomposition we normalized the eigenvectors. This mean that all the eigenvectors are orthonormal to each other. A square matrix with real entries whose columns and rows are orthogonal unit vectors is called a orthogonal matrix. Equivalently, a matrix Q is orthogonal if its transpose is equal to its inverse:

$$Q^T = Q^{-1} \tag{3.11}$$

which means

$$Q^T Q = Q Q^T = I \tag{3.12}$$

where I is the identity matrix. As a linear transformation, an orthogonal matrix preserves the dot product of vectors, and therefore acts as an isometry of Euclidean space, such as a rotation or reflection. As a matter of fact, an orthogonal matrix is a rotation matrix, hence we can state the following:

$$\Phi \in SO(n) \tag{3.13}$$

where Φ is the orthogonal matrix which contains the eigenvectors of the Laplacian matrix L, while $SO(n)$ is the special orthogonal group which is formed all the orthogonal n-by-n matrices with determinant 1. We can now formally define the intuition expressed above.

$$\underset{\phi_0 \in SO(n)}{\operatorname{argmin}} \quad \sum_{i}^{N} d^2(\phi_i, \phi_0) \tag{3.14}$$

where ϕ_0 is the eigenvectors centroid that we are looking for, d represents the geodesic distance between its parameters and ϕ_i in the eigenvectors matrix of the i-th mesh of the dataset. Given two rotation matrices R_1 and R_2 we can define the geodesic distance as

$$d_g(R_1, R_2) = \|\log(R_1^T R_2)\|_F \tag{3.15}$$

where $\|A\|_F$, or equivalently $\|A\|_2$ is the Frobenius norm. This is true because the logarithmic map of a rotation give us the rotation angle. From a different point of view, we know that the logarithm map follow the direction of the geodesic. If we compute the norm of the logarithm of $R_1^T R_2$, we obtain the angle between them. Since both ϕ_0 and ϕ_i are rotation matrices, we can rewrite 3.14 as

$$\underset{\phi_0 \in SO(n)}{\operatorname{argmin}} \quad \sum_{i}^{N} \|\log(\phi_i^T \phi_0)\|_2^2 \tag{3.16}$$

which represents the sum of the squared rotation angles, or equivalently, the rotation quantity of each shape i of the dataset.

$$\underset{\phi_0 \in O(n)}{\operatorname{argmin}} \quad \sum_{i}^{N} \sum_{\lambda} \Theta_{i\lambda}^2 \tag{3.17}$$

Using the Taylor expansion of $cos\Theta$, we can compute Θ^2 as

$$\cos\Theta = 1 - \frac{\Theta^2}{2} + O(\Theta^4) \tag{3.18}$$

$$\Theta^2 = 2 - 2\cos\Theta + O(\Theta^4) \tag{3.19}$$

We can thus approximate 3.17 as

$$\underset{\phi_0 \in O(n)}{\operatorname{argmin}} \quad \sum_{i}^{N} 2\left(n - Tr\left(\frac{1}{2}(\phi_i^T \phi_0 + \phi_0^T \phi_i)\right)\right) + O(\theta_{\lambda_i}^4) \tag{3.20}$$

where Tr is the linear operator *trace*. In linear algebra, the trace of an n-by-n square matrix A is defined as the sum of the elements on the main diagonal of A

$$Tr(A) = \sum_{i=1}^{n} a_{ii} \tag{3.21}$$

Or, equivalently, the trace can be defined as the sum of the dot products, hence the sum of the cosine of Θ. This operator owns several interesting properties

1. It is a linear map, that is given two square matrices A and B and a scalar c

$$Tr(A + B) = Tr(A) + Tr(B) \tag{3.22}$$

$$Tr(cA) = c \cdot Tr(A) \tag{3.23}$$

2. A matrix and its transpose have the same trace: $Tr(A) = Tr(A^T)$

3. The trace of a product can be rewritten as the sum of entry-wise products of elements:

$$Tr(A^T B) = Tr(AB^T) = Tr(B^T A) = Tr(BA^T) = \sum_{i,j} A_{i,j} B_{i,j} \tag{3.24}$$

which means that the trace of a product of matrices behaves similarly to a dot product of vectors.

4. The trace is invariant under cyclic permutations:

$$Tr(ABCD) = Tr(BCDA) = Tr(CDAB) = Tr(DABC) \neq Tr(ABDC) \tag{3.25}$$

5. Finally, if A is a square n-by-n matrix with real or complex entries and if $\lambda_1, ..., \lambda_n$ are the eigenvalues of A, then

$$Tr(A) = \sum_i \lambda_i \tag{3.26}$$

Let us return to 3.20. Thanks to properties (1), we know that

$$Tr\left(\frac{1}{2}(\phi_i^T \phi_0 + \phi_0^T \phi_i)\right) = \frac{Tr(\phi_i^T \phi_0 + \phi_0^T \phi_i)}{2} = \frac{Tr(\phi_i^T \phi_0) + Tr(\phi_0^T \phi_i)}{2} \tag{3.27}$$

Furthermore, thanks to (3), we know that

$$Tr(\phi_i^T \phi_0) = Tr(\phi_0^T \phi_i) \tag{3.28}$$

We can then rework 3.20 as

$$\underset{\phi_0 \in O(n)}{\operatorname{argmin}} \quad 2Nn - 2\sum_{i}^{N} Tr\left(\phi_i^T \phi_0\right) \tag{3.29}$$

$$= \underset{\phi_0 \in O(n)}{\operatorname{argmin}} \quad 2Nn - 2Tr\left(\left(\sum_{i}^{N} \phi_i^T\right)\phi_0\right) \tag{3.30}$$

It is well known that the minimization of the difference between a constant quantity (*i.e.* $2Nn$) and a member which include the parameter used to minimize the whole equation is the maximization of the second member, we obtain that 3.29 is equivalent to

$$\underset{\phi_0 \in O(n)}{\operatorname{argmax}} \quad Tr\left(\left(\sum_{i}^{N} \phi_i\right)^T \phi_0\right) \tag{3.31}$$

Let $A = \sum_{i}^{N} \phi_i$. We can compute the singular value decomposition (SVD) of A obtaining

$$\operatorname{svd}(A) = U\Lambda V^T \tag{3.32}$$

The singular value decomposition is a factorization of a real or complex matrix which is closely related to the eigendecomposition that was introduced in 3.1. Suppose M is an $m \times n$ matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization of the form

$$M = U\Sigma V^* \tag{3.33}$$

where U is an $m \times m$ unitary matrix over K, the matrix Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal,

and the $n \times n$ unitary matrix V^* denotes the conjugate transpose of the $n \times n$ unitary matrix V. Such a factorization is called the singular value decomposition of M. The diagonal entries of Σ are known as the singular values of M. Note that in case of a real matrix M, the conjugate transpose of M (namely M^*) is equal to M^T. Hence, we obtain

$$\underset{\phi_0 \in O(n)}{\operatorname{argmax}} \quad Tr\left(\left(\sum_i^N \phi_i\right)^T \phi_0\right) = \underset{\phi_0 \in O(n)}{\operatorname{argmax}} \quad Tr\left(U \Lambda V^T \phi_0\right) \tag{3.34}$$

And for properties (4) of the trace operator we have that 3.34 is equal to

$$\underset{\phi_0 \in O(n)}{\operatorname{argmax}} \quad Tr\left(\Lambda V^T \phi_0 U\right) \tag{3.35}$$

Let $P = V^T \phi_0 U$, we can rewrite 3.35 as

$$\underset{\phi_0 \in O(n)}{\operatorname{argmax}} \quad \sum_k \Lambda_{k,k} P_{k,k} \tag{3.36}$$

which is maximized when $P = I$, where I is the identity matrix. We can conclude then that the eigenvectors centroid can be computed as

$$\phi_0 = V U^T \tag{3.37}$$

The approach introduced above has outline some problems during the experimentations. The occurred issues were due to the fact that during the computation of the eigenvectors matrix of the manifold centroid we did not take into consideration the variance between them. The result was that the latest eigenvectors (whose variance was larger than the first) dominate the computation yielding flattened results (*i.e.* almost constant

geodesic distance between the eigenvectors matrix of the centroid with respect to the matrices computed on the meshes of the dataset). In order to overcome this issue we introduced a sort of normalization for each eigenvectors based on the inverse of the variance computed on the whole dataset. This is an iterative process where, at each iteration, we estimate both the eigenvectors matrix of the manifold centroid and the variance matrix D. We assume that initially the variance matrix D is equal to the identity matrix I. The iterative process consists of the following steps:

1. Singular value decomposition of the sum of all eigenvectors matrices of each shape of the dataset multiplied for the variance matrix D:

$$\text{svd}\left(\left(\sum_{i}^{N}\phi_i\right)D\right) \tag{3.38}$$

2. Estimation of the eigenvectors matrix of the manifold centroid computed as

$$\phi_0 = UV^T \tag{3.39}$$

3. Computation of the new D matrix, using the last estimation of the matrix ϕ_0

$$D = (d_{ij}) \quad d_{ii} = \frac{1}{\text{Var}\Theta_i} \tag{3.40}$$

$$\text{Var}\Theta_i = \frac{2}{n} \quad \sum_{j}\left(1 - \phi_{i,j}^T\phi_{i,0}\right) \tag{3.41}$$

   where i is the i-th eigenvector while j represents the j-th shape of the dataset.

4. Reiterate from (1)

3.2.2 Centroid eigenvalues vector

The eigenvalues matrix of the manifold centroid is computed in a more straightforward way. It is simply the exponentiation of the average of the logarithm of each eigenvalues. The j-th eigenvalue of the manifold centroid is then equal to

$$\lambda_{j0} = e^{\frac{1}{N}\sum_{i}^{N}\log\lambda_{ji}} \tag{3.42}$$

where N is the number of meshes in the dataset.

3.3 Geodesic distance

In order to define the probabilistic distribution of a dataset, we have first to compute the geodesic distances between the eigenvectors matrix of the Laplacian of a mesh and the eigenvectors matrix of the manifold centroid. For a theoretic introduction to the geodesic of a Riemannian manifold, refer to 2.2.4. The approach used for the computation of the eigenvectors of the manifold centroid (see 3.2) comes in handy for the solution of this task. Indeed, we want to compute

$$\sum_{\lambda}\Theta_i^2 \tag{3.43}$$

for each mesh i of the dataset. We can thus define the geodesic distance between the eigenvectors matrix ϕ_i and ϕ_i as

$$d^2(\phi_i, \phi_0) = 2n - 2\mathrm{Tr}(\phi_i^T D\phi_0) \tag{3.44}$$

where n is the number of eigenvectors while D is the diagonal matrix which contains the inverse of the variance (see 3.40). The result obtained by the computation of this distance can then be used to compute the probabilistic distribution of the shapes.

Unfortunately, the approach introduced in this paragraph comes from a very strong assumption. We demand indeed that every mesh of the dataset used in the learning phase has a direct correspondence with all the others. In other words, every point of a shape that belongs to a dataset is in correspondence with the point whose index is the same in all the other meshes. Formally, given a mesh X and Y with $i = 1, ..., n$ vertices, $\forall x_i \in X$ and $\forall y_i \in Y$, x_i and y_i are in correspondence. The shape matching problem, which is discussed in the subsequent paragraph, remains a challenge. Hence, in order to achieve our goal, we had to use an alternative approach. Let ϕ_0 be the eigenvectors matrix of the manifold centroid computed on a dataset. $\phi_0 \in \mathbb{R}^{n_1 \times 200}$. Let $D \in \mathbb{R}^{200 \times 200}$ be the diagonal matrix introduced in 3.40. Finally, let ϕ_* be a shape that does not belong to the dataset used to compute the manifold centroid and whose dimensions are $n_2 \times 200$. We define

$$W = \phi_0 D \phi_*^T \tag{3.45}$$

as the weights matrix to be used in a maximum bipartite matching. Thus, the problem that we have to solve becomes an *assignment problem*, namely we have to find a maximum weight matching in a weighted bipartite graph. In other words, assume that we have a certain number of jobs to do, and we have to choose from an equal number of workers that are able to do them. Any worker can be assigned to perform any task, incurring some cost that may vary depending on the assignment. It

is required to perform all tasks by assigning exactly one worker to each task in such a way that the total cost of the assignment is minimized. In order to solve this problem, we choose the *Hungarian algorithm* ([20]). In our particular instance, it is convenient to explain a matrix-wise interpretation of the steps it performs. As aforementioned, the main goal of the Hungarian algorithm is to to find a minimum edge weight matching given a $m \times n$ edge weight matrix. Hence, the input of the algorithm will be, in our instance, the weight matrix $W = \phi_0 D \phi_*^T$. The output produced are a pair of matrix (M, C), where M is an $m \times n$ matrix with ones in the place of the matchings and zeros elsewhere, while C is the cost of the minimum matching. Since our goal is to maximize this cost, the weights matrix W will be negated. The algorithm then executes the following steps:

- **STEP 1**: find the smallest number of each row and subtract that minimum from its row. This will lead to at least one zero in that row. Repeating this procedure for all rows, we obtain a matrix with at least one zero per row. Now we try to assign a matching such that each row is matched with only one column and the penalty incurred in each case is zero;

- **STEP 2**: if we are unable to make an assignment, we do the same as in *STEP 1* but column-wise;

- **STEP 3**: if it is still not possible to assign then all zeros in the matrix must be covered by marking as few rows and/or columns as possible. The goal is to mark all the rows and columns which contains a zero;

- **STEP 4**: from the elements that are left, we have now to find the lowest value. Then we subtract this from every unmarked element and add it to every element covered by two lines (that is, an element whose row and column contains a 0). The actions illustrated in *steps (3-4)* are then repeated until an assignment is possible, which is when the minimum number of lines used to cover all the 0's is equal to $\max(m, n)$.

We repeat these steps until we find a perfect matching, in which case it gives a minimum cost assignment.

Practically speaking, we are looking for the best permutation P of ϕ_*^T in order to maximize the trace

$$\underset{P}{\operatorname{argmax}} \quad Tr(WP) \tag{3.46}$$

The costs induced by this permutation can then be used in the computation of the geodesic distance between the eigenvectors of the Laplacian of a shape and the ones associated with the manifold centroid computed in 3.2.

3.4 Embedding and isometries

The main problem found in the aforementioned approach is that it holds only if the embedding of the Laplacian in a lower dimensional space is unique. In other words, it holds up to isometries (where an isometry is a function between two metric spaces which preserves the distances, *i.e.* rigid transformations like translations, rotations and reflections of the

space). Unfortunately, this is not true when speaking about a Riemannin manifold isometric embedding on a lower dimensional space like in our case. Thus, in order to accomplish our goal, we have to find the rigid transformation for each mesh of a dataset that align the mesh with the others (and consequently, the transformation that align each mesh with the manifold centroid of the dataset). Let us introduce an alternative form of the problem seen in section 3.2. The spectral decomposition of the Laplacian of a mesh has been defined as $L = \Phi\Lambda\Phi^T$. Let $X_i = \phi_i$ be a matrix associated to the mesh i where each columns are an embedding on a space of dimension 200. Let ϕ_0 be the matrix containing the first 200 eigenvectors of the manifold centroid and let R_i be the rigid transformation which align the mesh i with the other meshes of the dataset. We can rewrite 3.16 as

$$\underset{\phi_0, R_i}{\operatorname{argmin}} \quad \sum_i^N \|R_i X_i - \phi_0^T\|_2^2 \tag{3.47}$$

where $\|\cdot\|_2^2$ is the Frobenius norm. The Frobenius norm of A is defined as $\|A\|_2^2 = \mathrm{Tr}(A^T A)$, we can then rewrite 3.47 as

$$\underset{\phi_0, R_i}{\operatorname{argmin}} \quad \sum_i^N \|R_i X_i\|_2^2 + \sum_i^N \|\phi_0^T\|_2^2 - 2\mathrm{Tr}\left(\left(\sum_i^N R_i X_i\right)\phi_0^T\right) \tag{3.48}$$

Since $\|R_i X_i\|_2^2 = \mathrm{Tr}(X_i^T R_i^T R_i X_i)$ (where R_i and X_i are rotation matrices) and $\|\phi_0\|_2^2 = \mathrm{Tr}(\phi_0^T \phi_0)$ (where ϕ_0 is a rotation matrix) we can conclude, thanks to 3.12, that 3.48 is equal to

$$\underset{\phi_0, R_i}{\operatorname{argmin}} \quad 2Nn - 2\mathrm{Tr}\left(\left(\sum_i^N R_i X_i\right)\phi_0^T\right) \tag{3.49}$$

which has the form of the problem 3.29. We first initialize R_i and λ_0 to the identity matrix I. The steps of the optimization process are:

1. **Step 1**: ϕ_0 optimization.

$$\phi_0 = \underset{\phi_0}{\text{argmax}} \quad \text{Tr}\left(\left(\sum_i^N R_i X_i\right)\phi_0^T\right) \tag{3.50}$$

$$\text{svd}\left(\sum_i^N R_i X_i\right) = ULV^T \tag{3.51}$$

$$\Rightarrow \quad \phi_0 = UV^T \tag{3.52}$$

2. **Step 2**: R_i optimization.

$$R_i = \underset{R_i}{\text{argmax}} \quad \text{Tr}\left(\sum_i^N X_i \phi_0^T R_i\right) \tag{3.53}$$

$$\text{svd}\left(\sum_i^N X_i \phi_0^T\right) = ULV^T \tag{3.54}$$

$$\Rightarrow \quad R_i = VU^T \tag{3.55}$$

while λ_0 is computed as explained in 3.2.2. We can define a second model in which we can include in the optimization problem the computation of λ_0, the diagonal eigenvalues matrix of the manifold centroid. In order to do so, we have firstly to change the X_i value. Remember that the spectral decomposition of the Laplacian of a mesh has been defined as $L = \Phi\Lambda\Phi^T$. Let $X_i = \lambda_i^{\frac{1}{2}}\phi_i$ be a matrix associated of the mesh i where each columns is an embedding on a space of dimension 200 and such that $X_i^T X_i = L$. We obtain

$$\underset{\phi_0,\lambda_0,R_i}{\operatorname{argmin}} \quad \sum_i^N \|R_i\lambda_0 X_i - \phi_0\|_2^2 \tag{3.56}$$

$$= \underset{\phi_0,\lambda_0,R_i}{\operatorname{argmin}} \quad \sum_i^N \|R_i\lambda_0 X_i\|_2^2 + Nn - 2\mathrm{Tr}\left(\sum_i^N R_i\lambda_0 X_i\phi_0\right) \tag{3.57}$$

$$= \underset{\phi_0,\lambda_0,R_i}{\operatorname{argmin}} \quad \mathrm{Tr}\left(\sum_i^N \lambda_0 X_i X_i^T \lambda_0^T\right) + Nn - 2\mathrm{Tr}\left(\sum_i^N R_i\lambda_0 X_i\phi_0\right) \tag{3.58}$$

Note that in the final formulation of the problem $\mathrm{Tr}\left(\sum_i^N \lambda_0 X_i X_i^T \lambda_0^T\right)$ does not depend anymore on R_i, hence it can be optimized separately. We can now use an iterative process which maximize all the arguments of the problem 3.56. We first initialize R_i and λ_0 to the identity matrix I. The steps of the optimization process becomes:

1. **Step 1**: ϕ_0 optimization.

$$\underset{\phi_0}{\operatorname{argmax}} \quad \mathrm{Tr}\left(\left(\sum_i^N R_i\lambda_0 X_i\right)\phi_0^T\right) \tag{3.59}$$

$$\mathrm{svd}\left(\left(\sum_i^N R_i\lambda_0 X_i\right)\right) = ULV^T \tag{3.60}$$

$$\Rightarrow \quad \phi_0 = UV^T \tag{3.61}$$

2. **Step 2**: R_i optimization.

$$\underset{R_i}{\text{argmax}} \quad \text{Tr}\left(\sum_i^N \lambda_0 X_i \phi_0^T R_i\right) \tag{3.62}$$

$$\text{svd}\left(\lambda_0 X_i \phi_0^T\right) = ULV^T \tag{3.63}$$

$$\Rightarrow \quad R_i = UV^T \tag{3.64}$$

3. **Step 3**: λ_0 optimization.

$$\underset{\lambda_0}{\text{argmin}} \quad \text{Tr}\left(\sum_i^N R_i \lambda_0 X_i X_i^T \lambda_0^T R_i^T\right) - 2\left(\sum_i^N R_i \lambda_0 X_i \phi_0^T\right) \tag{3.65}$$

$$\Rightarrow \lambda_0 = \text{diag}\left(\frac{\text{diag}\left(\sum_i^N X_i \phi_0^T R_i\right)}{\text{diag}\left(\sum_i^N X_i X_i^T\right)}\right) \tag{3.66}$$

where *diag* is a function whose input is a vector d of dimension n and the output is a diagonal matrix D of dimension $n \times n$ such that

$$D_{i,j} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.67}$$

This process will converge after a low number of iteration through its steps. Once the parameters of each mesh are learned, we are able to compute the geodesic distance between each mesh of the dataset and the manifold centroid. Let M_0 be the embedding of the manifold centroid, such that $M_0 = \phi_0 \lambda_0 \phi_0^T$, we can compute the geodesic distance between M_i and M_0 as

$$d_g(M_i, M_0) = \|R_i\lambda_0 X_i - \phi_0^T\|_2^2 \tag{3.68}$$

All the issues explained in 3.3 still holds when it comes to compute the geodesic distance between the manifold centroid of a dataset and a shape which is external to it. We have still to find a way to put the embedding of each spectral decomposition in correspondence. Furthermore, we have to find the rigid transformation which align the two mesh. Let X^* the embedding of the mesh which does not belong to the dataset. Let R the rigid transformation that align X^* to the meshes of the dataset we are considering. Finally, let P the permutation matrix of X^*. We can thus define the geodesic distance between the manifold centroid and X^* as

$$\|R\lambda_0 X^* P - \phi_0^T\|_2^2 \tag{3.69}$$

In order to compute P and R we use again an iterative process. The problem we looking to solve is then

$$\underset{P,R}{\text{argmin}} \quad \|R_i\lambda_0 X^* P - \phi_0^T\|_2^2 \tag{3.70}$$

which is equivalent to

$$\underset{P,R}{\text{argmax}} \quad \text{Tr}(R_i\lambda_0 X^* P\phi_0^T) = \underset{P,R}{\text{argmax}} \quad \text{Tr}(\phi_0^T R_i\lambda_0 X^* P) \tag{3.71}$$

After initializing $R = I$, the optimization process can be defined as

1. **Step 1**: compute the permutation matrix P using the already introduced Hungarian algorithm (see 3.3). The weight matrix in this case is defined as $W = \phi_0^T R_i\lambda_0 X^*$

2. **Step 2**: R optimization

$$\text{svd}(\lambda_0 X^* P \phi_0^T) = U L V^T \tag{3.72}$$

$$\Rightarrow R = V U^T \tag{3.73}$$

which will converge after a low number of iterations (≈ 5).

3.5 Distributions computation

The direct comparison between metrics computed on different meshes is often problematic because it needs to find the correspondences between them first. Even if in the last years significant progress have been made in this field, finding a meaningful correspondence between shapes remains a challenge. This problem can be generalized as, given two shapes find a relation (or mapping) between their elements, namely their vertices. There are many different approaches to the problem, based on the goal the author aims at. For example, we can search for a full correspondence between shapes (see figure 3.1), or only in part. These kind of problems are usually hard to solve because the correspondence search spaces are usually pretty big, at least when the mesh is a dense discretization of the underlying surface. The global structure of the shapes must be considered and even the semantics of their parts play a role in the process to obtain meaningful solutions.

Fortunately, there is an alternative way that can be use to circumvent this problem, and this way involves the use of distributions as comparison parameters. Representing a metric by its distribution, and measuring

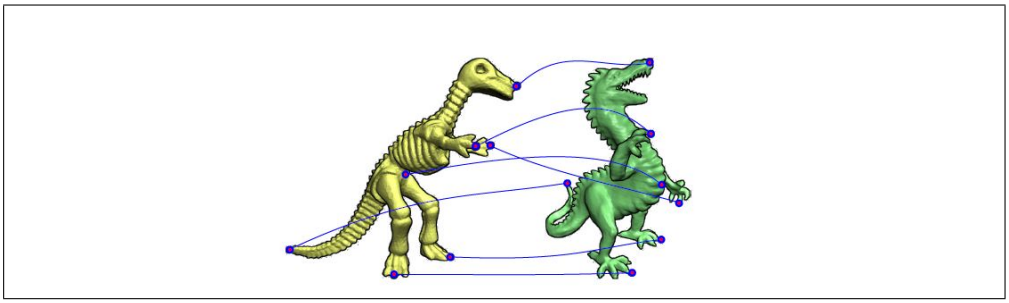

Figure 3.1: Correspondences example between two different shapes.

the similarity of two shape by comparing the distributions of the respective metrics do the trick (see [8]). We can then use one of the standard distribution dissimilarity criteria to reach our goal. We assume that the geodesic distances between the eigenvectors matrices of meshes that belong to the same dataset are distributed as a Γ-distribution. The Gamma distribution of probability is defined on the real non-negative numbers $[0, \infty[$. There are two ways to parametrize this distribution. The first is through the pair of positive numbers (k, θ), the second is through the pair of positive numbers (α, β). These parameters are tied by the relations $\alpha = k$ and $\beta = \frac{1}{\theta}$. In our work we used the former. The density function of a gamma distribution is defined as

$$f(x) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} e - \frac{x}{\theta} \tag{3.74}$$

where $\Gamma(k)$ is the Gamma function. Firstly, we have to estimate the value of parameter k given the geodesic distances between the meshes with respect to the manifold centroid of a dataset. Unfortunately, there is no closed-form solution for k. Hence, we have to approximate it. Let

$$s = \ln\left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{x_i}\right) - \frac{1}{N}\sum_{i=1}^{N}\ln(x_i) \tag{3.75}$$

then k is approximately

$$k \approx \frac{3 - s + \sqrt{(s-3)^2 + 24s}}{12s} \tag{3.76}$$

where x_i is the squared geodesic distance between a mesh i and the manifold centroid, k is the first parameter of the Gamma distribution and N is the number of meshes the dataset is made of. Once we have estimated the k parameter, we want to compute the maximum likelihood estimator of the θ parameter, which can be found taking the derivative of the log-likelihood function and setting it to zero. Hence, we compute θ as

$$\theta = \frac{1}{kN}\sum_{i=1}^{N}\sqrt{x_i} \tag{3.77}$$

Once we have computed both θ and k we are able to proceed to the second step, namely the computation of distribution of the eigenvalues of a shape. We assume that the eigenvalues are distributed as a normal distribution. A normal (or Gaussian) distribution is a continuous probability distribution defined as

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.78}$$

where μ is the mean of the distribution, while σ is the standard deviation (and thus σ^2 is the variance). As seen before, the first step is to compute these two parameters. Given the vector λ_0 containing the eigenvalues of the spectral decomposition of the manifold centroid, we denote

with λ_{i0} the i-th eigenvalue. We can define the mean of the distribution as

$$\mu_i = \log(\lambda_{i0}) \tag{3.79}$$

Let λ_{ij} be the i-th eigenvalue of the j-th mesh of the dataset. We can define the standard deviation of the mesh j as

$$\sigma_j = \frac{\sqrt{\sum_i^n (\log(\lambda_{ij}) - \log(\lambda_{i0}))^2}}{N-1} \tag{3.80}$$

where N is the number of meshes contained in the dataset, while n is the number of eigenvalues. We are now able to define the probability density function as

$$\lambda_{d_{ij}} = \frac{1}{\sigma_j x_{ij}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x_{ij})-\mu_i}{\sigma_j}\right)^2} \tag{3.81}$$

Once we have computed both the densities of the eigenvectors and eigenvalues of a given mesh, we are able to compute its density as

$$p(j) = \left(\prod_i \lambda_{d_{ij}}\right) \Gamma(k,\theta)(g_{d_j}) \tag{3.82}$$

where g_{d_j} is the geodesic distance between the mesh j and the manifold centroid of the dataset we are considering.

Chapter 4

Learning and inference

As mentioned in chapter 1, the two main results of this work are achieved through several distinct steps, subdivided in 2 different phases. The first phase is the learning phase. Our main goal is to define in an efficient way the computation of the probabilistic distributions of each datasets. These distributions will define our statistical model. The inputs of the learning phase are the datasets used. See chapter 5 for a brief introduction to the datasets used in this work.

4.1 Learning phase

The steps that characterize the learning phase are the following:

1. **Computation of the Laplacian matrix.** The first step aims to compute the so called Laplacian matrix. As explained in paragraph 2.3.3, the Laplacian matrix is nothing more than the discrete Laplace operator applied to a finite-dimensional graph. In

our case, this graph is the mesh itself, since it satisfies the properties required by a finite-dimensional graph, that is a finite number of edges (mesh triangulation) and vertices. Therefore, the Laplacian matrix is the Riemannian manifold that we are working with, while the Riemannian metric associated with it is the diffusion distance. This computation is done for each mesh of the datasets, and the set of matrices computed forms the input for the next step of the learning phase.

2. **Spectral decomposition.** Once the Laplacian matrices of each mesh are computed, we are able to apply the canonical matrix decomposition derived from the spectral theorem which is usually called spectral decomposition, eigenvalue decomposition or eigendecomposition. We can thus rewrite the Laplacian matrix L as

$$L = \Phi \Lambda \Phi^{-1} \tag{4.1}$$

where Λ is a diagonal matrix of the eigenvalues of L arranged in the ascending order while Φ is the matrix with the corresponding orthonormal eigenvectors. This step is necessary in order to achieve the goal of the next one, since our method computes the manifold centroid as a composition of the mean of both the eigenvector matrices and eigenvalue matrices. The outputs produced by this step are the set $\mathbf{\Phi}^i$, which is the set of all eigenvector matrices and the set $\mathbf{\Lambda}^i$ which contains all the eigenvalues matrices both computed on all the meshes which belong to the dataset i.

3. **Computation of the manifold centroid.** In order to define the statistical model, the computation of the manifold centroid has a

central role in our method. In the previous step we computed the sets $\mathbf{\Lambda}^i$ and $\mathbf{\Phi}^i$ relative to the dataset i. The computed matrices can then be used to compute the centroid as

$$M_C^i = \Phi_0 \Lambda_0 \Phi_0^{-1} \tag{4.2}$$

Where Φ_0 and Λ_0 are the non-standard mean of the set $\mathbf{\Phi}^i$ and $\mathbf{\Lambda}^i$ respectively. The method used in this work to compute the mean of these sets is explained in deep in paragraph 3.4.

4. **Computation of geodesic distances.** The input of this step is both the output of step (2) and step (3). Once the manifold centroid M_C^i is computed, we can use it to compute the geodesic distance (see 2.2.4) between the Laplacian matrix of each mesh and the manifold centroid. The output is thus a vector g_{ds}^i

$$g_{ds}^i = \{g_{d_j}^i\} \tag{4.3}$$

where $g_{d_j}^i$ is the geodesic distance between the M_C^i and the mesh Laplacian of the j-th mesh of the i-th dataset. This step of the learning phase is explained in paragraph 3.3.

5. **Distributions computation.** With the geodesic distances computed, we are able to compute the distributions of the meshes, treating eigenvalues and eigenvectors separated (that is, we assume that they both have their own kind of distribution). More precisely, we assume that eigenvectors follow a Γ-distribution, while eigenvalues a Gaussian one. See 3.5 for more details.

4.2 Use of the model

The outputs produced by the learning phase are then the distributions of the meshes which identifies a statistical space and the shape descriptor which have been computed in the second step of the learning phase. Once the statistical model has been learned, we are able to use it to classify shapes that are not in the initial dataset. And here starts the second phase, which involves the use of a set of tools that we have created and which allowed us to obtained our experimental results (which are reported in chapter 6). This phase is characterized by the following steps:

1. **Laplacian computation and spectral decomposition**. As in the previous phase, in the first step we compute the Laplacian matrix associated with the shape we are considering and then we compute its spectral decomposition. The outputs of this phase are then an eigenvectors matrix ϕ^* and an eigenvalues diagonal matrix λ^* such that $\phi^*\lambda^*(\phi^*)^T$ is equal to the Laplacian L^* of the shape $*$.

2. **Geodesic distance computation**. In the second step we compute the geodesic distance between the shape $*$ and the manifold centroid of the considered dataset as seen in 3.4.

Once we have computed the geodesic distance, we can use them to build a confusion matrix or to find the probability that this shape belongs to a certain dataset. The results obtained from the application of our method to the input datasets are presented in chapter 6.

Chapter 5

Datasets

The learning phase of the project needs several different models representing objects with the same shape but in different poses. The basic idea is to collect a certain number of shapes which are just in correspondence between them in terms of vertices. Starting from this dataset it is then possible to build the statistical model which will form the final results of this work. For this purpose two different datasets have been selected, the SCAPE and the TOSCA datasets.

5.1 SCAPE

The first dataset is a set of 70 models representing the same person in different poses. These models are the result of the SCAPE method application, where SCAPE stands for *Shape Completion and Animation of People*. It is a data-driven method used to build a model of the human shape which varies both in the shape of the subject and in the pose. The human shape representation of the human form incorporates both

the articulated deformation (skeleton articulation) and the non-rigid one. The results obtained by *Anguelov et al.* [1] are very interesting and it formed a basis for this work. For this reason it seems only right to briefly illustrate how this method works. The SCAPE method is composed of two main parts. In the first one the goal is to learn a pose deformation model which derives the non-rigid deformed surface as a function of the articulated skeleton pose. In the second one the goal is to learn a separated model which describes the human body shape variation. These two models can thus be combined to produce 3D models with realistic muscular deformation for different people in different poses when these people and poses do not belong to the training set. All the information about the shape come from a set of scans obtained through a full body scanner. From these information two set of data are produced: a set of poses and a set of human body forms. Then a mesh is selected as template, while all the others will be considered of instance. The template will be used as a reference for all the other scans. The next step aims to find all the correspondences between the mesh template and the instance meshes. This goal is reached through the positioning of markers on the surfaces. Once these markers are set, they can be associated with each other. Finally, a skeleton is built for the template mesh. The algorithm used for this task exploits the fact that vertices that belongs to the same limb of the skeleton are spatially contiguous and show similar motion between the scans. The object is thus decomposed in rigid parts. These parts are then detected in the other scans too, along with the link that lies between them.

It is now possible to learn the pose deformation model and thus to model the deformation which aligns the template mesh with all the other

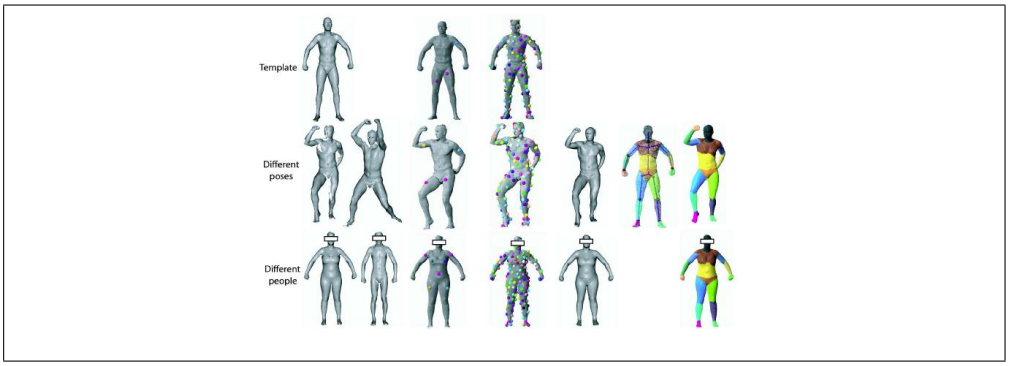

Figure 5.1: Some pics taken from the TOSCA dataset.

meshes in a triangle-wise way. The pose deformation is composed of a non-rigid transformation and a rigid one. Those deformations are applied to the triangle translated to the global origin.

$$\hat{v}_{k,j} = x_{k,j} - x_{k,1}, \quad j = 2, 3 \tag{5.1}$$

Firstly, a linear transformation matrix Q_{ik} is applied to the triangle. This matrix corresponds to a non-rigid deformation induced by the pose and it is specific for each triangle p_k and for each pose Y_i. The deformed polygon is then rotated by R_{il}, the rotation of its rigid part in the articulated skeleton. The same rotation is applied to each triangle which belongs to the same rigid part.

$$v^i_{k,j} = R^i_{l[k]} Q^i_k \hat{v}_{k,j}, \quad j = 2, 3 \tag{5.2}$$

This method makes a prediction for the edges of p_k as $R_k Q_k v_{k,j}$. Unfortunately, this prediction is rarely consistent. Hence, to build a coherent mesh the author proposed to solve for the $y_1, ..., y_m$ points location which minimizes the overall least squares error

$$\underset{y_1,\ldots,y_m}{\operatorname{argmin}} \sum_k \sum_{j=2,3} \|R^i_{l[k]} Q^i_k \hat{v}_{j,k} - (y_{j,k} - y_{1,k})\|^2 \tag{5.3}$$

The goal is to predict the deformations starting from the articulated pose, which is represented as the set of rotations with respect to the joints. The rotation of the joints are represented as their twist coordinate. A twist coordinate is a 3 dimensional vector whose direction represents the rotation axis while the module represents the angle. The non-rigid deformation matrix Q is predicted from the fulcrums of the two nearest joints. Each joint rotation is expressed through three parameters, thus the union of the two joints produces a total of 6 parameters. We finally add a constant parameter used as bias obtaining a 7×1 vector $a_{k,lm}$ that will be associated to each element of the matrix Q, obtaining

$$q^i_{k,lm} = a^T_{k,lm} \cdot \begin{bmatrix} \triangle r^i_{l[k]} \\ 1 \end{bmatrix} \quad l, m = 1, 2, 3 \tag{5.4}$$

Accordingly, the next step is to learn these parameters. The task would be easy if the matrices Q and the rotations R_i were known for each instance of Y_i, solving the system

$$\underset{a_{k,lm}}{\operatorname{argmin}} \sum_i \left(\begin{bmatrix} \triangle r^i & 1 \end{bmatrix} a_{k,lm} - q^i_{k,lm} \right)^2 \tag{5.5}$$

The trick is to estimate the matrices adapting them to the transformations seen in data. This is an underconstrained problem so it is necessary to introduce a smoothing constraint which prefers similar deformation on polygons that belong to the same rigid part. Thus it is possible to resolve for the correct set of linear transformations for each mesh with the equation

$$\underset{\{Q_1^i,\ldots,Q_P^i\}}{\operatorname{argmin}} \sum_k \sum_{j=2,3} \|R_k^i Q_k^i \hat{v}_{k,j} - v_{k,j}^i\|^2 + w_s \sum_{k_1,k_2 \text{ adj}} I(l_{k_1} = l_{k_2}) \cdot \|Q_{k_1}^i - Q_{k_2}^i\|^2 \tag{5.6}$$

The variations of the shape of the body are modelled independently from the variation of the pose, introducing a new set of linear transformation matrices S_k^i, one for each instance i and for each triangle k. It is assumed that the triangle p_k observed in the instance mesh i is obtained first by applying the pose deformation matrix Q, second by the shape deformation matrix S and third by the rotation R associated to the joint

$$v_{k,j}^i = R_{l[k]}^i S_k^i Q_k^i \hat{v}_(k,j) \tag{5.7}$$

To define the body shape deformation space, the S matrices are assumed to arise from a lower dimensional subspace. For each mesh a vector of size $9 \times N$ is created and it will contain the parameters of the matrices S. It is assumed that these vectors are generated from a simple linear subspace, which can be estimated using *PCA* (Principal Component Analysis)

$$S^i = \mathcal{S}_{U,\mu}(\beta^i) = \overline{U\beta^i + \mu} \tag{5.8}$$

where $U\beta i + \mu$ is a reconstruction in vectorial form of the $9 \times N$ parameters of the matrix from the PCA, and $\overline{U\beta^i + \mu}$ is the representation of this vector as a set of matrices. Given the matrices S for each i, k is possible to resolve the system for the PCA parameters U and μ and the mesh specific coefficients β^i. Unfortunately, the matrix is not known and it has to be estimated before. The same trick used in 5.6

$$\underset{S^i}{\operatorname{argmin}} \sum_k \sum_{j=2,3} \|R_k^i S_k^i Q_k^i \hat{v}_{k,j} - v_{k,j}^i\|^2 + w_s \sum_{k_1,k_2 \text{ adj}} \|S_{k_1}^i - S_{k_2}^i\|^2 \quad (5.9)$$

Once the pose deformation model and shape deformation model are learned, they can be used for a variety of applications, like the shape completion (*i.e.* hole filling), partial shape completion (the input is an incomplete scan and a set of markers on it) or to produce animations from marker motion capture sequences. For more details on the applications of this method see the original paper.

5.2 TOSCA

The second dataset selected for this work is the TOSCA dataset (see [6]). This dataset includes high resolution three-dimensional non-rigid shapes in a variety of poses for non-rigid shape similarity and correspondence experiments. The database contains a total of 73 objects, including 11 cats, 9 dogs, 8 horses, 6 centaurs, 12 female figures (Victoria dataset), and two different male figures, containing 7 (David dataset) and 20 poses (Michael dataset).

The models included in this datasets are very detailed (dense), with about 50000 vertices for each model. Objects within the same class have the same triangulation and an equal number of vertices numbered in a compatible way. This can be used as a per-vertex ground truth correspondence in correspondence experiments or in a project like the one we propose.

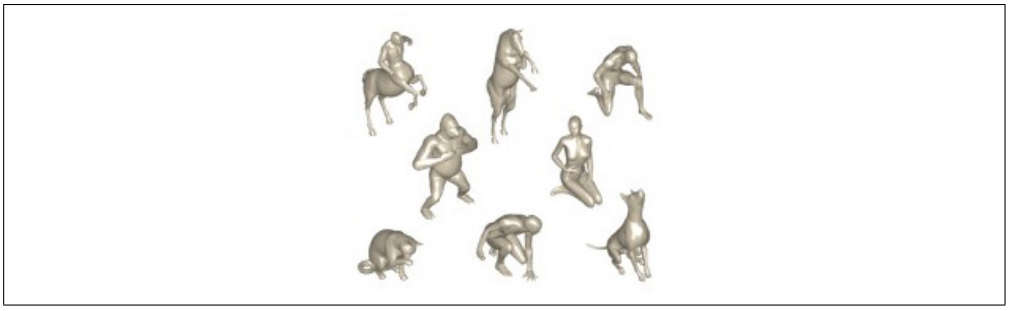

Figure 5.2: Some pics taken from the TOSCA dataset.

Chapter 6

Experimental results

After the definition of our statistical model and after a brief introduction about how we use it, we are now able to present the obtained experimental results. We used 2 different datasets (see 5) with a total of 8 different shapes, which are:

- SCAPE: the scape dataset contains 71 meshes representing the same man in different poses. It is therefore a full human body of a male dataset.

- Cat: the cat dataset is part of the aforementioned TOSCA dataset (which includes different "class" of shape), like the other which will follow. This dataset is composed of 11 meshes of a cat in different poses.

- Centaur: the centaur dataset contains 6 different meshes of the Greek mythological creature. It has the upper body of a man, and the lower body of a horse.

- David: this is the second dataset representing a human body. Part of the TOSCA dataset, it contains 7 meshes of a man in different poses.

- Dog: composed of 9 different meshes, this datasets contains, as the name suggests, a dog in different poses.

- Horse: the horse datasets contains 7 meshes of a horse engaged in different actions.

- Michael: the Michael dataset is the third collection of meshes which represent a man in different poses. It is composed of 20 meshes.

- Victoria: the last collection includes 12 meshes of a woman in different poses.

Hence, the datasets used to obtain our experimental results are composed of 144 meshes divided in 8 different classes. Remember that a shape is just a surface representing a particular object, like a man or a cat, and it is independent from the pose, the scale and the position of the subject. In order to test the resilience to the noise of our model and to study how the results change when in input is far from perfect, we have forged other datasets starting from the aforementioned ones, in which we added random noise to each vertex of a mesh. We used 3 different level of noise, obtaining as results 4 sets of datasets which we will call *perfect*, *noised1*, *noised2* and *noised3*, where the index indicates a growing level of noise added in the relative dataset. An example of the meshes used for our tests and the same meshes with synthetic noise added can be seen in figure 6.1.

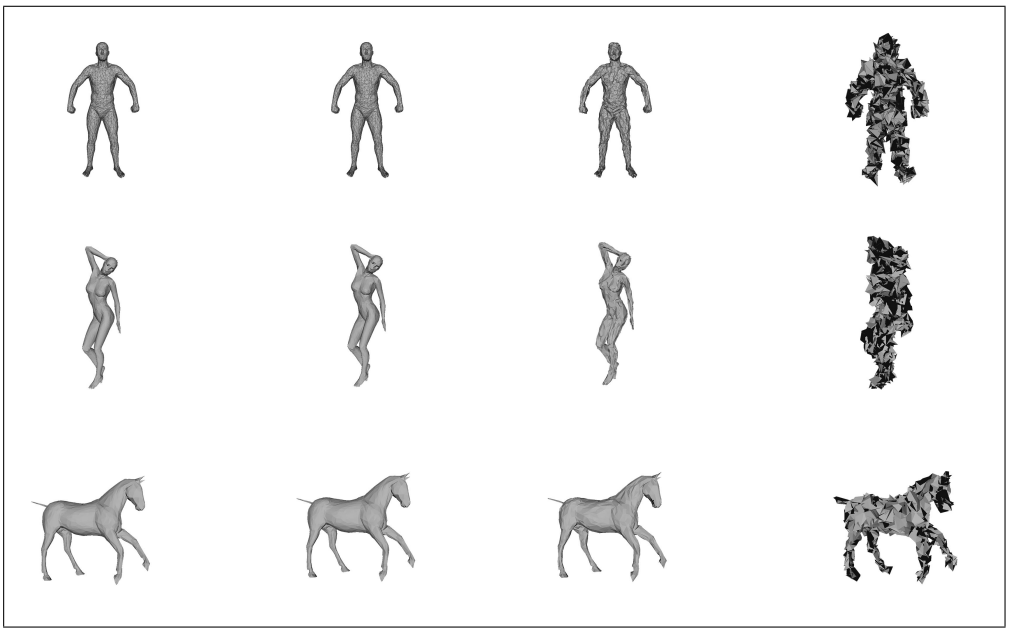

Figure 6.1: Examples of meshes which belongs to different datasets with different levels of noise. From top to bottom the meshes belong respectively to the datasets SCAPE, Victoria and Horse with growing noise level (from left to right).

In order to test our model, we have firstly computed the shape descriptor (*i.e.* the manifold centroid) of each dataset. The output of this step is a set of 8 eigenvectors matrices and 8 eigenvalues diagonal matrices. To ensure that the approach we formulate to compute the manifold centroid was correct, we used a principal coordinates analysis algorithm (also known as multidimensional scaling) to plot the distances between each mesh (manifold centroid included). In figure 6.2 we present what we obtain. The empty red circle represents the manifold centroid, while the blue crosses correspond to the meshes which belong to the dataset.

We proceed then with the estimation of the distributions parameters (see 3.5), which allow to compute the probability density of a certain mesh with respect to a given dataset descriptor. Once the parameters are computed, we are able to build the so called *confusion matrix*. A confusion matrix is a widely used table layout (in particular in artificial intelligence) that allows visualization of the performance of a classification algorithm. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. In our case, the confusion matrix is built starting from the probability density of each mesh with respect to each class (*i.e.* dataset) we used. A mesh is thus assigned to the class towards which the density is higher. The results obtained through our method can be seen in table 6.1.

An alternative way to show the separation between the datasets is through the geodesic distances matrix. After building a matrix whose rows are the meshes and whose columns are the geodesic distances with respect to a certain shape (class), we can plot them as an image. A lower

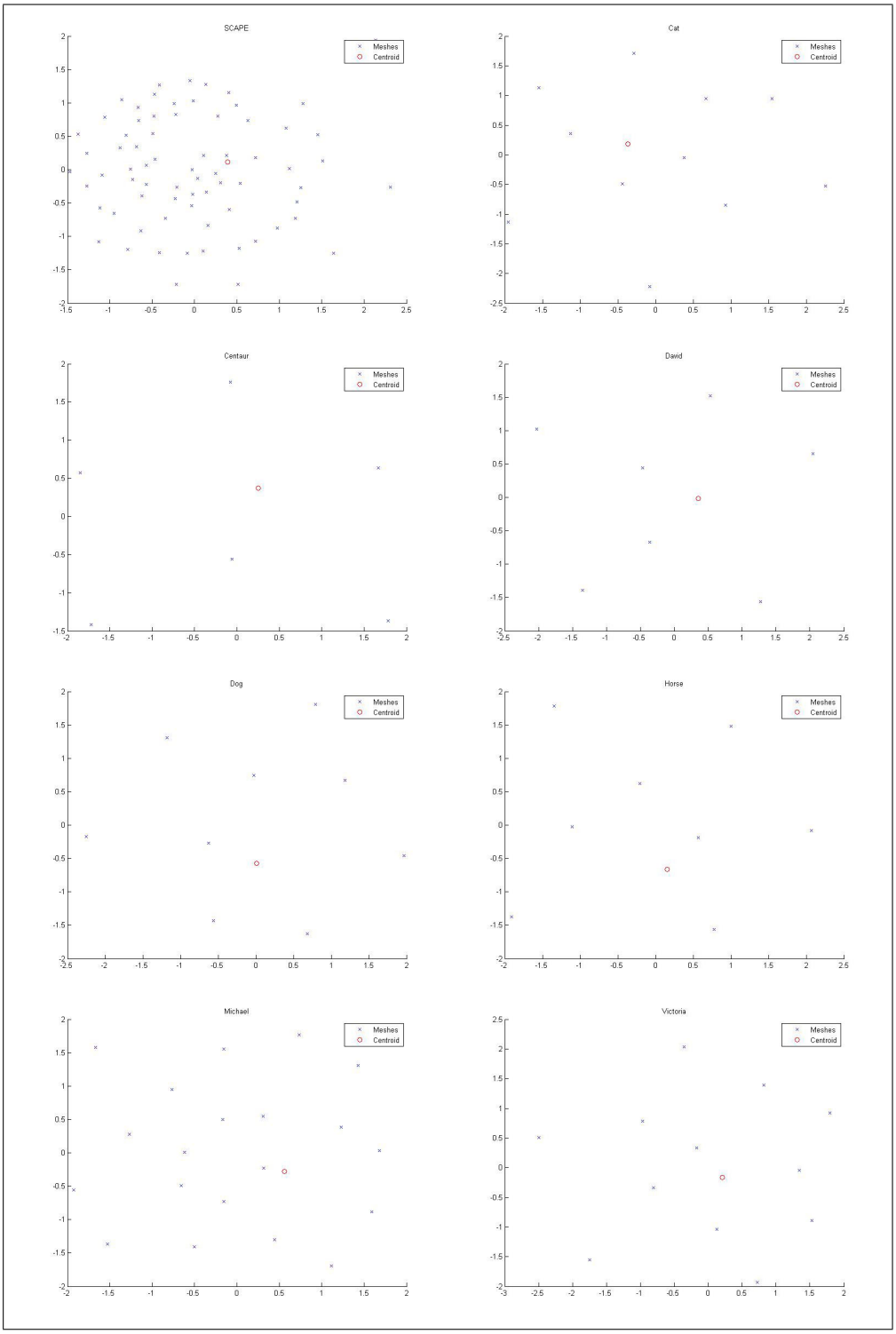

Figure 6.2: Plots resulting from the application of the multidimensional scaling to the distances matrices between meshes which belong to the same dataset.

Shapes	SCAPE	Cat	Centaur	David	Dog	Horse	Michael	Victoria
SCAPE	71	0	0	0	0	0	0	0
Cat	0	11	0	0	0	0	0	0
Centaur	0	0	6	0	0	0	0	0
David	0	0	0	7	0	0	0	0
Dog	0	0	0	0	9	0	0	0
Horse	0	0	0	0	0	8	0	0
Michael	0	0	0	0	0	0	20	0
Victoria	0	0	0	0	0	0	0	12

Table 6.1: Resulting confusion matrix

distance will result in a darker color. In figure 6.3 is possible to see the geodesic distances computed on each datasets with different level of noise. The darker diagonal means that the geodesic distances between meshes that belong to the corresponding dataset are lower than the distances between meshes which are not.

The geodesic distance between a mesh and the shape descriptor of a certain shape can be used as a metric which shows how far or near (or equivalently how similar) two shapes are. In figure 6.4 we show the average distances between the meshes which belong to a certain dataset with respect to a different dataset.

The last experiment which we performed uses the aforementioned geodesic distances matrix as the input of the principal component analysis procedure. The PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This trans-

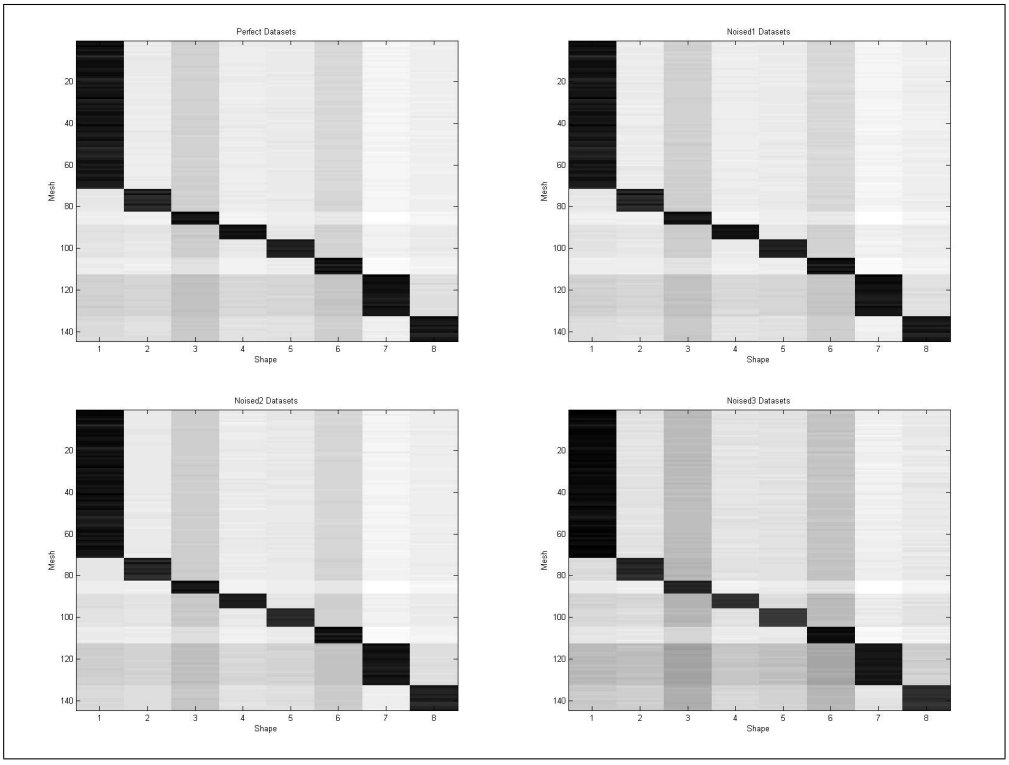

Figure 6.3: Plot of the distances matrix between each mesh of each dataset with respect to a certain manifold centroid. One cell of the image represents the geodesic distance between the i-th mesh and the j-shape (class).

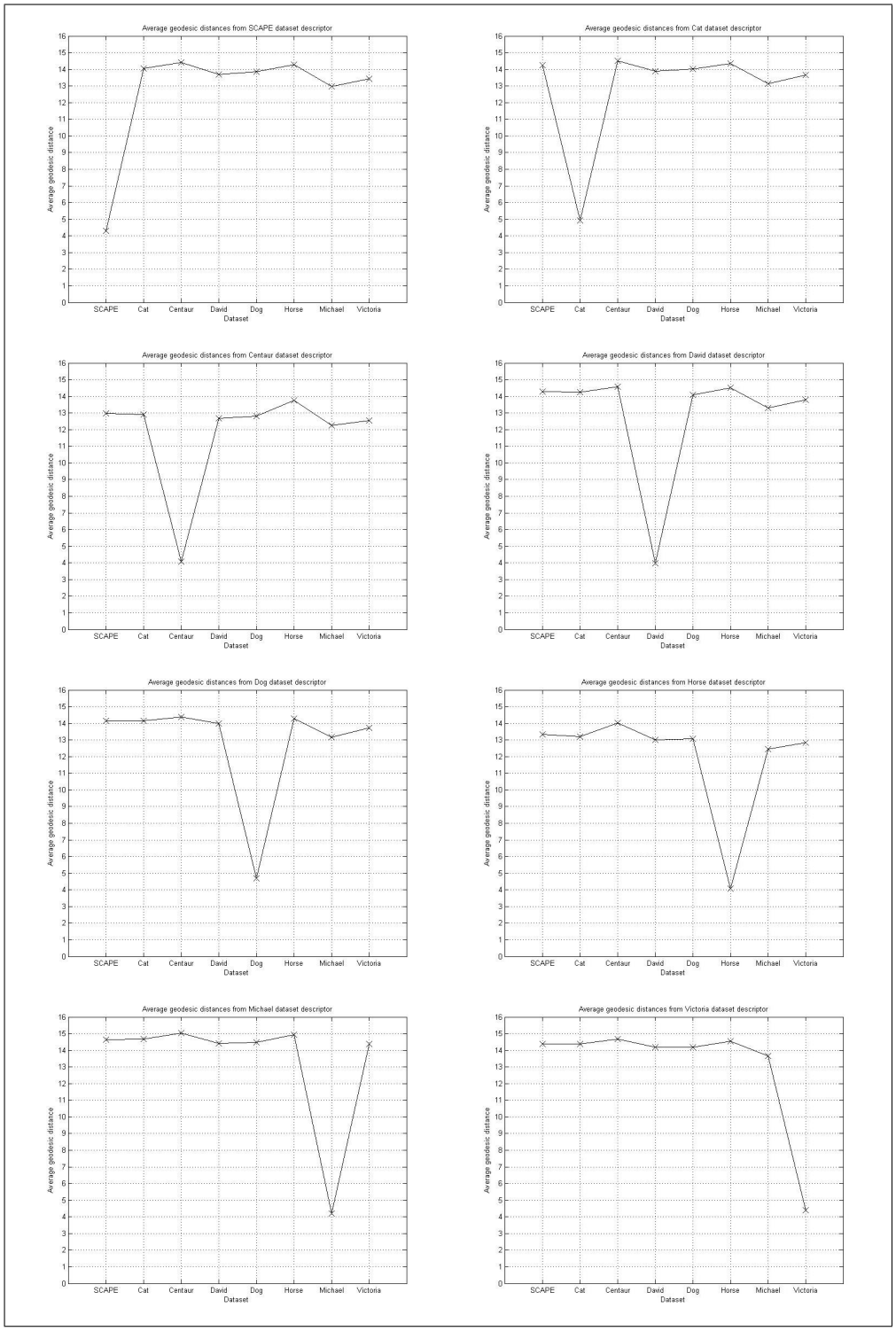

Figure 6.4: These images show the average geodesic distance of a dataset with respect to a shape descriptor.

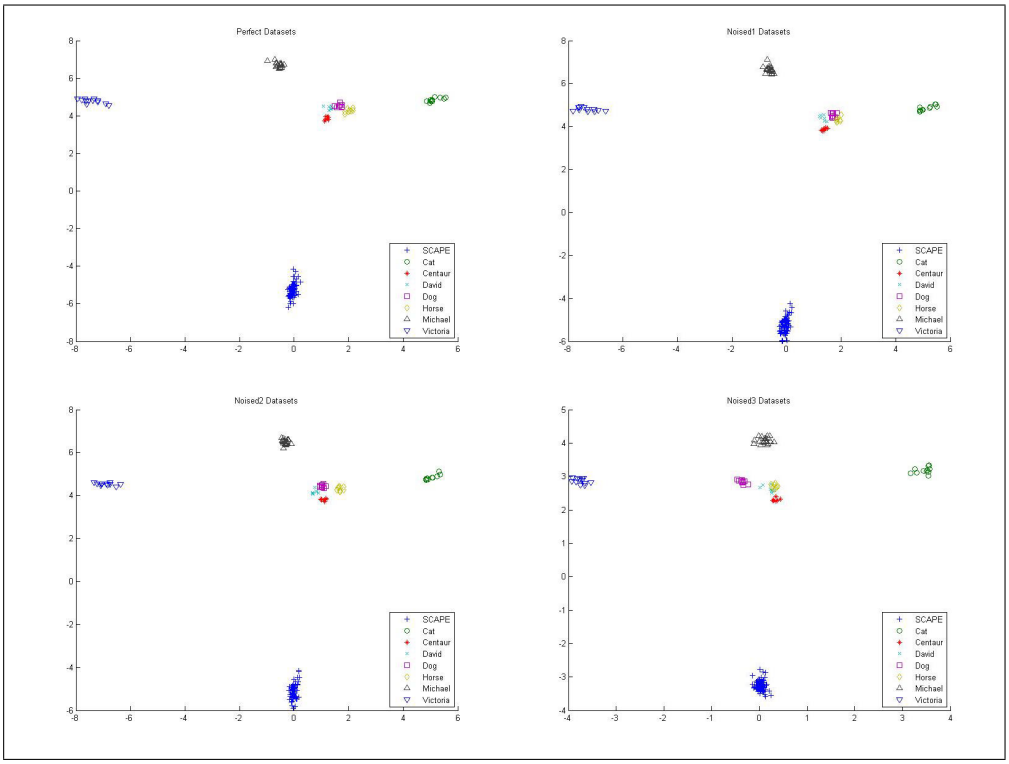

Figure 6.5: PCA applications to the distances matrix of the *perfect*, *noised1*, *noised2* and *noised3* datasets. Each different symbols on the plot correspond to a different dataset.

formation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to the preceding components. The application of this procedure allows us to plot the separation between the classes given the distances between them. In figure 6.5 we show the obtained results.

The SCAPE dataset has proved to be very useful due to the number of mesh in it. In fact, in order to prove the robustness of our algo-

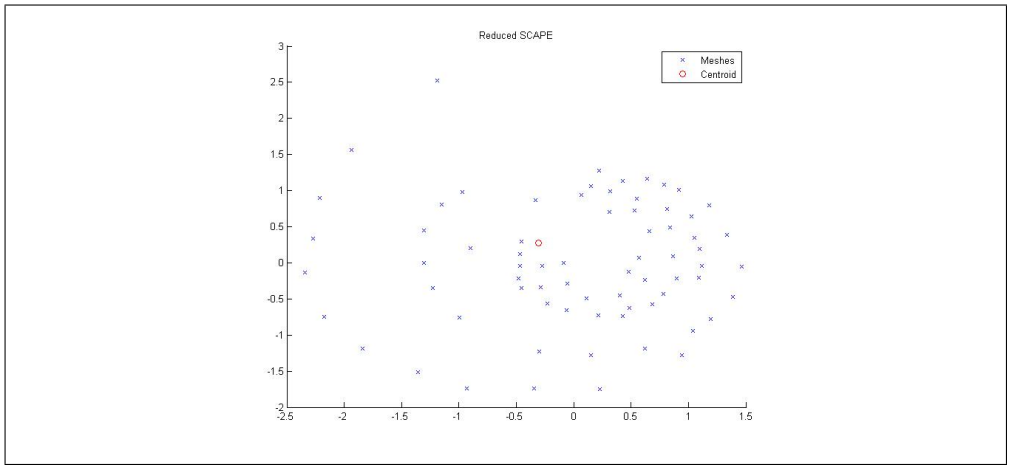

Figure 6.6: Plot resulting from the application of the multidimensional scaling to the distances matrix computed between the meshes, which belong to the SCAPE dataset whose descriptor is computed using only part of the whole set.

rithm, we have to try it with a set of meshes which represents the same shape, but they have not been used in the learning phase. Thanks to its 71 different meshes, we were able to partition the dataset in 2 subsets with almost the same cardinality. The first has been used during the learning phase to compute the manifold centroid of the *reduced* SCAPE dataset, while the second has just been used to compute the probability density (and the geodesic distances) between the meshes and the shape descriptor. Figure 6.6 shows the multidimensional scaling application to the distances matrix. Furthermore, in figure 6.7 we show the difference between the geodesic distances intra-dataset computed using the whole SCAPE dataset and the reduced one.

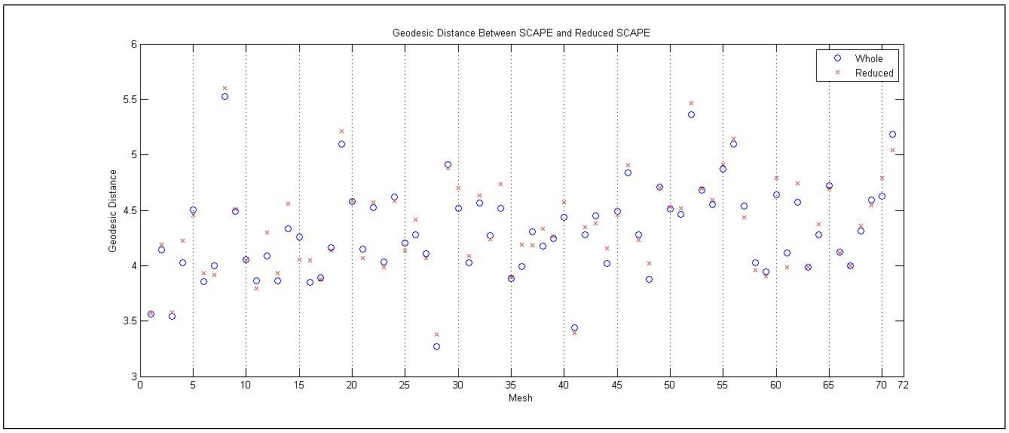

Figure 6.7: Differences between the geodesic distances with respect to the SCAPE manifold centroid computed with the whole dataset and a subset of it.

Chapter 7

Conclusions

In this dissertation we proposed a novel technique to compute a shape descriptor of non-rigid shape, which can be used in shape retrieval problems. Furthermore, we defined a statistical model that can be used to find how much the underlying surface of a mesh is similar to another one. Our method uses an iterative optimization process to compute the shape descriptor of a given dataset. The descriptor can so be used to compute the geodesic distance between a shape (a dataset) and a mesh which does not belong to it. This computation is obtained through another iterative optimization process which involves the use of a bipartite matching problem solver, in order to minimize the issues arising from the lack of correspondences between the meshes. The defined descriptor has most of the desirable properties, like isometry invariance, similarity invariance, efficiency and compression. As shown in the experiments, the model we defined allows us to separate correctly different shapes, assigning the associated mesh to the right class. Furthermore, we shown that the use of a mesh in the learning phase and the consequent computation of the

geodesic distance yields the same results as the computation of the same distance with a mesh whose underlying surface is the same, but it does not belong to the training set. The application of our method have shown that similar shapes have lower geodesic distances (and therefore higher probability). This is an interesting results since it allows a proper use of our method in tasks, like similar objects retrieval in shapes database (where our descriptor could be an efficient index), objects identification for copyright purpose or automatic shape retrieval problem (*i.e.* classification problems). The experiments shown that the model is resistant to noise. Even after adding a high amount of noise, we were able to separate correctly the meshes which belong to different classes. But since the Laplace-Beltrami operator is a second order operator and therefore pretty sensible to noise, we could not add as much noise as we want to test the resilience of the model. Indeed, adding a higher amount of noise prevents the computation of the Laplacian matrix of a mesh. Finally, the results have shown that in order to compute a good shape descriptor which characterizes in a good way the underlying surface of the meshes of a dataset we need a good amount of meshes representing a shape in different poses. The ideal number of meshes which has been identified through experiments is ≈ 10. In fact, the datasets with a lower number of meshes, even if they are still perfectly separable from the other classes, result *nearer* to other datasets. For instance, it is easy to see that the PCA applications to the distances matrix tend to separate in a better way shapes whit a higher number of representatives (like SCAPE, Cat, Michael and Victoria) while sticking together the datasets with fewer meshes (David, Centaur, Dog and Horse).

Bibliography

[1] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. *ACM Trans. Graph*, 24:408–416, 2005.

[2] Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 441–448. MIT Press, Cambridge, MA, 2005.

[3] Mikhail Belkin, Jian Sun, and Yusu Wang. Discrete laplace operator on meshed surfaces. pages 278–287, 2008.

[4] P. Bérard, G. Besson, and S. Gallot. Embedding riemannian manifolds by their heat kernel. *Geometric and Functional Analysis GAFA*, 4:373–398, 1994.

[5] A. M. Bronstein. Spectral descriptors of deformable shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2011.

[6] Alexander Bronstein. *Numerical geometry of non-rigid shapes*. 2007.

[7] Michael M. Bronstein and Alexander M. Bronstein. Analysis of diffusion geometry methods for shape recognition.

[8] M.M. Bronstein and A.M. Bronstein. Shape recognition with spectral distances. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):1065 –1071, may 2011.

[9] Manfredo Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[10] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. pages 7426–7431, 2005.

[11] Bruno Colbois. Laplacian on riemannian manifolds, 2010.

[12] Martin Vito Cruz. The spectrum of the laplacian in riemannian geometry, 2003.

[13] Bjorn Ian Dundas. Differential topology, 2009.

[14] Alexander Grigor'yan. *Heat Kernel and Analysis on Manifolds*. American Mathematical Society, 2009.

[15] Mark Harmer. Laplace-beltrami operator, 2007.

[16] Noel J. Hicks. *Differential Geometry*. Van Nostrand Reinhold Company, 1965.

[17] Hugues Hoppe, Tony DeRose, Tom Duchamp, John Mcdonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *COMPUTER GRAPHICS (SIGGRAPH '92 PROCEEDINGS)*, pages 71–78, 1992.

[18] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[19] Jurgen Jost. *Riemannian Geometry and Geometric Analysis*. Springer, 2011.

[20] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[21] Alexander M. Bronstein Michael M. Bronstein. Analysis of diffusion geometry methods for shape recognition. 2010.

[22] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè, and Michela Spagnuolo. Discrete Laplace–Beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390, June 2009.

[23] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-spectra as fingerprints for shape matching. pages 101–106, 2005.

[24] Roberts Strichartz. Analysis of the laplacian on the complete riemannian manifold, 1983.

[25] Max Wardetzky. Convergence of the cotangent formula: An overview.

[26] Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun. Discrete laplace operators: No free lunch, 2007.

[27] D. White and R.C. Wilson. Spectral generative models for graphs. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 35 –42, sept. 2007.

[28] Guoliang Xu. Convergence of discrete laplace-beltrami operators over surfaces. *Comput. Math. Appl*, 48:347–360, 2004.

[29] A. Young. Eigenvalues and heat kernel. 2003.

[30] Wei Zeng, Ren Guo, Feng Luo, and Xianfeng Gu. Discrete heat kernel determines discrete riemannian metric. *Graph. Models*, 74(4):121–129, July 2012.