Intrinsic mesh matching for near-isometric deformations using double-order affinities

by

Ioannis Koutsoumpas

A thesis submitted in partial satisfaction of the
requirements for the degree of
Master of Research

in

Game and Media Technology

in the

Graduate Division

of the

University of Utrecht

Committee in charge:

Professor Michael Wand
Professor Marc van Kreveld

Winter 2015

Intrinsic mesh matching for near-isometric deformations using double-order affinities

Abstract

Intrinsic mesh matching for near-isometric deformations using double-order affinities

by

Ioannis Koutsoumpas

Master of Research in Game and Media Technology

University of Utrecht

Shape matching is among the most basic research fields in digital geometry processing, with applications ranging from industrial design to three-dimensional medical image analysis. Our focus is restricted to triangle meshes undergoing deformations that can be described by intrinsic isometries, that is, near-isometric changes. In this thesis, we propose a shape matching algorithm comprised by a feature detection and feature matching phase. Specifically, a shape descriptor is introduced, called the vicinity area descriptor, based on the surface area around each vertex bounded by an isoring for a given geodesic radius. We improve the distinctiveness of the local signature by extending it from a scalar to a vector descriptor referring to arbitrary number of areas defined by inner isorings. The most descriptive points are then extracted using non-maximum suppression. By also considering the preservation of geodesic distances among the corresponding pairs of features, we compute a double-order affinity matrix. This combinatorial affinity matrix encodes the pointwise and pairwise relations of features regarding the two meshes. This matrix is then fed to the spectral matching algorithm, a graph matching method, in order to establish correspondences between the two surfaces. Experiments include benchmarks under various conditions regarding internal variables and state-of-the-art methods comparisons. It is showed that the proposed framework is robust over near-isometric deformations and keeps well against modern algorithms.

To the people that follow their passion

Contents

List of Figures

List of Tables

Acknowledgments

I want to express my deepest gratitude to my advisor, Professor Michael Wand, for his exceptional guidance and providing me with an excellent atmosphere for doing research. Without his advice, this thesis would not have come into being.

I have also to thank the friends and colleagues that inspired me and shared their passion for computer science.

Finally, I would like to thank my family for always encouraging and caring for me.

Chapter 1

Introduction

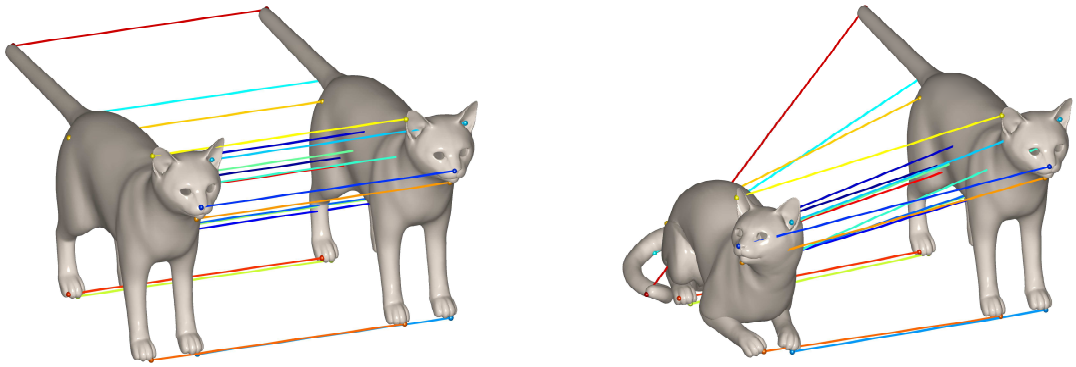
Over the last decade, three-dimensional shape processing became popular in the research areas of geometric processing, medical imaging and three-dimensional modeling. One of its keystone tasks is the establishment of correspondences among two shapes [39][2][21][23][48]. Correspondence detection is of utmost importance for surface completion, statistical shape modeling, deformable surface tracking, symmetry analysis, shape matching and shape calculus [4].

We restrict our attention to automatic non-rigid shape matching, an interesting yet complex task. The majority of shape matching applications involve input meshes representing objects in a variety of poses and deformations. For instance, input meshes could present deformations of cloth, faces, brains surfaces or other anatomical organs. Although a simple parametrization exists in rigid matching for all possible deformations (i.e. translation, rotation and reflection), non-rigid mesh deformations are high dimensional and arbitrarily complex (see Fig. 1.1).

Fortunately, in practice, non-rigid deformations tend to follow certain rules (e.g. a piece of cloth that can be stretched up to a point but not teared), thus implying various solution approaches. In this work, we perform non-rigid shape matching by taking advantage of certain surface properties that are invariant under *isometric deformations*. In particular, isometric deformations imply that, all points of the surface prior and after the deformation preserve their intrinsic geodesic distances up to minor error (near-isometric deformations). Generally, geodesic distance between two points on a surface, can be described as the length of the shortest path connecting them. For a detailed description, refer to Section 2.1. Specifically, intrinsic isometry can be formally described as:

Definition 1 Consider two shapes M and N represented as compact Riemannian manifolds without boundary. A surjective map $T : M \to N$ is called an intrinsic isometry such that, for all pairs of points $(x, y) \in M$ it holds that $d_g^M(x, y) = d_g^N(T(x), T(y))$, where $d_g^M(.,.), d_g^N(.,.)$ are the geodesic distances defined on M and N in respect.

Worth noting that, in the rest of the thesis, three-dimensional shapes represented by compact, connected Riemannian manifolds are approximated by triangular meshes.



(a) Two shapes related by a rigid transformation (translation in 3D).

(b) Two shapes related by a non-rigid transformation (near-isometric deformation).

Figure 1.1: Non-rigid shape deformations are highly dimensional.

In our framework, we employ two types of surface properties: pointwise surface descriptors and distances among pairs of points on the surface. In addition, we employ a graph matching method in order to incorporate tolerance for both descriptors and finally compute the minimal dissimilarity correspondence.

1.1 Motivation

First, our key motivation was the proposal of a local surface shape descriptor that is isometry invariant yet is straightforward and relatively simple to implement. Second, despite the variety of surface descriptors, no local region signatures are proposed based on the vicinity surface area. Third, most shape matching algorithms sample points on the mesh in order to reduce the complexity of the problem with risk of ignoring important information. We present a method that computes feature points by initially analyzing specific properties on the entire shape without increasing the dimensionality of the problem.

1.2 Method

At first, we build the *distance map*, an array containing all geodesic distances for every pair of vertices on the mesh using the *fast marching* method [28]. Distance map is thoroughly used for both pointwise and pairwise descriptors computations as well as during the matching process.

We define the pointwise surface descriptor as the local signature containing the area magnitude bounded by the isocurve (or isoring) centered at a vertex given a geodesic radius

(also referred as base radius). Isocurve can be conceived as a circle of a radius defined over a geodesic metric on a surface. The isoring radius is alternatively called *maximum propagation*, since it indicates the distance limits of propagation of the distance function (fast marching method in our case). Scalar surface descriptor is then extended to a vector descriptor such that arbitrary inner isorings can be defined and exponentially distributed (w.r.t. largest geodesic radius), yielding multiple area values. Thus, surface properties are captured in more detail. Having calculated the area descriptor for all vertices, we extract the most distinctive points. This is done by employing *non-maximum suppression*, a local maximum search scheme, for the given base radius. These characteristic points represent the feature points on the current mesh. In addition, we denote the pairwise descriptor as the geodesic distance between two detected features. Feature point detection is applied on both source and target meshes producing two lists including the feature vertices along with their respective score vectors.

By considering the aforementioned feature lists, the second phase starts with the composition of the *combinatorial affinity matrix*. This array combines the comparison results of pointwise and pairwise descriptors along with the corresponding given standard deviations. Throughout this thesis, descriptor relations are denoted as (1^{st} and (2^{nd}-order affinities in respect. A scheme using both these affinities is characterized as a *double-order affinity method*. The combinatorial affinity matrix is computed in a manner suitable for processing by the *spectral matching* algorithm [29], a graph matching scheme. By employing spectral matching, we take advantage of the spectral properties (eigenvalues and eigenvectors) of the weighted adjacency matrix and compute the correspondence vector. This binary vector indicates the valid correspondences between the features of the two shapes thus denoting the optimal matching. Finally, matching error is calculated as the deviation of the resulted matching from the ground truth.

Experiments prove the robustness of the proposed matching framework regarding near-isometric deformations and provide comparisons to state-of-the-art algorithms. We analyze the behaviour of the matching scheme under various conditions (e.g. different base radii, number of isorings per descriptor, single order versus double-order affinities) and compare the results for different distance functions. In addition, spectral matching results are investigated for both Greedy and Hungarian methods (internal spectral matching functions). Furthermore, comparison between *Blended Intrinsic Maps* and *Möbius* Voting [30] schemes indicate the efficiency of the proposed method.

The algorithms of *Planned Landmark Sampling* [54] and *Geodesic Fans* [64] inspired the proposal of of the area descriptor on the base of iteratively applying a sampling structure (similar to isocurves and spokes respectively) in order to gain the maximum possible information. Furthermore, we gather local topology information from the vicinity of a point, similarly to [54][64][19]. In contrast to geodesic fans which use a finite number of spokes to gather curvature information, we apply our descriptor to the entire vicinity. Thus, surface information acquirement is independent of sampling density. Regarding the extraction of feature points, we avoid sampling techniques (used in [55]) for the same reason. That is, each feature point is a result of a process involving all vertices of the mesh.

1.3 Contribution

First, vicinity area descriptor is a novel signature that efficiently captures geometry characteristics among deformed meshes and proves robust against near-isometric changes. Specifically:

- It is scalable, thus describing neighborhoods of various sizes. To wit, it allows for control over how coarsely information is gained and expresses different types of shape properties.
- Scale is easily controlled. Vicinities are indicated by the center and geodesic radius of the isoring.
- It corresponds to curvature perceptivity. The proposed local signature denotes the rate of non-planarity of the surrounding region.

Second, the presented descriptor extends to a vector of arbitrary dimensions through the use of multiple isorings. In particular:

- It enhances the accuracy of the results.
- Adapts on the triangulation complexity.
- Allows for control of sensitivity regarding noise and differences in surface tessellation.

Third, the control over the number and type of feature points in combination with the graph matching incorporation, balances the cost of correspondence guesses with the proper exploitation of acquired information.

1.4 Structure

In Chapter 2 we provide an overview on previous research concerning discrete geodesics and three-dimensional shape matching. Background knowledge regarding the methods employed throughout the shape matching framework are described in Chapter 3. The main method consists of two sections: Chapter 4 presents the local descriptors and the methods behind feature extraction. The phase of matching process is analyzed in Chapter 5. In addition, Section 6 describes the experiments and analyzes the results. In Epilogue (see Section 7), we draw conclusions about our algorithm and propose future improvements. Implementation notes are presented in Section 8.

Chapter 2

Related Literature

A survey of the literature on the fields relevant to our research topic follows.

2.1 Discrete Geodesics on Triangular Meshes

We employ discrete geodesics during the computation of first and second-order affinities as well as spectral matching. Although Euclidean shortest path computation is NP-hard [6], finding the geodesic shortest path may be a problem of polynomial time. The majority of geodesic distance computation algorithms use front propagation, an alternation of Dijkstra's algorithm [17]. In their work, Mitchell *et al.* [38] presented the *continuous Dijkstra* method. They simulate the continuous propagation of a wavefront of points equidistant from the source across the surface, by updating the wavefront at discrete events. Given a source vertex, this data structure computes the actual shortest path to any point on the mesh in time $O(k + \log m)$, where k is the number of faces crossed by the path and m is the number of surface edges. This algorithm requires $O(m^2)$ space and runs in $O(m^2 \log m)$ time. The algorithm of Kapoor [26] adopts wave front propagation approach and by efficiently treating wavefront arc-edge crossing, achieves $O(n \log^2 n)$, where n is the number of mesh vertices. On the other hand, the method of Chen and Han [8] does not track the wave front propagation. That is, they employ a data structure based on surface unfolding, thus improving the time complexity to $O(n^2)$ with $O(n)$ space. Kimmel and Sethian [28] employ *fast marching* method [50] in order to define a distance function from a source vertex to the entire surface. They then integrate back a differential equation to extract the geodesic path. The entire process has time complexity $O(n \log n)$.

The last method presented by Kimmel and Sethian is adopted in our framework. Although it defines an approximation algorithm (in section 4.1 we show how we improve accuracy), it is the fastest, something that proves to be the most efficient scheme for our method.

2.2 Shape Matching

Local Signature

In their work, Shum *et al.* [51] propose a local measure that employs the L_p distance among local curvature functions. These functions are mapped to a semi-regular triangulation of the unit sphere. However, this algorithm is limited to consider only closed surfaces which are topologically spherical. Several shape segmentation algorithms use isosurfaces and extreme curvatures [62], the sign of the curvature [35][63] or watersheds of a curvature function [33] [32] [47]. Watershed methods are sensitive to the user-specified watershed depth threshold and noise. Surface segmentation approaches still do not provide detailed information about small dissimilarities among local regions. To wit, changes on the mesh (e.g. near-isometric deformations) are likely to have strong influence on the segmentation form.

Several methods proposed for the computation of local features. Shape contexts [40] use a point on the object in order to represent its shape. Specifically, they employ a two-dimensional histogram including the relative coordinates of other points sampled from the mesh. In their method, Planitz *et al.* [45] define a feature signature by considering a local region around specific points. However, the criteria of angles and distances among normals in a local support region prove to be sensitive to point distributions. Worth mentioning that, the aforementioned schemes are sufficient for shape discrimination in the context of shape retrieval but in general, offer limited shape matching among similar meshes.

Zhang *et al.* [65] and Tung and Matsuyama [59] define sparsely sampled landmarks using the extremal points of a geodesic function. Nevertheless, in general situations, these landmarks are not capable of describing a unique diffeomorphism. Lipman and Funkhouser present a matching scheme based on Möbius voting [30]. They observe that, for genus-zero surfaces, the isometry group is a subgroup of the Möbius group, parametrized by three distinct correspondences. In relation to the shape parametrization of [30], Ovsjanikov *et al.* introduce *Heat-Kernel Signature* [53], a multi-scale point signature based on the heat diffusion process. They prove that a single point is sufficient to define an isometry under certain conditions [44]. Another scheme based on local signatures called *Blended Intrinsic Maps* [27], was proposed by Kim *et al.*. Specifically, they consider weighted combinations of intrinsic maps in order to define a map between the two meshes.

Tevs *et al.* [54] present a sampling-based shape matching that detects optimized landmark points. To wit, they employ an entropy-based planning scheme in order to select important matches and reduce sampling cost. A local descriptor called geodeisc fan is presented in [64]. In particular, a geodesic fan consists of a set of spokes with multiple samples per spoke that capture surface properties. In our framework, we propose a local surface area descriptor in the spirit of the work of Tevs *et al.* [54] and geodesic fans [64]. That is, we consider multiple neighborhoods around a central vertex, similar to [64][54] and expand the isocurves concept [54] to the entire vicinity surface area.

Feature Distinctiveness

Various approaches used in order to answer how characteristic a feature points is. A statistical measurement of the *frequency-inverse document frequency* is employed in [3]. This technique quantifies the importance of a feature point by checking the rate of occurrence in other shapes. On the contrary, our method considers features in a single shape. In their algorithm, Schmid *et al.* [49] as well as in [54], incorporate entropy in order to measure the saliency of feature points. In our framework, we compute candidate feature distinctiveness by employing Non-maximum Suppression (NMS) (an example of its use can be found in [5][31][37][60]) as a local maximum search algorithm consisting of two nested loops. This approach is simple to implement and versatile due to the adaptive search radius.

Feature Matching

Fischler and Bolles [18] introduce a matching algorithm called RANSAC (random sampling consensus). This method samples correspondences selected from two images (a problem similar to shape matching) and estimates the parameters of a geometric consistency model through an iterative scheme. However, the process contains several parameters (some of them are empirically determined) whose values have to be chosen properly in advance. These parameters have to be carefully selected, since they strongly affect the performance and robustness of the algorithm. A variant of RANSAC named PROSAC (progressive sampling consensus) was proposed by Chum and Matas [9]. By smartly sampling the high quality data first and then progressively sampling the rest of the dataset, they achieve better performance (in the worst case, it degenerates to RANSAC). Worth noting that, both these sample consensus methods do not efficiently respond to images with repetitive patterns (similar to repetitive triangulation patterns) because of too many outliers. REINF (reinforcement matching) [15] incorporates global context information into local feature matching. To wit, it extends circular bin techniques by using affine-invariant log-polar elliptical bins. Furthermore, *relaxation method* [52] introduces a probabilistic matching framework which iteratively updates initial probabilities based on a compatibility function.

In order to make the correspondence estimation more robust, we adopt *spectral matching* method introduced by Leordeanu and Hebert [29]. At first, they represent correspondences between two feature sets by a properly constructed graph. They then form a compatibility matrix that encodes point-wise and pairwise affinities among features. Finally, they detect correct assignments by computing the principal eigenvector of the aforementioned matrix and applying specific mapping constraints (in our case, one-to-one correspondences).

Chapter 3

Prerequisites

In this chapter we provide the description of concepts employed in the proposed algorithm as well as in various benchmarks.

3.1 Geodesic

Geodesic curve (also called as *geodesic*) is the generalized concept of straight line for smooth surfaces, that is, the shortest path joining two points over a surface. Equivalently, a geodesic is a locally length-minimizing curve. In general, geodesics in space depend on the Riemannian metric [25] which affects the notion of distance and acceleration. The basic intrinsic metric defined over a surface M is called *geodesic metric*. To wit, it measures the lengths of the shortest paths on M such that

$$d_M(x, x') = \inf_{\gamma \in \Gamma(x,x')} l(\gamma) \tag{3.1}$$

where $\Gamma(x, x')$ is the set of all admissible paths between the points x and x' on M and $l(\gamma)$ the length of a path γ.

In the proposed framework we use geodesics on discrete surfaces in three dimensions in both first and second affinity computations. In the following two paragraphs we provide a brief description of geodesics in continuous as well as discrete space.

Let a smooth surface in $\mathbb{R}^3$ be parametrically defined by $x = x(u, \nu), y = y(u, \nu), z = z(u, \nu)$, where u, ν are parameters from a subset $A \subset \mathbb{R}^2$. The *continuous geodesic* can be found by minimizing the arc length

$$I = \int ds = \int \sqrt{dx^2 + dy^2 + dz^2} \tag{3.2}$$

For details on minimization of Equation 3.2, see [7].

The generalization of geodesic curves to a discrete surface S, are called *discrete geodesics*. Aleksandrov and Zalgaller [1] define *quasi-geodesics* as the limit curves of geodesics on a

family of converging smooth surfaces. In addition, they describe *shortest discrete geodesics* as critical points of the length functional over polyhedral surfaces. Furthermore, Polthier and Schmies [46] introduce *discrete geodesic curvature* based on the definition of *geodesic curvature* (geodesic curvature generalizes the notion of curvature of a plane curve to surfaces [7]). In addition, they propose *straightest geodesics*, as the polygonal curves over S with zero geodesic curvature everywhere.

3.2 Fast Marching Method

The *Fast Marching* algorithm is a numerical method introduced by Sethian [50] and employed by Kimmel *et al.* [11] in order to compute 2D paths. A similar method was also proposed by Tsitsiklis in [58]. In addition, fast marching was applied for path extraction in 3D images [16][10] for medical image analysis. In our framework, we adopt the extension of Kimmel and Sethian [28] that allows to find the geodesic distance among vertices on a manifold. Worth noting that, the farst matching method is similar to Dijkstra algorithm [17] in the sense of front propagation from the source to all vertices in the domain.

We briefly describe the process of geodesic path computation using the fast marching method. Consider a metric $P(g)dg > 0$ on a manifold M. The weighted geodesic distance among $x_S, x_T \in M$ is defined as

$$d(x_T, x_S) \coloneqq \min_{\gamma}\Big(\int_0^1 \|\gamma'(t))\|\, P(\gamma(t))dt\Big) \tag{3.3}$$

where γ is a piecewise regular curve such that $\gamma(0) = x_S$ and $\gamma(1) = x_T$. For $P = 1$, the integral in Eq. 3.3 denotes the length of the curve γ and d is the geodesic distance. In order to efficiently calculate the distance function $U(x) \coloneqq d(x_S, x)$, the following formulation is employed. Let $C_t \coloneqq \{x \setminus U(x) = t\}$ be the level set curve that propagates according to the evolution equation $\frac{dC_t(x)}{dt} = \frac{1}{P(x)}\vec{n_x}$, where $\vec{n_x}$ is the exterior unit vector normal to the curve at x and the function U satisfies the nonlinear *Eikonal* equation

$$\|\nabla U(x)\| = P(x). \tag{3.4}$$

By taking the inverse fraction of function P, the function $F = 1/P > 0$ represents the propagation speed of the front C_t.

In order to calculate the value u of U at a point $x_{i,j}$ on a grid, fast marching algorithm uses the following upwind finite difference scheme:

$$\begin{aligned} \max(u - U(x_{i-1,j}), u - U(x_{i+1,j}), 0)^2 \\ + \max(u - U(x_{i,j-1}), u - U(x_{i,j+1}), 0)^2 = P(x_{i,j})^2. \end{aligned} \tag{3.5}$$

A method to solve the aforementioned second order equation (see Eq. 3.5) is described in [12]. Hence, an optimal ordering of the grid points is computed such that the complexity of the procedure takes $O(N \log(N))$, where N is the number of points.

The algorithm in [28] generalizes fast marching to arbitrary triangulations thus allowing for fast front-propagation on a mesh. Furthermore, they propose a scheme in order to overcome the numerical instabilities arising from triangulations containing obtuse angles.

3.3 Spectral Matching

In their work, Leordeanu *et al.*[29] proposed a spectral technique for acquiring consistent correspondences among two sets of features.

Let *bipartite graph* be a graph consisting of vertices decomposed into two disjoint sets such that, no two graph vertices within the same set are adjacent. Consider the bipartite graph matching problem as the matching comprised by the subset of edges such that, no two edges share an endpoint. The *weighted adjacency matrix* of a graph, stores the weights of a graph whose nodes represent the potential assignments and whose weights are the agreements among pairs of candidate assignments. Spectral matching method steps on the spectral properties of the weighted adjacency matrix of the graph and efficiently solves the correspondence problem by avoiding the rapid increase of complexity due to the combinatorial nature of the process.

Let P and Q be two sets containing n_P and n_Q data features respectively. A *correspondence mapping* is defined as a set C of assignments (i, a), where $i \in P$ and $a \in Q$. Depending on the problem, various mapping constraints can be applied on C. For instance, a data feature in P is allowed to pair with at most one feature from Q (one-to-one correspondence) or with many features from Q (one-to-many correspondence). The matching measure for a correspondece (i, a) is called *affinity*. Furthermore, an additional *pairwise affinity* is defined that measures the compatibility between two pairs of features (i, j) and (a, b). Having acquired a list L of n candidate matches, affinities of each matching $(i, a) \in L$ and pairwise affinities of each matching $(i, a), (j, b) \in L$ are stored in a symmetric matrix $M \in \mathbb{R}_{+}^{n_P n_Q \times n_P n_Q}$. Specifically:

1. $M_{ia,ia}$ represents the affinity of individual matches $(i, a) \in L$.
2. $M_{ia,jb}$ measures the affinity of the pairwise relationship between the pairs of features $(i, j) \in P$ and $(a, b) \in Q$.

The correspondence problem can be described as the computation of cluster C of correspondences (i, a) that maximizes the inter-cluster score $S = \sum_{ia,jb \in C} M_{ia,jb}$ respecting the mapping constraints. The cluster C can be represented by a binary vector x such that $x_{ia} = 1$ if $(i, a) \in C$ and $x_{ia} = 0$ if $(i, a) \notin C$. Thus, the inter-cluster score is written as

$$S = \sum_{ia,jb \in C} M_{ia,jb} = x^T M x \tag{3.6}$$

Hence, the optimal binary vector x^* is defined as:

$$x^* = \arg\max_x x^T M x \tag{3.7}$$

Specifically, x^* is acquired by applying *Rayleigh quotient* [22] such that $w^* = \arg\max_x R(M, x)$. Thus, we have:

$$x^* = \arg\max_x \frac{x^T M x}{x^T x}, x \in \mathbb{R}^{n_P n_Q} \tag{3.8}$$

where x^* is the principal eigenvector of M that maximizes the inter-cluster score $x^T M x$.

In Section 5.3 we present two post-processing methods we employed in order to discretize the resulting vector according to the desired mapping constraints (one-to-one correspondence).

Chapter 4

Feature Points for Shape Corspondence

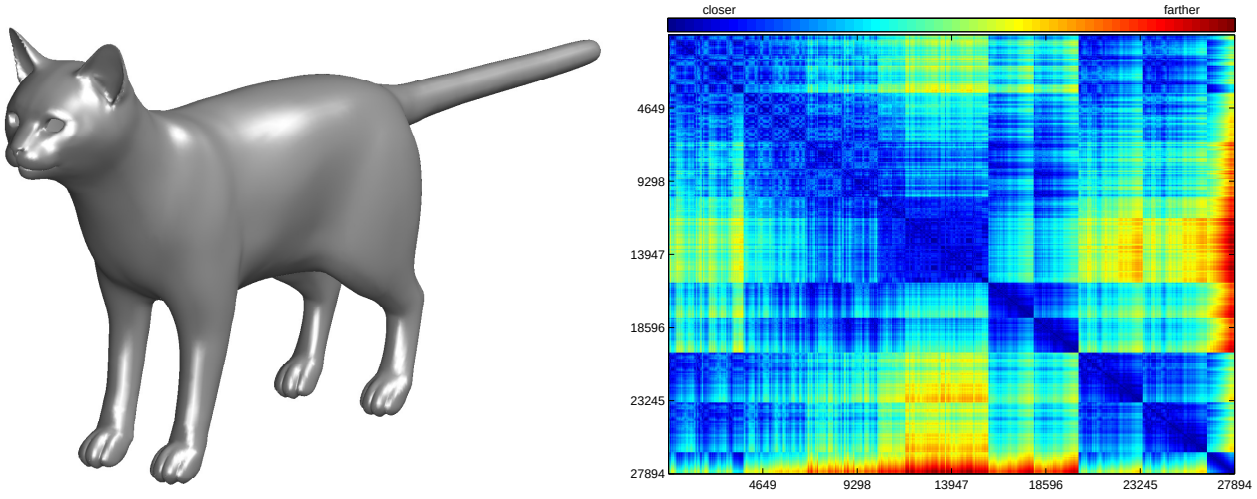
This chapter describes the proposed surface descriptor and feature detection process.

4.1 Distance Map

In the first step of the framework we compute the distance map, an array of all pairs of geodesic distances between the vertices of the mesh. This array is used whenever geodesic distance retrieval is required that is, in both first and second order affinity calculations as well as during matching error (see Sec. 5.4).

Thus, we construct the *distance map* defined by the matrix $D \in \mathbb{R}_{+}^{n \times n}$, where n is the number of vertices, such that $D_{i,j} = d_g(i, j)$. For this purpose, we employ fast marching framework [50] implemented in [56], which approximates geodesic distances on *2-manifolds* in $\mathbb{R}^3$. The process consists of repeating n times the fast marching algorithm in order to compute the geodesic distances from each of the n vertices to all others. The process of computing the geodesic distance from one vertex to all others is called *geodesic propagation* and the vertex at the origin of a propagation is called *source point*. Specifically, in each step $s \in [0, 1, \ldots, n-1]$ we calculate a column-wise vector $v_s = d_g(s, .)$ of n elements containing all distances from source s. By merging these vectors vertically the *distance map* is constructed (see Fig. 4.1b). The complexity of the procedure is $O(nm)$, with m being the complexity of the *geodesic propagation*. Using fast marching scheme the complexity of the entire process becomes $O(n \log(n))$.

Given the geodesic distance definition, it is expected for the *distance map* to be symmetric. However, this is not the case since the fast marching algorithm does not compute exact distances thus introducing an *approximation error*. In order to make the geodesic distance retrieval more accurate, we introduce the *mean geodesic distance* defined as $\overline{D_{i,j}} = \frac{1}{2}(D_{i,j} + D_{j,i})$.



(a) Cat model in neutral pose.

(b) Distance map.

Figure 4.1: Depiction of a mesh and the corresponding distance map. Horizontal and vertical axes represent the endpoint vertices of a discrete geodesic path on the mesh.

4.2 Vicinity Area Descriptor Definition

The measurement of polygonal area around a point reflects the formation of the triangles. Our approach uses the area in the vicinity of a vertex, which is an intrinsic property of the surface, as a base metric. In contrast to point metrics (e.g. curvature), our descriptor accumulates shape information from a region around a vertex. The basic idea behind the *vicinity area descriptor* is that, area deformation follows any infinitesimal changes of edge lengths during near-isometric deformations. The presented descriptor is designed to be multi-scale. Strictly speaking, it is capable of characterizing regions of varying size and exposing the degree of non-planarity.

Initially, consider the intrinsic *isocurve* or *isoring* $C_{\mathcal{M}}(x_o, \rho)$ on manifold $\mathcal{M}$ around a point $x_o \in \mathcal{M}$ as defined by Tevs *et al.*[54]. That is, $C_{\mathcal{M}}(x_o, \rho) = \{y \in \mathcal{M} \mid d_{\mathcal{M}}(x_o, y) = \rho\}$, where ρ is a scalar value and $d_{\mathcal{M}}$ is the geodesic distance function on $\mathcal{M}$ as defined in Section 3.1. In addition, ρ is interpreted as the *geodesic radius* of the isoring $C_{\mathcal{M}}(x_o, \rho)$.

We denote $\mathcal{P}^C_{\mathcal{M}}(x_o, \rho)$ the set of points in the vicinity of central point $x_o \in \mathcal{M}$ bounded by the isocurve $C_{\mathcal{M}}(x_o, \rho)$. That is, $\mathcal{P}^C_{\mathcal{M}}(x_o, \rho) = \{y \in \mathcal{M} \mid d_{\mathcal{M}}(x_o, y) \leq \rho\}$.

Considering $\mathcal{M}$ as a mesh $\mathcal{S}$ and x_o as a vertex $v_o \in \mathcal{S}$, the vicinity area descriptor is described as the sum of the areas of the triangles τ included in $\mathcal{P}^C_{\mathcal{S}}(v_o, \rho)$. Thus, let $\mathcal{T}$ be the set of the triangles inside the vicinity such that $\mathcal{T} = \{\tau \mid \overline{D_{v_o,v_1}} \leq \rho, \overline{D_{v_o,v_2}} \leq \rho, \overline{D_{v_o,v_3}} \leq \rho, \tau \in \mathcal{S}\}$, where v_1, v_2, v_3 are the vertices of a triangle τ and $\overline{D}$ is the mean geodesic (see Sec. 4.1). Hence, the vicinity area descriptor is described as:

$$\mathcal{A}^{\mathcal{T}}_{\mathcal{S}}(v_o, \rho) = \sum_i \alpha(\tau_i) \tag{4.1}$$

where $\tau_i \in \mathcal{T}$ and $\alpha(\tau)$ is the polygon area function.

Extension to Multiple Vicinities

We extend the area single vicinity descriptor to arbitrary number of areas around the central vertex, bounded by exponentially distributed isocurves. Hence, the descriptor is transformed from a scalar value to a vector. Moreover, it better corresponds to intuitions about curvature and captures the degree of anisotropy. This extension is implemented in spirit of the use of multiple isocurves in Landmark-based descriptors [54] and heat equation in Heat kernel signature [53]. Multiple regions are defined based on the outer isoring $C_{\mathcal{M}}(x_o, \rho_B)$ and the central vertex (see Fig. 4.2).

Geodesic radii of each isoring follow a *geometric sequence* $\{a_k\}, k \in \{1, 2, \dots\}$ such that $a_k = r^k$ where r is the *common ratio*. Specifically, given the total number of isorings R and the geodesic radius of the base ring ρ_B, geodesic radius of every ring is given by:

$$\rho_d = \Big[\exp(\frac{\log \rho_B}{R}) \Big]^d \tag{4.2}$$

where d is the index of each isoring such that $d \in \{R, \dots, 1\}$ and $\rho_R = \rho_B$. Base radius ρ_B is alternatively called *maximum propagation distance* since it also denotes the distance limit of the propagation algorithm of the geodesic function. The selection of number of vicinity areas R is of prime importance and adapts to the experiment requirements (e.g. complexity of the mesh, results accuracy, running costs).

This extension intends to capture intrinsic properties of the shape in more detail and make the descriptor more distinctive. For instance, exponential distribution of isorings is more likely to capture the uniqueness of triangles formation that comprise the tip of a nail on a cat mesh (see Fig. 4.2). This occurs because triangulation tends to become more complex as geodesic radius decreases (getting closer to the tip). Hence, for parts of the shape with growing complexity, it is more efficient to decrease the distance between consecutive samplings (in our case, distance between isorings). It is more efficient to accumulate sampling in more complex triangulations since we better capture the smaller deviations of the descriptor. In case we employed linear distribution of samplings, big steps between successive isorings radii would probably overlook small yet significant triangulation alterations. In Figure 4.2 we present both distributions of multiple vicinities. Worth mentioning that base isorings $C_{\mathcal{M}}(x_o, \rho_B)$ are identical since they define the basis of the outer (base) vicinity discretization. A depiction of linear and exponential isoring distribution can also be found in Figure 6.4.

Experiments on varying number of radii and comparison results on exponential over linear isoring distribution are presented in Section 6.1 and Section 6.2 in respect.

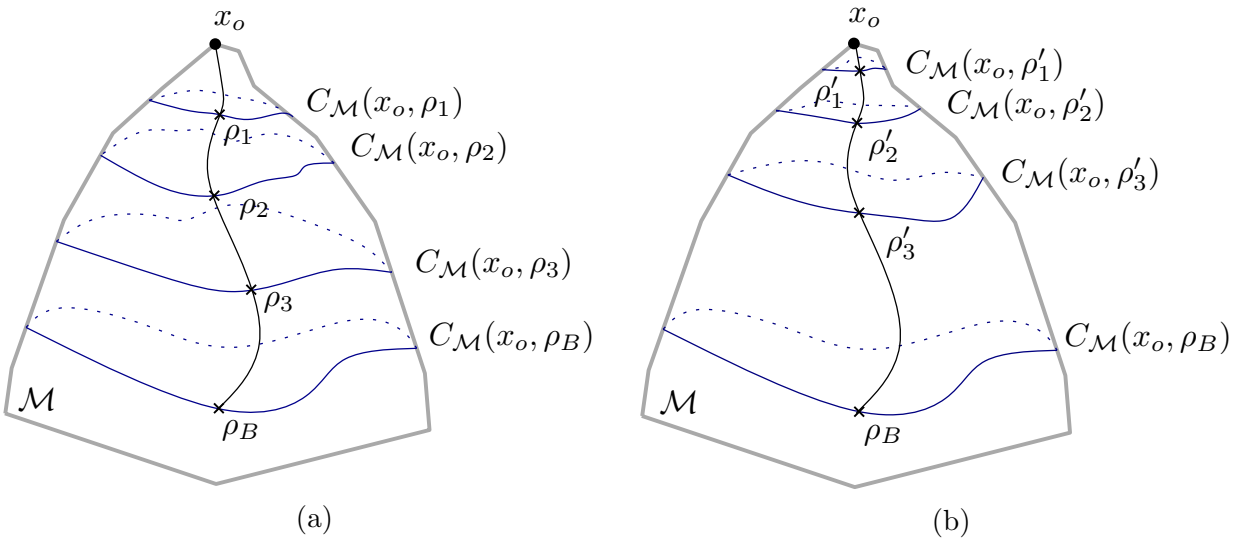


Figure 4.2: Linear (a) and exponential (b) vicinity discretization.

4.3 Descriptor Score

The quantification of the feature descriptor is not comprised by a scalar but from multiple values depending on the number of vicinity areas around the central vertex. Hence, each score in source mesh is compared to its corresponding score on the target mesh. Worth noting that we normalize each score by dividing by $\pi\rho^2$, which is the expected area of flat piece of surface. To wit, for central vertex v_o and k different radii, feature score is denoted by the vector:

$$\mathcal{I}_{v_o} = (\frac{1}{\pi\rho_1^2}\mathcal{A}_{\mathcal{S}}^{\mathcal{T}}(v_o, \rho_1), \frac{1}{\pi\rho_2^2}\mathcal{A}_{\mathcal{S}}^{\mathcal{T}}(v_o, \rho_2), \ldots, \frac{1}{\pi\rho_k^2}\mathcal{A}_{\mathcal{S}}^{\mathcal{T}}(v_o, \rho_k)) \tag{4.3}$$

where $\rho_{1\ldots k}$ are the radii in ascending order. Following the normalization step, all components of vector descriptor $\mathcal{I}_{v_o}$ are inside the range $[0, 1]$. High number of isorings do not always guarantee better results since the process can fall into local minima. For instance, for simple parts of the mesh, same score can be repeated along several isorings (no changes in area) or yield zero score after reaching a certain point (isorings fall inside a single triangle).

Three-dimensional polygonal area computation

Score computation involves three-dimensional polygonal area calculations. Since the polygon is not planar, we compute its area by summing the areas of its structural elements on which a surface normal can be defined (three-dimensional triangles). Thus, we express the area of the triangle by the magnitude of the cross-product of two edge vectors [20]. Consider the definition of magnitude of the cross product: $|\nu \times w| = |\nu||w||sin(\theta)|$, where θ is the angle among the two vectors ν and w. For a three-dimensional triangle τ_{v_0,v_1,v_2} with vertices

v_0, v_1, v_2, we set $\nu = v_1 - v_0$ and $w = v_2 - v_0$. The polygon area function $\alpha(\tau_{v_0,v_1,v_2})$ (see Eq. 4.1) is thus defined as:

$$\begin{aligned}\alpha(\tau_{v_0,v_1,v_2}) &= \frac{1}{2}|\nu \times w| = \frac{1}{2}|(v_1 - v_0) \times (v_2 - v_0)| \\ &= (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0),\end{aligned} \tag{4.4}$$

where $v_i = (x_i, y_i, 0)$ are triangle vertices.

Worth noting that, the aforementioned formula is computationally efficient, since it does not involve calculations of trigonometric functions or roots.

4.4 Feature Extraction

During feature extraction, we search for the most descriptive features on the mesh , since this decreases correspondence error during the matching phase. For this reason, we apply non-maximum suppression on the vicinity defined by $C_{\mathcal{M}}(x_o, \rho_B)$.

NMS is interpreted as local maximum search. That is, the solution relies on detecting a local maximum that is greater than all others in a given neighborhood. It is widely applied on computer vision algorithms for the extraction of salient points throughout an image, or even the whole scale space [31][37][60].

Regarding the proposed framework, the neighborhood is determined by the isocurve $C_{\mathcal{S}}(v_o, \rho_k)$ where v_o is the central vertex and ρ_k is the largest of the geodesic radii (i.e. ρ_B) $k \in [1, 2, \ldots, k]$. We define the vertex with the largest score to be the *dominant vertex* denoted by v_o^*. The vertices of which the scores are compared for maximality belong to the set $\mathcal{P}_{\mathcal{S}}^{C}(v_o, \rho)$. Worth noting that, the input base radius ρ_B in NMS has to be carefully selected, since it has linear relation with the number of the extracted feature points. To wit, in extreme cases, matching will consider a single vertex or all vertices (dense map) as feature point(s).

Worth mentioning that, small ρ_B does not necessarily implies denser located features (if, for instance, examine the results for ρ_B with small difference). This is because, besides the current vicinity, candidate feature rejection will possibly repeat several times since a points with higher score will get detected in each vicinity. This will cause the next feature to result far enough from the previous one (in greater distance than the initial vicinity radius).

In order to accelerate feature extraction process, we added an exclusion algorithm regarding NMS. That is, each time a v_o^* is detected, all vertices in the vicinity set $\mathcal{P}_{\mathcal{S}}^{C}(v_o^*, \rho)$ are excluded from the remaining feature extraction process. This extension is crucial for experiments with complex meshes and large ρ_B, since a large number of vertices is excluded from the feature search.

In Figure 4.3 we represent detected feature points along with their scores (interpreted by the color of each sphere). A detailed comparison is presented in Figure 6.3 in Section 6.1.

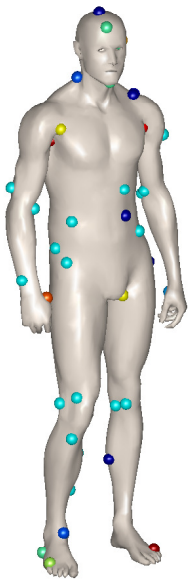


Figure 4.3: Human model in neutral pose along with 45 detected features (colored spheres) for $\rho_B = 8.580805$. Warmer colors indicate higher scores.

$feature\_vertex_1$	$isoring\_score_1^k$	$isoring\_score_1^{k-1}$	$\dots$	$isoring\_score_1^1$
$feature\_vertex_2$	$isoring\_score_2^k$	$isoring\_score_2^{k-1}$	$\dots$	$isoring\_score_1^2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$feature\_vertex_n$	$isoring\_score_n^k$	$isoring\_score_n^{k-1}$	$\dots$	$isoring\_score_n^1$

Figure 4.4: Feature list representation along with the corresponding isoring scores. Where n is the number of extracted features and k is the total number of isorings.

4.5 Feature Computation Algorithm

This section provides the algorithmic representation of feature extraction and multiple-ring score computation (see Alg. 1). For a depiction of the result L_a^{vic}, refer to Figure 4.4. This involves the first part of the proposed framework since it does not include any of the subsequent matching procedures (see Chap. 5).

Algorithm 1: Feature Extraction and Multiple-Ring Score Computation

Input: mesh $m_{v,f}$ where v vertices and f faces $\in m_{v,f}$,
max propagation distance d_{max},
number of isorings R
Output: list of feature vertices with the corresponding ring scores $L^{feat}_{\nu,a^{rng}}$

$map_{m_{v,f}} \leftarrow MapVerticesToFaces(m_{v,f})$
$L^{vic}_{\nu}, L^{vic}_{a} \leftarrow$ `ComputeDescriptorScore(`$L^{m_{v,f}}_{\nu}, d_{max}$`)`
`/* entering` ***NMS*** `*/`
foreach $v^{curr} \in m_{v,f}$ **do**
  $isfeature \leftarrow true$
  foreach $v^{vic} \in L^{vic}_{\nu}(v^{curr})$ **do**
    if $L^{vic}_{a}(v^{vic}) > L^{vic}_{a}(v^{curr})$ **then**
      $isfeature \leftarrow false$
  if $isfeature = true$ **then**
    `/* exclude vertices in` $L^{vic}_{\nu}(v^{curr})$ `from` ***NMS*** `search */`
    $ExcludeNms(L^{vic}_{\nu}(v^{curr}))$
    `/* list including feature points and ring scores (base ring for now) */`
    $L^{feat}_{\nu,a^{r}} \leftarrow < u^{curr}, L^{vic}_{a}(v^{curr}) >$
    `/* compute scores for inner rings */`
    for $r = R + 1$ **to** 1 **do**
      $\rho_r \leftarrow InnerRingGeodesicRadius(r, d_{max})$
      $a_r \leftarrow$ `ComputeDescriptorScore(`$L^{vic}_{\nu}(v^{curr}), \rho_r$`)`
      $L^{feat}_{\nu,a^{rng}} \leftarrow a_r$
  return $L^{feat}_{\nu,a^{rng}}$

`/* input arguments:a list of vertices and a propagation distance */`
Function `ComputeDescriptorScore(`L_v, d`)`
  foreach $v \in L_v$ **do**
    `/* store vetrices in the list` $L^{vic}_{\nu}(v)$ `*/`
    $L^{vic}_{\nu}(v) \leftarrow StoreVerticesInVicinity(v, d)$
    $L^{vic}_{f}(v) \leftarrow StoreFacesInVicinity(map_{m_{v,f}}, L^{vic}_{\nu}(v))$
    $area \leftarrow SumArea(L^{vic}_{f}(v))$
    `/* list including vicinity area for all vertices of the mesh */`
    $L^{vic}_{a}(v) \leftarrow Normalize(area)$
  return $L^{vic}_{\nu}, L^{vic}_{a}$

Chapter 5

Isometry-Invariant Matching

In this chapter we present the method of incorporating spectral matching [29] introduced in Chapter 3.3 in order to facilitate the near-isometric mesh matching framework (first eigenvector computation through *iterative power method* and *double stochastic conversion* step implementation can be found in [13]). Since spectral matching can be used for point registration in $\mathbb{R}^d$, we employ it in order to match two set of points defined on a pair of 2-manifolds. That is, we represent the points as graph nodes and their distances as graph edges. Thus, feature correspondence problem is reduced to solving the bipartite graph matching problem.

Let two sets of points $S = \{x_i\}_{i=1}^{n_1}$ and $T = \{y_i\}_{i=1'}^{n_2}$ in $\mathbb{R}^3$ be the sets of extracted feature points from source and target meshes in respect. The solution to the assignment problem relies on acquiring the set of assignments described as

$$C \coloneqq \{c_{i,i'}\}_{i=1}^{n} = \{x_i, y_{i'}\}_{i=1}^{n}, \tag{5.1}$$

where $n \leq min(n_1, n_2)$. The correspondences (candidate assignments) are represented by the binary vector $z \in \{0,1\}^{n_1 n_2}$ which is the row-wise reformation of the assignment matrix $Z \in \{0,1\}^{n_1 \times n_2}$ such that $Z_{ii'} = 1$ *iff* x_i corresponds to $y_{i'}$ and $Z_{ii'} = 0$ *iff* x_i does not match with any $y_{i'}$.

Worth noting that, candidate assignments are a superset of the solution assignments C. Hence, we define the set of candidate assignments C^c and we extend the range of occurrences such that $C^c \supseteq C$ and

$$C^c = \{c_{i,i'}\}_{i=1}^{n_1 n_2} \tag{5.2}$$

Considering the sets S and T, we denote the m-order affinity measure Ω_m, as the measure that quantifies the matching of m pairs of points. To wit:

$$\Omega_m = \Omega_m(c_{i_1}, \ldots, c_{i_m}) = \Omega_m(i_1, \ldots, i_m) = \Omega_m(\{x_{i_1}, y_{i'_1}\}, \ldots, \{x_{i_m}, y_{i'_m}\}) \tag{5.3}$$

where $i_s \in \{i_1, i_2, \ldots, i_{m-1}, i_m\}$ is the index of each pair of points (equivalently symbolized by consecutive lowercase letters similar to $\{i, j, k, \ldots\}$). By incorporating spectral matching, we combine first-order affinities Ω_1 (vertex-to-vertex) and second-order affinities Ω_2(vertex pair-to-vertex pair) into a single solution.

5.1 Linear Assignment Affinity

Linear assignment affinity matrix describes the matching affinity among features of the two shapes. That is, for extracted feature points $x_i \in S$ and $y'_i \in T$ equipped with arbitrary number of rings, affinity is expressed as the *sum of squared differences* (SSD) for each ring in a Gaussian function such that

$$\Omega_1(x_i, y_{i'}) = \exp\Big(- \frac{1}{2{\sigma_1}^2} \sum_{k}^{R} (\mathcal{I}_{x_i}(k) - \mathcal{I}_{y_{i'}}(k))^2 \Big) \tag{5.4}$$

where k denotes the index of each ring, R the total number of rings and $\mathcal{I}_{v_o}(k)$ the score of the k-th ring corresponding to the feature defined on vertex v_o (see Eq. 4.3). Furthermore, σ_1 denotes the *standard deviation* (sd_1) and is interpreted as the tolerance referring to the difference of feature point scores.

Hence, the Gaussian weighted affinity matrix is defined as

$$A^l_{i,i'} = \Omega_1(x_i, y_{i'}) = \Omega_1(c_{i,i'}), \quad c_{i,i'} \in C^c \tag{5.5}$$

The optimal linear assignment is given by

$$\begin{aligned} z^* &= \arg\max_{z} (z^T \mathrm{a}), z^* \in \{0,1\} \\ s.t. \;\; & Z^* \mathbb{1} \leq \mathbb{1} \;\; and \;\; (Z^*)^T \mathbb{1} \leq \mathbb{1} \end{aligned} \tag{5.6}$$

where $\mathbb{1}$ is the *all-ones vector*, $\mathrm{a} \in \mathbb{R}^{n_1 n_2}$ a row-wise reformation of A^l and $Z^* \in \{0,1\} \in \mathbb{R}^{n_1 \times n_2}$ a row-wise reformation of z^*.

The constraints condition in Equation 5.6 suggests that in case a row of Z^* includes more than a single *binary one*, the binary vector z^* would include values larger than 1. This implies that, only a point $x_i \in S$ can be matched with a point $y'_i \in T$ or not matched at all.

The solution to Equation 5.6 can be computed using the Hungarian algorithm in polynomial time [41], binary linear programming [42] or approximated by Dynamic Programming [14].

5.2 Pairwise Affinity

Pairwise affinities Ω_2 measure the simultaneous matching of two assignments $c_{i,i'}$ and $c_{j,j'}$. An important property is that they can capture *local isometry* of the shape. Hence, second-order affinities are expressed by the Gaussian function:

$$\Omega_2(\{x_i, y_{i'}\}, \{x_j, y_{j'}\}) = \exp\Big(- \frac{1}{2{\sigma_2}^2} (\overline{D_{x_i,x_j}} - \overline{D_{y_{i'},y_{j'}}})^2 \Big) \tag{5.7}$$

where $\overline{D_{ij}}$ is the geodesic distance among (i,j) (see Sec. 4.1) and σ_2 the standard deviation (sd_2) expressing the tolerance regarding the difference among the two geodesic distances.

Similarly to Equation 5.5, the Gaussian weighted second order affinity matrix is given by

$$A^p(i(n_2-1)+i', j(n_2-1)+j') = \Omega_2(c_{i,i'}, c_{j,j'}) \tag{5.8}$$

In addition, the solution is given by the assignment that maximizes the sum of the corresponding pairwise affinities

$$C^* = \arg\max_C \sum_C \Omega_2(c_{i,i'}, c_{j,j'}), \quad c_{j,j'}, c_{i,i'} \in C^c \tag{5.9}$$

This, implies the following optimization problem:

$$\begin{aligned} z^* &= \arg\max_z (z^T A^p z), \quad z^* \in \{0,1\}^{n_1 n_2} \\ &s.t. \;\; Z^* \mathbb{1} \leq \mathbb{1} \;\; and \;\; (Z^*)^T \mathbb{1} \leq \mathbb{1} \end{aligned} \tag{5.10}$$

where z is the binary row-wise vector and Z the indicator matrix (see Eq. 5.6) and $A^p \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$.

5.3 Combinatorial Affinity Matrix

Motivated from the *adjacency matrix* Leordeanu and Hebert presented [29], we combine linear assignment and pairwise affinities introduced previously into a signle matrix. We define the *combinatorial affinity matrix* A^c based on A^l and A^p such that $A^c \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$.

Given two candidate assignments of model features $c_i = \{x_i, y_{i'}\}$, $c_j = \{x_j, y_{j'}\}$, A^c represents the affinity level of individual assignments if $c_i = c_j$ and the affinity level of pairwise correspondences if $c_i \neq c_j$. Since pairwise affinity for identical pairs can be interpreted as point-wise affinity, $diag(A^p)$ is replaced by a $\in \mathbb{R}^{n_1 n_2}$ (vectorized form of A^l) in order to construct A^c. Specifically

$$A^c_{i,j|i',j'} = \begin{cases} A^l_{i,i'} = \Omega_1(c_{i,i'}) & \text{if } A^c_{i,j|i',j'} \in diag(A^c_{i,j|i',j'}) \\ A^p_{i,j|i',j'} = \Omega_2(c_{i,i'}, c_{j,j'}) & \text{if } A^c_{i,j|i',j'} \notin diag(A^c_{i,j|i',j'}) \end{cases} \tag{5.11}$$

Similarly to Section 3.3 and 5.2, the acquired binary vector solution z^* is represented as

$$z^* = \arg\max_z (z^T A^c z) \tag{5.12}$$

Since our framework demands *one-to-one feature* correspondences, solution z^* must follow a post-processing quantization that imposes these mapping constraints (see end of Sec. 3.3).

The first solution implies the use of Hungarian algorithm [41] as implemented in [24]. That is, it calculates the discrete solution z^* that maximizes the dot product with the eigenvetor ν, where ν is the principal eigenvector of A^c (see Eq. 5.12). That is:

$$z^* = \arg\max_z (z^T \nu) \tag{5.13}$$

Concerning the second solution, we follow the Greedy algorithm Leordeanu *et al* proposed, summarized in the following steps:

$$A^c = \begin{pmatrix} \Omega_1(c_{1,1}) & \Omega_2(c_{1,1}, c_{1,2}) & \cdots & \Omega_2(c_{1,1}, c_{n_1,n_2}) \\ \Omega_2(c_{1,2}, c_{1,1}) & \Omega_1(c_{1,2}) & \cdots & \Omega_2(c_{1,2}, c_{n_1,n_2}) \\ \vdots & \vdots & \vdots & \vdots \\ \Omega_2(c_{1,n_2}, c_{1,1}) & \Omega_2(c_{1,n_2}, c_{1,2}) & \cdots & \Omega_2(c_{1,n_2}, c_{n_1,n_2}) \\ \Omega_2(c_{n_1,1}, c_{1,1}) & \Omega_2(c_{n_1,1}, c_{1,2}) & \cdots & \Omega_2(c_{n_1,1}, c_{n_1,n_2}) \\ \vdots & \vdots & \vdots & \vdots \\ \Omega_2(c_{n_1,n_2}, c_{1,1}) & \Omega_2(c_{n_1,n_2}, c_{1,2}) & \cdots & \Omega_1(c_{n_1,n_2}) \end{pmatrix}$$

Figure 5.1: Depiction of the combinatorial affinity matrix A^c including first-order Ω_1 and second-order Ω_2 affinities.

1. Let z^* be the leading eigenvector of A^c. Initialize the solution vector z with $n_1 n_2 \times 1$ zero vector. Initialize a set L as the set of all candidate correspondences (i, i').
2. Compute $(i, i')^* = \arg\max_{i,i' \in L} z^*_{i,i'}$. If $z^*_{i,i'} = 0$, stop and return the solution z. Otherwise set $z_{i,i'} = 1$ and remove $(i, i')^*$ from L.
3. Remove all candidate correspondences in conflict with $(i, i')^*$.
4. If L is empty return z as solution. Otherwise go to step 3.

Both algorithms were applied in our framework and the results are presented in Chapter 6.

Considering the definition of C^c (see Eq. 5.2), a representation of the combinatorial affinity matrix A^c can be found in Figure 5.1.

5.4 Correspondence Vector and Matching Error

The resulting binary vector from Section 5.3 is reformed to the fundamental solution representation, called *correspondence vector*. Correspondence vector is a $|c| \times 2$ vector indicating the derived pairs of corresponding vertices from source to target shape, where $|c|$ is the number of correspondences. This vector is not only employed during the matching error calculation but also for the depiction of the result.

We define matching error e as the sum of deviations of matches from ground truth, divided by the number of correspondences. Match deviation is considered as the geodesic distance of the vertex i' from the exemplar vertex i'_G on the target shape (see Fig. 5.2). Specifically:

$$e = \frac{1}{|c|} \sum_i e_{ii'} = \frac{1}{|c|} \sum_i d_g(i', i'_G) \tag{5.14}$$

Worth noting that, $e \in [0, L^T_{max}]$, where L^T_{max} is the longest geodesic path on target shape.

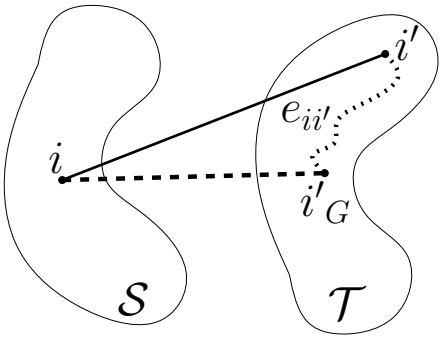


Figure 5.2: Representation of matching error calculation for a single correspondence between source and target shape.

5.5 Matching Computation Algorithm

The following algorithm describes the second phase of the matching framework. To wit, having computed feature descriptors for both shapes, it calculates the correspondence vector and matching error and depicts the result.

Algorithm 2: Correspondence vector computation

Input: source mesh m,
target mesh m',
m feature vertices with the corresponding ring scores, $L^{feat}_{\nu,a^{rng}}$,
m' feature vertices with the corresponding ring scores, $L^{feat}_{\nu,a^{rng}}{}'$,
comparison tolerance of feature descriptor scores σ_1,
comparison tolerance among geodesic distances σ_2
Output: correspondence vector Hungarian $corrv_h$,
correspondence vector Greedy $corrv_g$,
correspondence error Hungarian err_h,
correspondence error Greedy err_g

`/* geodesic distances among feature points */`
$A^2 \leftarrow ComputeSecondOrderAffinityArray()$
`/* combinatorial affinity matrix */`
$A^c \leftarrow ComputeCombinatorialAffinityArray(L^{feat}_{\nu,a^{rng}}, L^{feat}_{\nu,a^{rng}}{}', \sigma_1, \sigma_2, A^2)$
`/* entering spectral matching, binary vector */`
$binv_h, binv_g \leftarrow SpectralHandler(A^c, L^{feat}_{\nu,a^{rng}}, L^{feat}_{\nu,a^{rng}}{}')$
$corrv_h, corrv_g \leftarrow CorrespondencesFirstSecondOrder(binv_h, binv_g, L^{feat}_{\nu,a^{rng}}, L^{feat}_{\nu,a^{rng}}{}')$
$err_h, err_g \leftarrow CalculateError(corrv_h, corrv_g)$
$VisualizeCorrespondences(m, m', corrv_h, corrv_g)$
return $corrv_h, corrv_g, err_h, err_g$

Chapter 6

Results and Analysis

Experiments include a variety of benchmark models involving feature point extraction, matching error and performance measurements. To wit, we test the behaviour of the proposed framework for multiple variable combinations as well as additional special features. We proceed with the comparison to existing internal functions (processes focused to solve a specific part of the algorithm) and state-of-the-art shape matching methods. Framework evaluations were held using meshes from the standard TOSCA [57] data set. The data set includes synthetic triangle meshes of approximately 50,000 vertices. Matching experiments refer to pairs of meshes purposely selected to represent intense deformations with each pair falling into a single model class (i.e. no tests between cats and dogs). Furthermore, we simplify the input meshes using *Quadric Edge Collapse Decimation* [61] filter for efficiency reasons. That is, both source and target meshes have a minimum point spacing of $0.8 \ldots 0.1\%$ of the longest bounding box size. Platform specifications are *Intel i7-2630QM 2,00 GHz* and *6 GB DDR3 RAM*. Implementation involves C++ (with extensive use of C++11 features) and *MATLAB 8.1 R2013a* (for details refer to Sec. 8).

We firstly deal with benchmarks concerning the proposed local descriptor and then the shape matching part follows. A separate subsection is dedicated on each experiment. We initially provide statements of observations, including statistics, tables and graphs. Then, the interpretation of the outcome follows, pointing out the relationships, trends and generalizations among the results.

6.1 Local Signature

This section describes the benchmarks and examines the results regarding the proposed descriptor.

Descriptor Score Distribution

This benchmark presents the behaviour of the proposed area descriptor regarding different base radii. Figure 6.2 depicts how local signature scores (1-isoring case) spread over the shape or accumulate to specific locations.

Smaller base radii ρ_B imply scores scattered around various values. To wit, they indicate a great number of locations that present higher scores (comparing to the rest of scores on the current mesh). This is expressed by the large variety of colors over the mesh (see Fig. 6.2(a)). Hence, the framework is responsive to salient points with small deviations from other scores on the shape. The descriptor exposes more details but feature detection is intolerant to imperfect isometries. Performance issues have to be also considered, since, as ρ_B decreases, number of feature points increase along with candidate correspondences. This implies unmanageable affinity matrices (size complexity equals to $n_1 n_2 \times n_1 n_2$, where n_1, n_2 are the extracted feature points on source and target shape) that can be only be handled by high-end systems.

On the contrary, larger ρ_B produce scores gathered around certain values. Use of large ρ_B makes the method tolerant to the error that near-isometric changes present, because infinitesimal alternations are "flattened" during descriptor quantification. Hence, salient points are sorted out from minor changes of the triangulation. Worth mentioning that, because of the "smoothing" nature of the proposed descriptor for large base radii, surface properties substantial for establishing a successful matching, are likely to be ignored.

The fact that the area descriptor is not a point descriptor, makes it a signature of multiple purposes. To wit, as showed in Figure 6.2, it "mutates" depending on the circumstances (i.e. base radius), thus exposing different surface properties.

Plots in Figure 6.1 demonstrate the dispersion of the descriptor scores for the basic case of single-isoring descriptor. The term *max propagation* refers to the geodesic radius of the outer isoring, $C_{\mathcal{M}}(x_o, \rho_B)$. It denotes the propagation limit of the geodesic function (starting from a central vertex). The graph shows that variety of scores increases with the increase of max propagation. In order to increase the validity of statistical dispersion measurements, we apply *interquartile range* or *interdecile range* (*iqr*). In contrast to *variance*, *standard deviation* and *mean absolute deviation*, iqr is robust to outliers (for details refer to [36]). Generally, score dispersion demonstrates an increasing monotony, as expected. Any observed dispersion decrease is due to triangulation abnormalities and low triangulation density in specific regions of the surface.

In conclusion, optimal base radius depends on multiple factors some of which are curvature, triangulation density and hardware limitations. In Section 6.2 we investigate matching error for varying ρ_B in combination with several framework variables. It can be seen that area accumulation differs from other intrinsic point properties (e.g. Gaussian curvature). One of the main reasons is that the proposed local signature is not a point but an adjustable-vicinity descriptor. Worth noting that, mesh coloring refers to each of the three shapes individually, thus meaning that same colors do not necessarily indicate identical scores (among two meshes). That is, they denote intra-shape and not inter-shape variations.

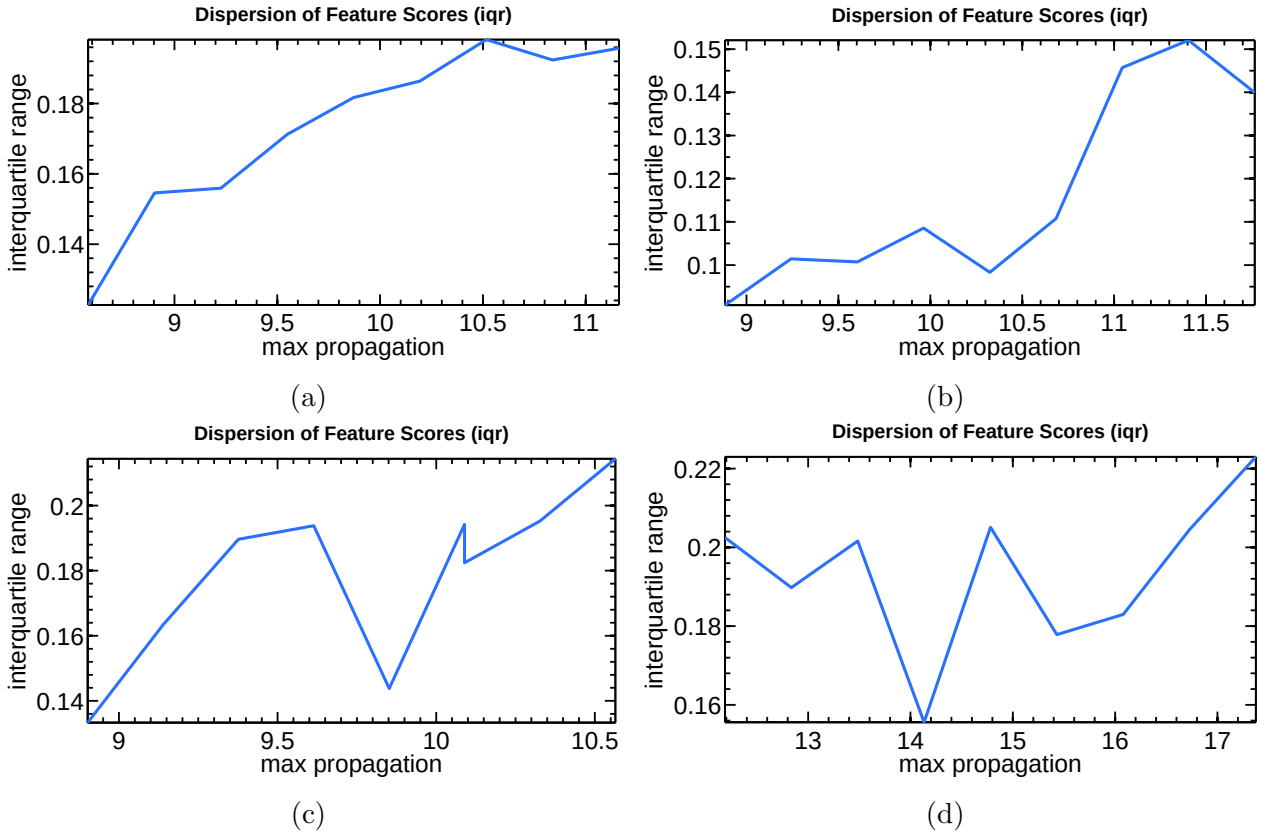
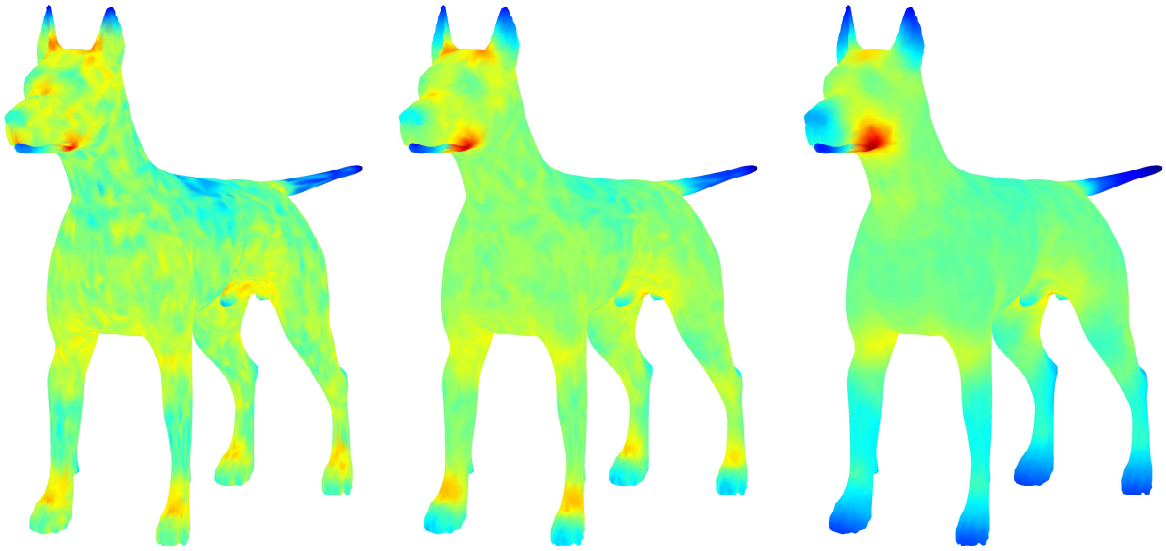


Figure 6.1: Dispersion of feature scores (interquartile range) to maximum propagation (ρ_B) for human (a), cat (b), dog (c) and centaur (d) models.

Non Maximum Suppression and Detected Features

This benchmark analyzes the placements and descriptor scores of features for different base radii (see Fig. 6.3). It can bee seen that, due to NMS, while base radius ρ_B grows larger, the number of detected features decreases and they generally become more sparsely located. In addition, along with the increase of ρ_B, feature scores increase since candidate features enclose larger regions of the mesh. Although higher feature scores reveal high concentration of triangles around a vertex, this does not necessarily enhances the subsequent matching. To wit, it declares the local distinctiveness of the descriptor regarding its vicinity. Worth mentioning that, as features get more sparse, pairwise feature distance tolerance contributes more to the final matching. This is because it decreases ambiguities during correspondences establishment. Besides the fact that the demonstrated monotonic relation applies in general (no specific value dependence), in this case, selected max propagation range is the same applied in matching benchmarks (thoroughly justified in Section 6.2).



(a) $\rho_B = 8.4468$ (b) $\rho_B = 10.0894$ (c) $\rho_B = 15.0171$

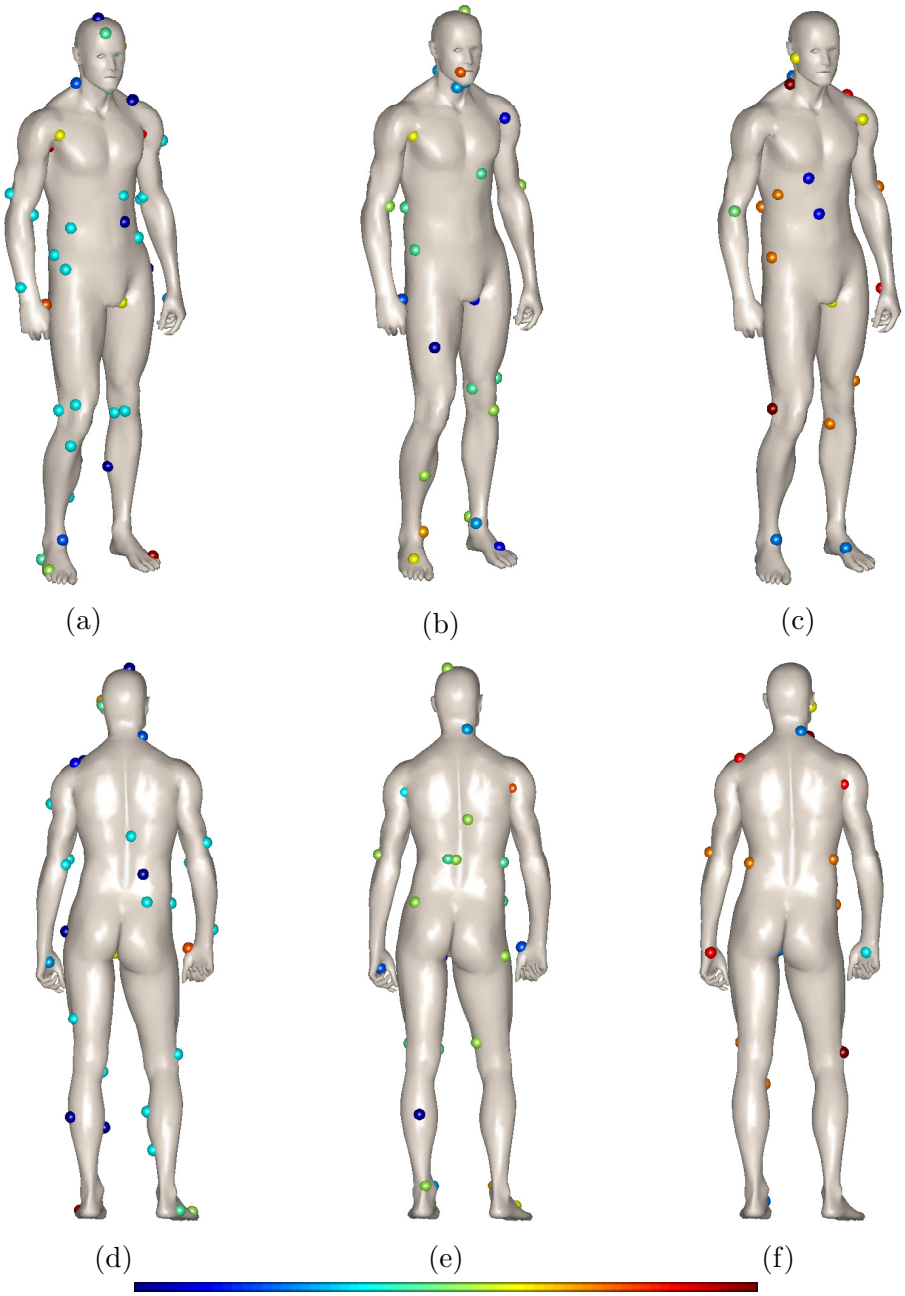
Figure 6.2: Color descriptor score representation for three different ρ_B. Warmer colors indicate higher scores. Descriptor scores become more evenly distributed as ρ_B decreases.

Exponential Isoring Distribution

In this paragraph, we explain the advantages of using exponential over linear isoring distribution when employing vector descriptor. Outcome witnesses that it is more efficient to place the borders of area vicinities around the central vertex according to exponential decrease of geodesic radius.

Figure 6.4 shows the covered vicinity area for each of the isorings for both linear and exponential isoring distribution. Geodesic radius reduction starts based on the radius of the outer isoring (note the identical ρ_B in Figure 6.4(a)(b)). Specifically, it is observed that captured details between the smallest and the subsequent isoring (i.e. $C_{\mathcal{M}}(x_o, \rho_1)$ and $C_{\mathcal{M}}(x_o, \rho_2)$) regarding the elephant's eye, are lost in linear radius increase case. In addition, it is showed that, the majority of inner vicinities in Figure 6.4(a) cover areas with no significant changes among them. thus almost "repeating" the captured information. On the other hand, in figure 6.4(b) inner vicinities consider areas with noticeable shape alternations. Hence, the uniqueness of the descriptor is enhanced. In this example (see Fig. 6.4), base radius is enlarged for demonstration reasons.

In Section 6.2 we provide matching error comparison data for linear and exponential multiple inner isorings distribution. We furthermore examine vector descriptor behaviour for different number of inner isorings.

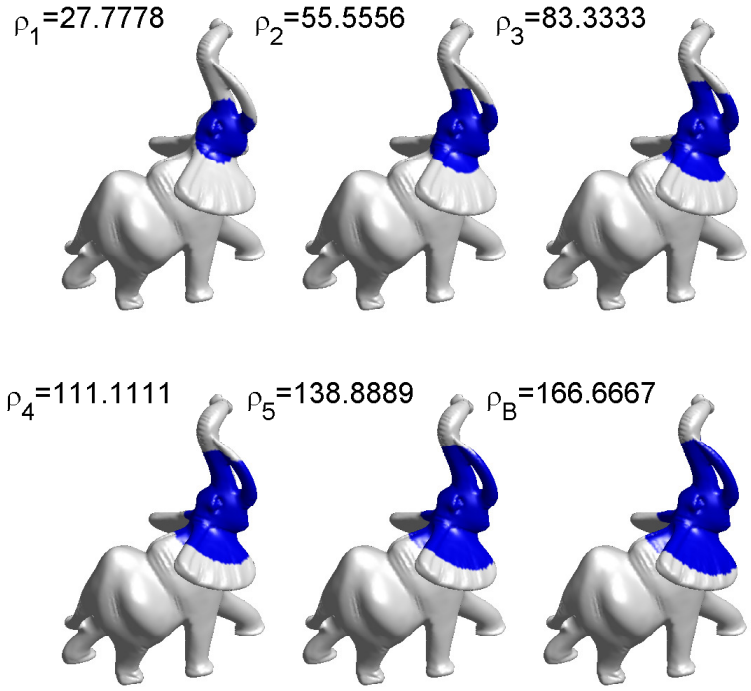


(a) (b) (c)

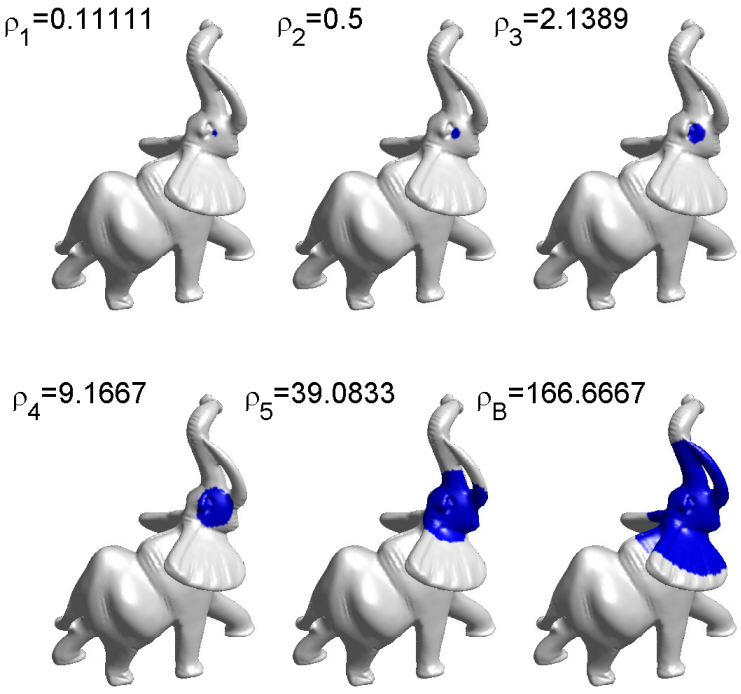
(d) (e) (f)

(g) Scores: low (dark blue) to high (dark red).

Figure 6.3: Detected features f for different base radii ρ_B on human model. Front and back captures for $f = 45, \rho_B = 8.580805$ (a)(d), $f = 34, \rho_B = 9.871207$ (b)(e) and $f = 19, \rho_B = 11.16161$ (c)(f). As sphere color gets warmer, descriptor score increases.



(a) linear decrease of base radius.



(b) exponential decrease of base radius.

Figure 6.4: Vector descriptor propagation areas for linear (a) and exponential (b) decrease of ρ_B (6-element descriptor).

6.2 Shape Matching

Along this section we analyze the experiments referring to shape matching and provide comparisons to existing methods.

4D Scatterplots

The main synthetic benchmark, includes pairs of four-dimensional scatterplot for each pair of meshes (see Fig. 6.5). Each of the three dimensions, *max propagation*, sd_1 and sd_2 denote the selected base radius ρ_B, the standard deviation for the descriptor score and the standard deviation among the pairwise geodesic distances. Strictly speaking, the two standard deviations represent the tolerance of the first and second-order affinities in respect. Color, denotes the fourth dimension and expresses the matching error. We choose to represent matching error not as a dimension in space, in order to emphasize the goal of the experiment. Pairs of plots visualize the results for the two primary internal spectral matching models, that is, Hungarian and Greedy. Every plot includes $9^3 = 729$ matchings induced by 9 consecutive values of each variable. During the tests we consider the basic case of *1-isoring* descriptor (scalar descriptor). We analyze multi-isoring descriptor results based on the best scatterplot matchings in Section 6.2. Base radius limits are selected in order to extract an adequate number of feature points while respecting hardware capabilities. Let $\overline{l_{max}}$ be the average longest edge for source and target shape and $\overline{L_{max}}$ the average longest path on the mesh (referring to same-class mesh pair). Using base radii among $\overline{l_{max}}$ and $2 \times \overline{l_{max}}$, we keep the number of detected feature points among ~ 20 and ~ 44 (resulting correspondences are equal to the minimum number of feature points from the two shapes). Keeping in mind that 70×75 candidate correspondences can produce a ~ 3.5 GB affinity matrix file, we avoid gigabytes-sized combinatorial affinity matrices. Tolerance of area descriptor magnitude is restricted inside $[0.1, 0.9]$ in contrast to the expected $[0, 1]$. Furthermore, standard deviation among pairwise geodesic distances, ranges from $\overline{l_{max}}$ to $\overline{L_{max}}/8$ contrary to the generalized $[0, \overline{L_{max}}]$. Range and discretization of values is selected in order to, not only focus on the most interesting spectrum of the experiment but to deliver an adequate collection of matchings. This four-dimensional scatterplot enhances our macroscopic view on the inter-relations of the matching error and the fundamental framework variables. The best match is selected for each model pair and used as the exemplary variable combination for the subsequent experiments. Matching error is measured as in Section 5.4.

It can be argued that smallest errors occur during the use of small max propagation values since many feature points yield larger number of candidate correspondences. On the other hand, optimal matchings can emerge for large ρ_B. This is because fewer feature points are more stable (explicitly located), thus improving matching success. It is observed that, regarding max propagation, optimal results are detected for values of $5 \ldots 6\%$ of the average longest geodesic path $\overline{L_{max}}$.

In addition, for fixed *max propagation* and sd_2, changes of sd_1 have small influence on the resulting error. To wit, results (over z-axis) remain nearly unchanged regardless the

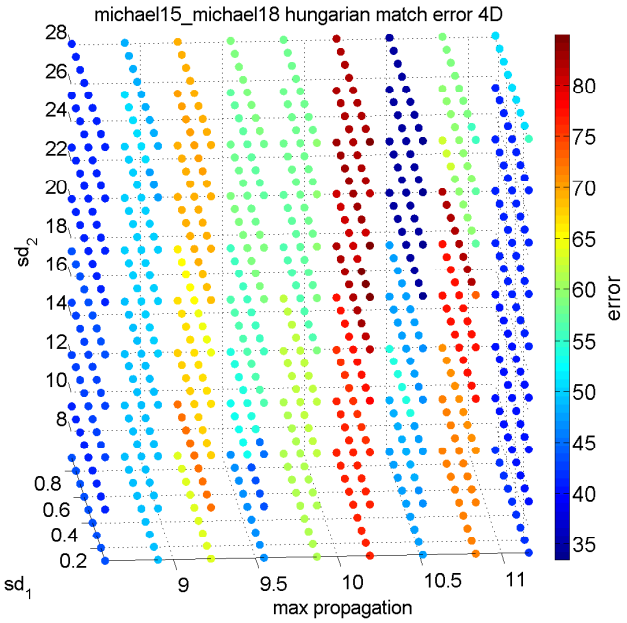
value of sd_1. A conjecture on this observation is that descriptor score differences present smaller variance comparing to the pairwise geodesic distance differences (see Fig. 6.1). It is possible that if we change the scope of sd_1 to $0 \dots 0.1$ and decrease ρ_B enough to produce dense correspondences, different results will emerge. On the other hand, changes in pairwise geodesic distance of features sd_2 (over y-axis) have a strong influence on the outcome. This witnesses the significant contribution of second-order affinities. This significance is also enhanced by the fact that, for different max propagation ρ_B, feature points are located on slightly altered locations on source and target shape and not on "corresponding" placements as expected (corresponding vicinities change due to imperfect isometry preservation). Thus, relations among pairwise geodesic distances are vital for the proposed framework.

In Table 6.1 we demonstrate the optimal combination of matching parameters along with the minimum achieved matching error. Furthermore we provide the number of feature points on shape pairs as well as performance data. Tests on multiple isorings were held having interim conclusion regarding the optimal ρ_B, sd_1 and sd_2. Worth mentioning that, computation costs are almost dedicated to feature extraction rather than matching procedure. Although Hungarian spectral matching is expected to prevail as a combinatorial optimization algorithm, Greedy scheme suits better to dog and centaur experiments.

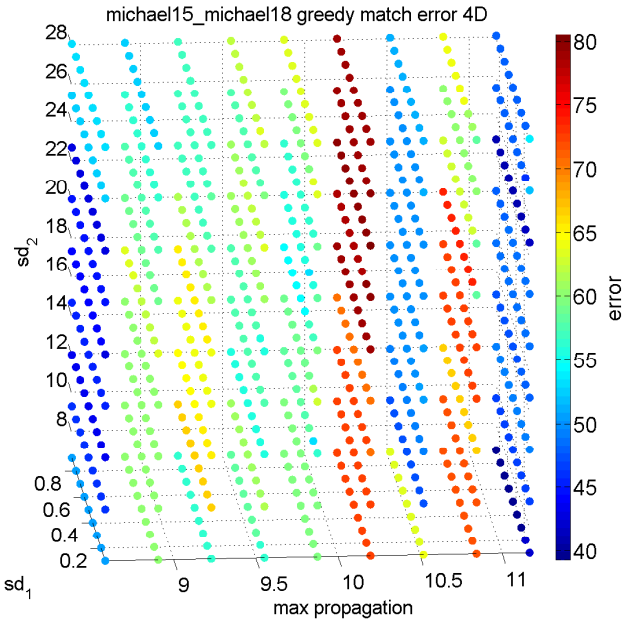
Three-dimensional model illustrations of top matches for the considered 4 shape classes can be found at the end of the chapter (see Fig. 6.9).

model	ρ_B	$\lvert r \rvert$	sd_1	sd_2	$\lvert f_S \rvert$	$\lvert f_T \rvert$	e_H	e_G	t_f	t_m
human	10.5164	9	0.1	25.326	25	23	33.036	50.991	225.31	3.12
cat	10.6839	1	0.1	6.8824	33	29	33.973	34.862	256.15	3.48
dog	9.13885	1	0.8	10.43	35	33	36.962	36.031	175.02	2.25
centaur	17.3755	1	0.1	10.187	23	20	39.644	38.964	515.49	3.55

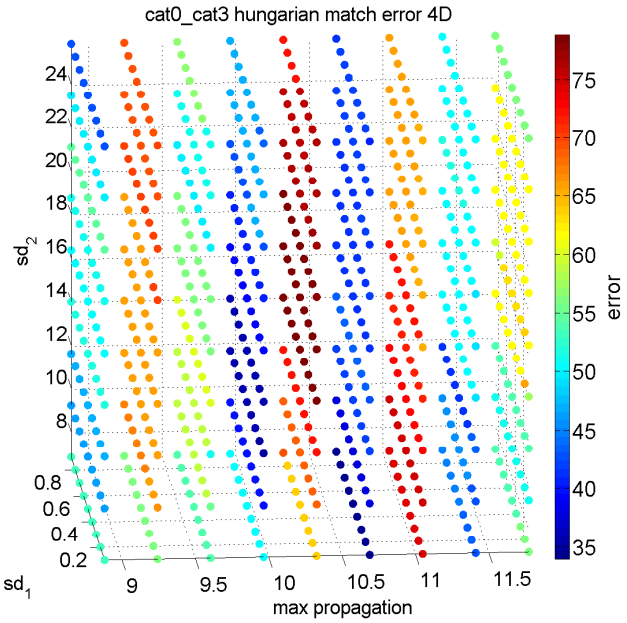
Table 6.1: Summarized optimal matching parameters and statistics for different model pairs. Base radius is indicated by ρ_B and $|f_S|$, $|f_T|$ are the detected feature points for the source and target shape respectively. e_H and e_G denote matching error for Hungarian and Greedy spectral matching scheme and t_f, t_m are execution times (seconds) of features extraction and matching processes in respect.



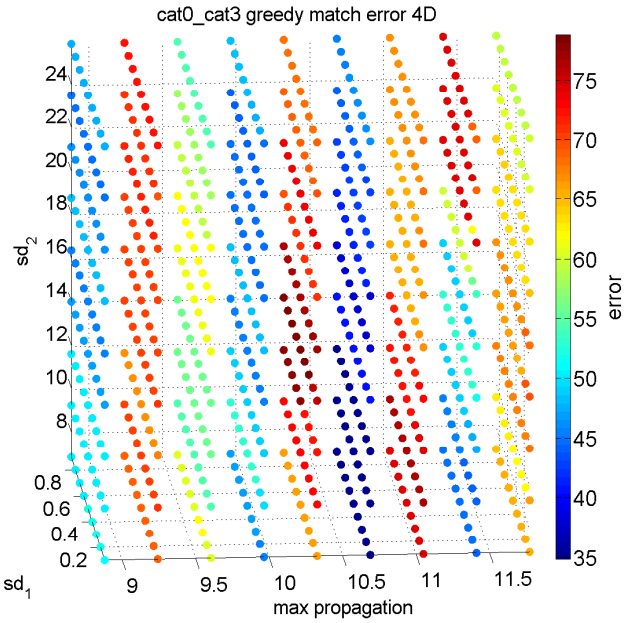
(a) Michael - Hungarian matching



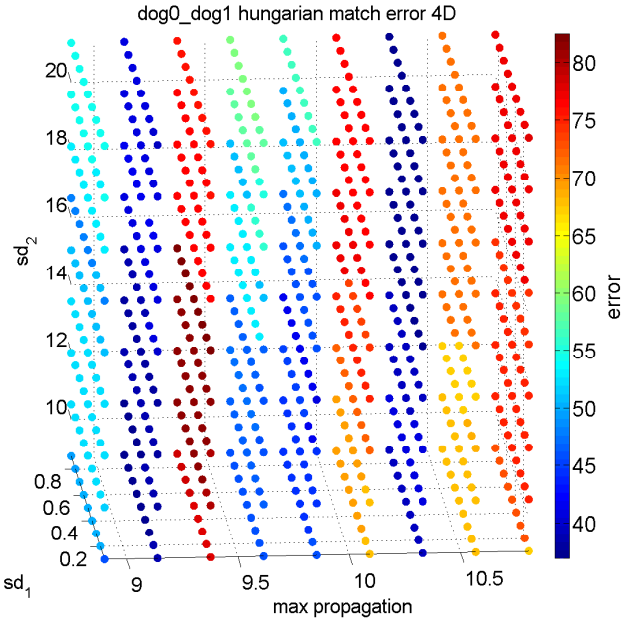
(b) Michael - Greedy matching



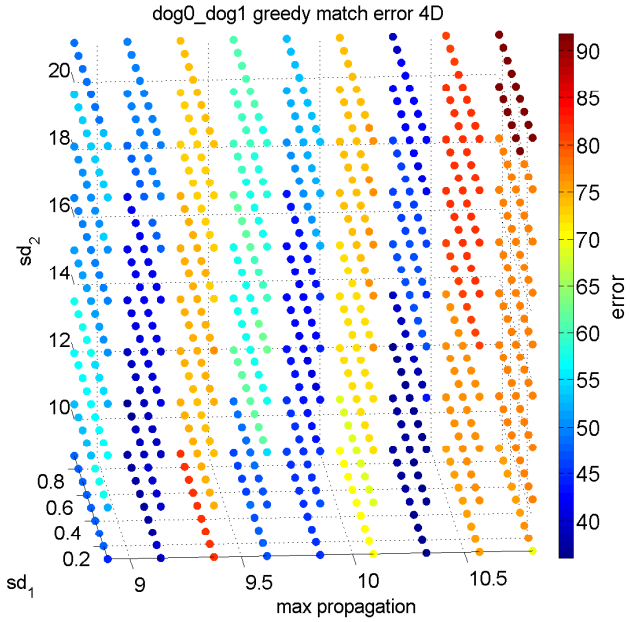
(c) Cat - Hungarian matching



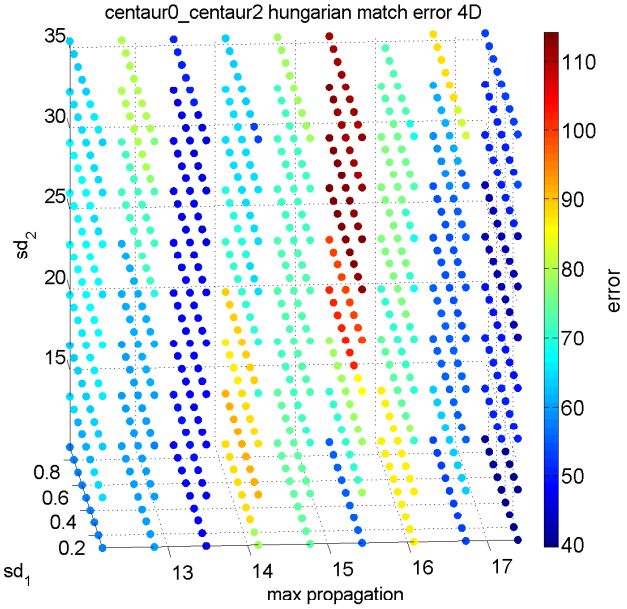
(d) Cat - Greedy matching



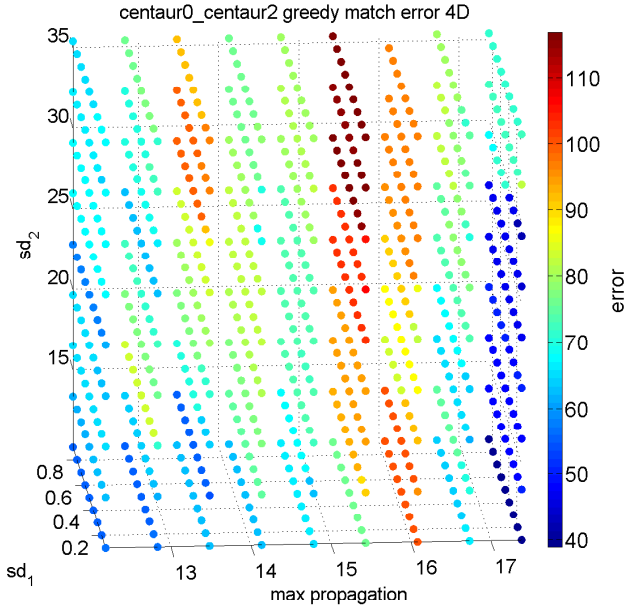
(e) Dog - Hungarian matching



(f) Dog - Greedy matching



(g) Centaur - Hungarian matching



(h) Centaur - Greedy matching

Figure 6.5: Four-dimensional diagram represents the relation between max propagation (x-axis), area descriptor standard deviation comparison (z-axis), geodesic distance standard deviation comparison (y-axis) and matching error (color). Depicted experiments include human (a)(b), cat (c)(d), dog (e)(f) and centaur (g)(h) models over spectral matching (Hungarian/Greedy).

Multi-isoring Vector Descriptor

In the majority of the experiments, vector descriptor exposes equal or worse results comparing to the single-isoring case. Nevertheless, tests on the human model pair proved advantageous for multiple isorings local signature (see Fig. 6.6). This is sufficient to argue that vector descriptor extension enhances the results of the framework. It is observed that the relationship between error decrease and number of inner isorings is not linear or not fully predictable and mostly relies on the triangulation.

Regarding vector descriptor benchmarks, we examined cases ranging from 1 to 9 vector descriptor propagation limits. Table 6.2 presents matching error results for 9 isorings following linear and exponential approach for both Greedy and Hungarian matching internal functions. Besides the fact that exponential case is the best approach, Greedy method presents a noticeable behaviour. To wit, it is observed that error slightly decreases in linear method, something that relies on the fact that the algorithm makes definitive decisions at each stage without reconsidering the result in the future.

model	r^D	$\|r\|$	e_H	e_G
human	*lin.*	9	34.708	49.64
	exp.		33.036	50.991

Table 6.2: Error results for linear and exponential multiple inner isorings distribution. Inner isorings denote the vector descriptor borders. Inner isoring distribution is indicated by r^D and isorings number is $|r|$. e_H and e_G denote matching error for Hungarian and Greedy spectral matching scheme. Although Greedy matching error is smaller in linear distribution than in exponential, exponential still presents the smallest error for both Greedy and Hungarian.

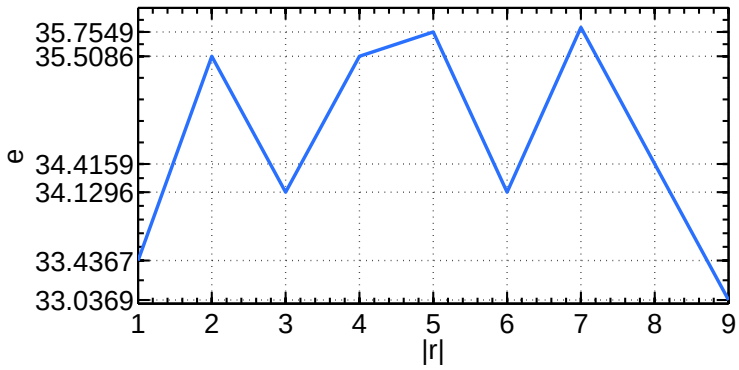


Figure 6.6: Matching error for different number of isorings per descriptor (Hungarian method). It is observed that use of multiple isorings decreases the matching error (from $|r| = 1, e = 33.4367$ to $|r| = 9, e = 33.0369$). This plot refers to the human model.

Exact Geodesic and Fast Marching Functions

Table 6.3 demonstrates computation times and error matching results for exact geodesic [26] and fast marching [28] algorithm. Although [26] presents slightly better results as an exact geodesic computation algorithm (high accuracy is achieved by defining $1e100$ additional vertices on every edge of the mesh), we prefer fast marching because of its lower complexity advantage ($O(n \log^2 n)$ and $O(n \log n)$ in respect). Complexity difference impact is also proved in practice, considering that descriptor scores have to be computed for all vertices of the mesh before detecting the dominant points. We only consider minimum times for feature detection since performance influence on matching phase is comparably insignificant. Worth mentioning that, as geodesic distances slightly vary among the two schemes, resulting feature points are not always identically placed.

In case performance is not of prime priority, [26] suits better for dense triangulations and small base radii (i.e. high number of feature points). In that case, small deviations among descriptor scores that otherwise would be negligible, are crucial during both feature detection and shape matching phase.

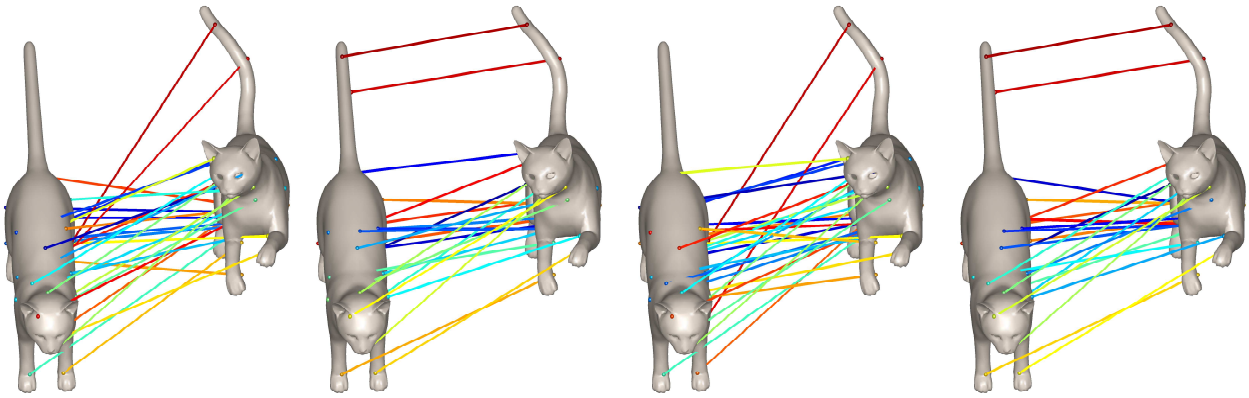
model	method	t_m	e_{min}
human	[Kap99]	360.01	32.991
	[Kim98]	225.31	33.036
cat	[Kap99]	362.11	33.882
	[Kim98]	256.15	33.973
dog	[Kap99]	248.32	35.921
	[Kim98]	175.02	36.031
centaur	[Kap99]	752.24	38.465
	[Kim98]	515.49	38.964

Table 6.3: Results for the use of exact geodesic [26] and fast marching method [28] for geodesic distances computation. t_m is execution time (seconds) of features extraction and e_{min} is the smallest error between e_H and e_G. Matching error difference is relatively small comparing to the increased computation costs of [26].

Single-Order versus Double-Order Affinities

Figure 6.7 illustrates matching results involving the use of first order and combined first and second-order affinities. Employment of Greedy scheme is disadvantageous for both Ω_1 and $\Omega_1\Omega_2$ cases. Although feature points (local maxima) are detected on corresponding locations (something that proves that the proposed descriptor is isometry invariant), deviations among feature scores do not allow for an exclusive use of first-order affinities Ω_1. The outcome also aims to indicate the vital contribution of second-order affinities in the framework. Illustration

results do not witness an acceptable matching for explicit Ω_1 application although they are distant from the worst case of $e_{max} = 206.95$ (see Tab. 6.4).



(a) $\Omega_1, e_H = 129.972$ (b) $\Omega_1\Omega_2, e_H = 33.973$ (c) $\Omega_1, e_G = 138.281$ (d) $\Omega_1\Omega_2, e_G = 34.862$

Figure 6.7: Example of cat model matching using first-order affinities (a)(c) and both first and second-order affinities (b)(d) for Hungarian and Greedy spectral matching. Large matching deviations are observed at the front paws and tail of the cat.

Internal Spectral Matching Functions

We introduce two diagram groups referring to the relations of the two internal matching functions (see Fig. 6.8). The first group depicts the normalized matching error to the percentage of top matches. The second one illustrates Hungarian-Greedy error difference to the respective number of correspondences. Regarding the first class of plots, matching error is normalized by dividing the correspondence vector error (vector including the error of each correspondence) with its $2-norm$ before sorting and summing up error in ascending order. To wit, $corr\_vector\_error\_normalized = corr\_vector\_error/norm(corr\_vector\_error)$, where $norm$ is the $2-norm$ of a vector.

It is observed that, depending on the percentage of top correspondences, different conclusions are made on each internal matching scheme. In general, deviation between Hungarian and Greedy grows larger (up to a point) as the percentage of top correspondences increases. This is because Greedy method makes definite decisions at each step in contrast to Hungarian.

Generally, gradient in the first group of diagrams (see Fig. 6.8 (a)(c)(e)(g)) is explained as "the quality of current correspondence" since it witnesses the contribution of the current match to the total error. As top correspondences reach their full percentage, a steep error increase in noticed. This is a consequence of the remaining less interesting feature points that the algorithm is "forced" to match. Hence, this point is interpreted as the start of insignificant correspondences and denotes an automatic fraction of best coarse correspondences to keep.

State-of-the-art Shape Matching Methods Comparison

The following table (see Tab. 6.4) demonstrates comparison statistics between the proposed mesh matching framework and two state-of-the-art methods, that is, Blended Intrinsic Maps [27] and Möbius Voting [30].

Concerning basic parametrization in Blended Intrinsic Maps, we use 256 samples, a parameter that approximates confidence and similarity of matches. Furthermore, for Möbius Voting we consider 100 samples and 10^5 votes (votes are used in the prediction of successful matches). For details on the parameters please refer to [27] and [30] in respect.

In the case of [27], we compute dense (d) as well as sparse correspondences *(s)*. As regards to sparse correspondences, in order to keep the comparison unbiased, we retain only the vertices (from the dense result) that are respectively denoted as feature points in the area descriptor framework (on the source shape) and then compute the error. Hence, sparse correspondences of [27] are equal to the number of correspondences of the proposed framework. The method in [30] presents its correspondences explicitly, thus we do not interfere on the selection of featured points.

Error calculation for both the existing methods as well as ours, is held as explained in Section 5.4 (each of the tests considers the respective number of correspondences).

Although our algorithm presents the highest matching error, there are two points worth noting: First, we consider smaller number of candidate correspondences (mainly due to hardware limitations), something that is frequently more "punitive" regarding the matching outcome. Second, taking into consideration the error extrema $0 \dots e_{max}$, our framework presents error that falls into $13 \dots 21.3\%$ of the lowest part of the error range. By comparing it to the corresponding $7.7 \dots 13.5\%$ of [27] and [30], the general outcome becomes very promising.

model	*|c|(B(d,s),M,A)*				*BlendedM(d,s)* [27]		*Möbius* [30]	*AreaDesc*	e_{max}
human	6615	23	58	23	22.363	17.121	28.727	33.036	224.07
cat	7002	29	58	29	24.836	28.44	29.026	33.973	206.95
dog	6342	33	58	33	24.491	28.024	22.776	36.031	169.064
centaur	7885	20	98	20	21.003	21.639	27.073	38.964	282.599

Table 6.4: Comparison with existing shape matching methods. *|c|(B(d,s),M,A)* denotes the number of established correspondences for Blended Intrinsic Maps (dense, sparse) [27], Möbius Voting [30] and the proposed area descriptor. Number after double horizontal lines indicate error values. e_{max} denotes the maximum possible error which is equal to the longest path on the target shape.

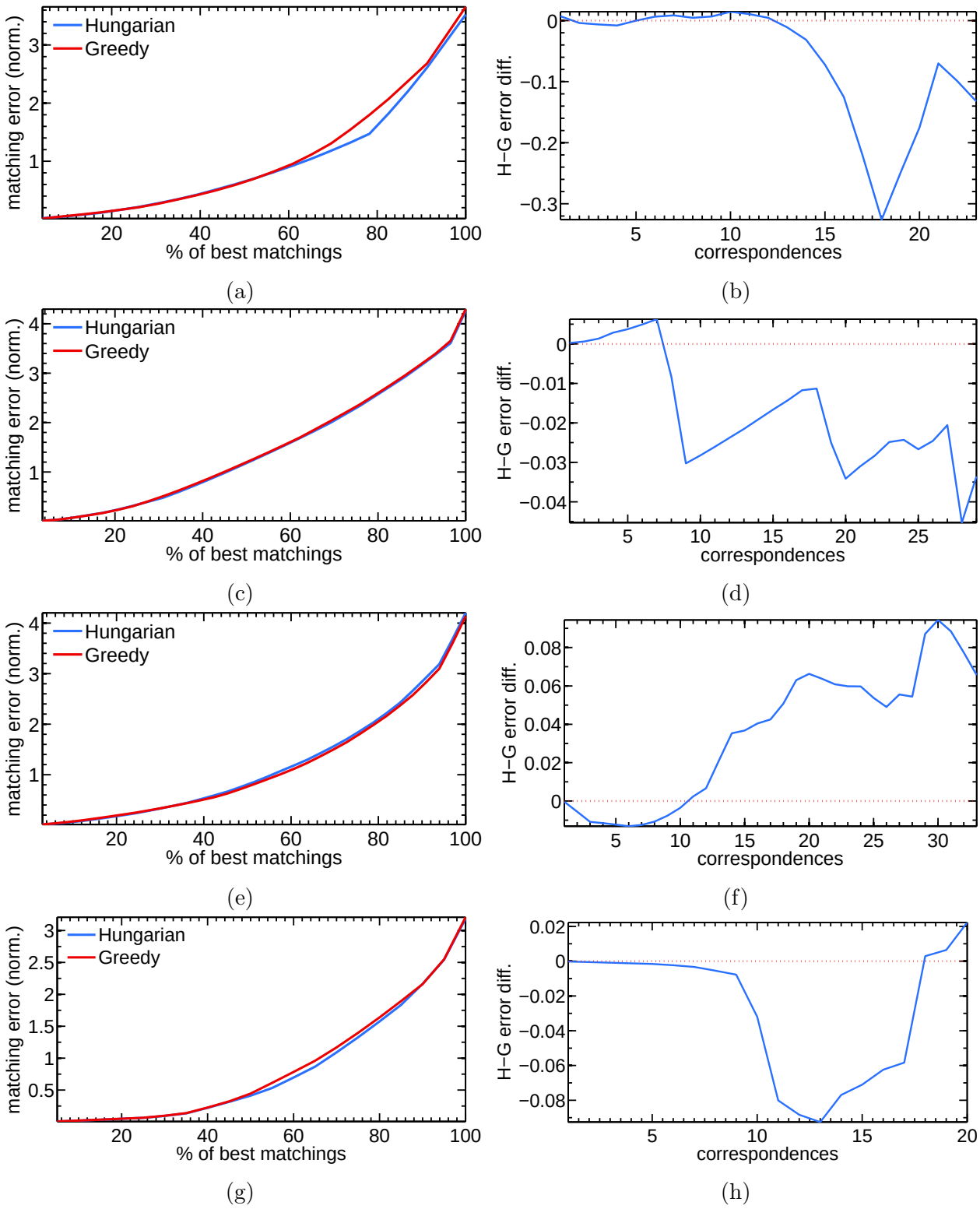
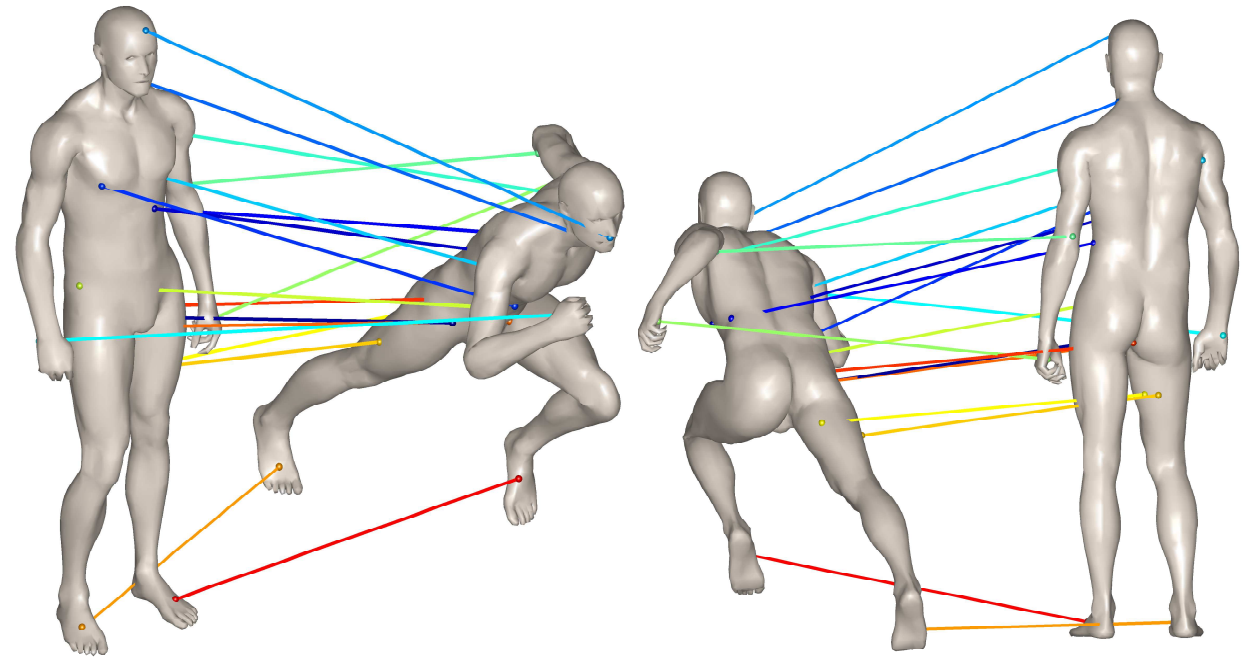
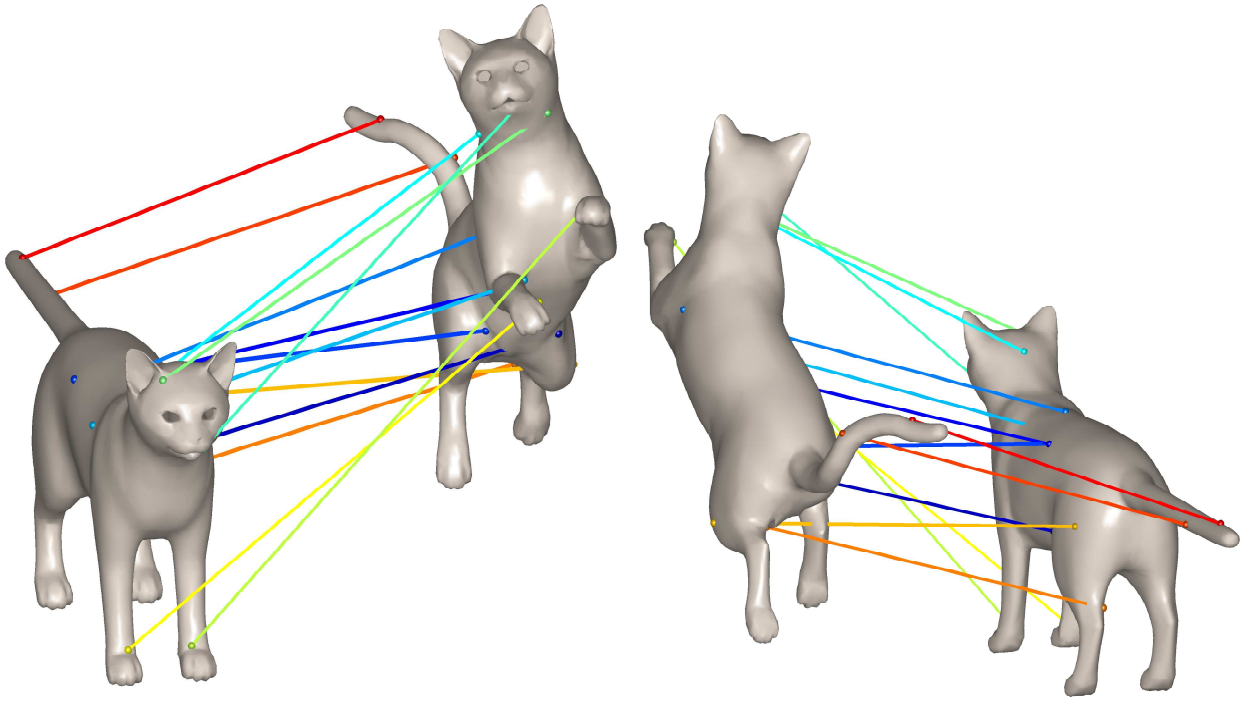


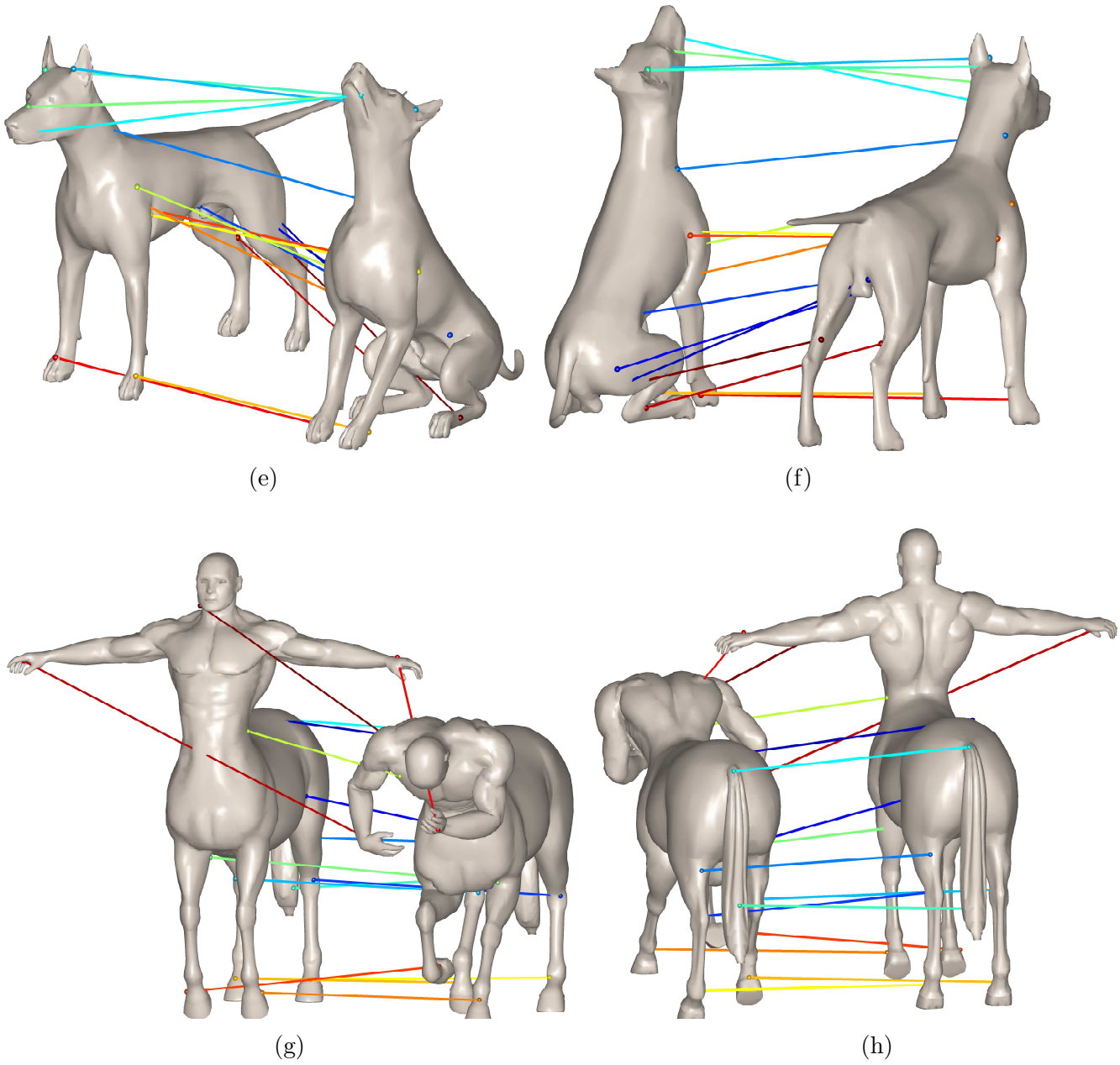
Figure 6.8: Diagram pairs including normalized matching error to percentage of top matchings and Hungarian-Greedy spectral matching error difference to number of correspondences respectively. Plots refer to human (a)(b), cat (c)(d), dog (e)(f) and centaur (g)(h) models.



(a) (b)



(c) (d)



(e) (f)

(g) (h)

Figure 6.9: Best matching results for human (a)(b), cat (c)(d), dog (e)(f) and centaur (g)(h) models. Experiments also considered different numbers of inner isorings. Feature points are rendered as spheres. Points that are in correspondence have same color and are connected by a line.

Chapter 7

Epilogue

This chapter provides a summary of the problem and points out the main findings. Next, limitations are denoted along with proposals of directions that further research should go.

7.1 Conclusions

In this thesis, we introduce a framework for isometry invariant shape matching. We primarily focus on detecting isometry maps between shapes aiming to derive a robust and efficient method that can be applied in practice.

Initially, we compute a set of characteristic feature points on both shapes based on the area descriptor. Next, we seek the optimal combination of correspondences such that feature scores and the pairwise geodesic distances among all equivalent pairs of feature points are retained. This procedure is assigned to a graph matching scheme called spectral matching. We includ benchmarks exclusively dedicated to the area descriptor followed by tests on the matching framework.

One of the main contributions of this thesis was the proposal of the vicinity area descriptor (inspired by [54] and [64]) and the fact that is an isometry invariant local signature. Its adaptable referring vicinity, exposes various surface properties making it a multi-scale and a multi-pupose descriptor. In addition, we showed that, although easy to conceive, our method presents results that decently stand against state-of-the-art shape matching algorithms.

7.2 Limitations

Regarding limitations, our framework takes for granted geometry of known, consistent topology. As showed, we used synthetic meshes for our experiments assuming the absence of measurement holes. Hence, there are issues to resolve before applying the method to real-world scanner data (this also refers to future work). Moreover, area descriptor is "partially" scale-invariant. To wit, although we include a descriptor score normalization step (division of score with the respective flat disc area), scale factor of one of the shapes has to be known.

Hence, we can adapt the magnitude of the geodesic base radius accordingly. Furthermore, matching process is hardware-depended. Memory requirements as well as running times rise rapidly in the case of dense correspondences establishment. In addition, concerning large base radii or simple triangulations, feature points are not only sparse but likely to occur in inconsistent locations. This preconceives the process to an improper matching. Lastly, although the descriptor is isometry invariant, deviations on future scores imply that, exclusive use of first-order affinities is not capable of carrying out acceptable results.

7.3 Future Work

Various future extensions exist referring to several stages of the proposed framework. They concern both the quality of the result as well as performance enhancements.

In relation to geodesic computation, the algorithm of Martínez *et al.* [34] comprises an interesting extension of the fast marching method. That is, it improves geodesic approximation through an iterative process and runs in similar speed.

In our framework, standard deviation in the case of vector descriptor is indicated by a scalar. This number is chosen based on the score of the base radius. Results would be improved in case we applied a "tailored" tolerance regarding each isoring score, that is, multiple standard deviations.

An extension worth testing is to replace -the commonly used geodesic distance- second order affinity. To wit, a pairwise vector descriptor based on 2-dimensional *LaplaceBeltrami operator* [25] on each point of the curve would increase the saliency of candidate correspondences.

Efficient Non-Maximum Suppression [43] can be used during feature detection. In contrast to the regular NMS, it remains unaffected from the neighborhood size and drastically minimizes the number of internal comparisons.

We are optimistic that, future tests on improved hardware would reduce matching error. To wit, it would be possible to handle larger numbers of candidate correspondences and operate on denser triangulations.

Furthermore, a challenging enhancement is to establish a method that automatically indicates the optimal base radius for the matching. Hence, tests on a range of maximum propagations is avoided. Moreover, by detecting a feature set emerging from a diversity of local maxima (base radii), would capture more interesting aspects of the surface. Another challenging problem for future work is to provide a theoretical justification concerning the stability of the framework under near-isometries. Is there any threshold for the used standard deviations? Which is the isometry error threshold that area descriptor fails to provide a decent map? How about matching shapes in different classes? We think it is also interesting to study the proposed framework in the context of self-similarities. Thus, the algorithm applications would broaden to symmetry detection and semantic shape segmentation.

Chapter 8

Implementation

In the following subsections, we provide several noteworthy implementation approaches. The proposed shape matching framework spreads over $\sim 3,700$ lines of C++11 and MATLAB code (excluding existing functions/methods). For execution instructions, please refer to *Usage.txt.*

8.1 Storing Distance Map

We integrate MATLAB API in C++ in order to handle *.mat* files (extension containing MATLAB code). The first reason is that, reading .mat files through MATLAB API takes milliseconds instead of reading text-based files in C++. Secondly, the developed framework shares code between MATLAB and C++. In addition, *vector::assign* instruction assigns new contents to the vector, replacing its current contents being incomparably faster than the traditional *vector::push_back* approach. This integration allows us to execute the code as a single program and without having to run separate codes manually.

```
...
#include "mat.h"

void DistanceMapStorage::DistanceMapMatRead(const char* file, vector<double>& v){
        // open .mat file
        MATFile *pmat = matOpen(file, "r");
        if (!pmat) return;
        // extract the specified variable
        mxArray *arr = matGetVariable(pmat, "DMAP");
        if (arr && mxIsDouble(arr) && !mxIsEmpty(arr)){
                // copy data
                mwSize num = mxGetNumberOfElements(arr);
                double* pr = mxGetPr(arr);
                if(pr){
                        v.resize(num);
                        v.assign(pr, pr + num);
                }
        }
```

```
        // cleanup
        mxDestroyArray(arr);
        matClose(pmat);
}
```

Listing 8.1: DistanceMapMatRead

8.2 Detect Vertices Inside the Vicinity

The following function describes the computation of a vicinity vertex multimap. To wit, given the outer isoring radius, it maps every vertex on the mesh (considered as *source vertex*) to a list including the vertices in its vicinity (*target vertices*). The notable idea is that, instead of linearly linking every vertex to its surrounding points, we read only the upper triangular part of the distance map while exchanging the key and values on map-element insertion. Thus, the algorithm recursions become $\frac{N^2-N}{2}$ comparing to N^2 of the naive method (where N the total vertices of the mesh). This is due to the fact that: "vertex t lies inside the vicinity of vertex s" is equivalent to "vertex s lies inside the vicinity of vertex t".

```
...
#include <map>
...
void AreaDescriptor::StoreVerticesIdsInVicinityForEveryVertexFastMarchingMatReadMethod(){
        unsigned j, k = 1;
        for (unsigned i = 0; i < (total_vertices - 1); ++i){
                j = k;
                while (j < total_vertices ){
                    if ( isless (mesh3Dptr->GetDistanceSourceToTargetDistanceMapVectorMethod(i, j),
                     propagation_distacne)){
                                vicinity_vertices_multimap .emplace(i, j );
                                vicinity_vertices_multimap .emplace(j, i );
                        }
                        j++;
                }
                k++;
        }
};
```

Listing 8.2: StoreVerticesIdsInVicinityForEveryVertexFastMarchingMatReadMethod

8.3 Detecting Faces in Vicinity

This code excerpt presents the process of detecting the faces inside the vicinity of a vertex. The key concept is the use of a map (faceCount) that maps each face to its occurrences inside the vicinity. Thus, when a value of the map reaches 3, the corresponding face is pushed in the vicinity faces list. A similar process is used for the search of faces inside inner isorings.

```
...
#include <map>
...
//declarations
// multimap including vicinity vertices [value] to each vertex [key]
multimap<unsigned, unsigned> vicinity_vertices_multimap;
// multimap includes vertices [key] to adjacent faces [value]
multimap<unsigned, unsigned> vertex_to_faces_multimap;
//map from to face [key] to number of occurences in vicinity [value]
map<unsigned, unsigned> faceCount;
//vector of faces inside the vicinity of the current central vertex
vector<int> facesList_vicinity;
...
void AreaDescriptor::FacesInVicinity(int central_vertex){
        for (auto vvm_it = vicinity_vertices_multimap.lower_bound(central_vertex); vvm_it !=
          vicinity_vertices_multimap.upper_bound(central_vertex); ++vvm_it)
                TraceVertexToFacesMultimap(vvm_it->second);

        //don't forget to also consider the central vertex
        TraceVertexToFacesMultimap(central_vertex);
}

void AreaDescriptor::TraceVertexToFacesMultimap(int multimap_key){
        pair <std::multimap<int, int>::iterator, std::multimap<int, int>::iterator> ret;
        ret = vertex_to_faces_multimap.equal_range(multimap_key);

        for (auto it = ret.first; it != ret.second; ++it){
                //FaceCount, counter of occurences. Look if it's already there.
                if (faceCount.find(it->second) == faceCount.end())
                // Then we've encountered the word for a first time.
                        faceCount[it->second] = 1; // Initialize it to 1.
                else{ // Then we've already seen it before..
                        faceCount[it->second]++; // Just increment it.
                        if (faceCount[it->second] == 3)//Entire triangle inside vicinity!
                                facesList_vicinity.push_back(it->second);
                }
        }
}
```

Listing 8.3: FacesInVicinity

8.4 Integrating MATLAB API to C++ Matching Framework

Executing MATLAB instructions in C++ environment is crucial for the accuracy of calculations and execution time (see Sec. 8.1) of the framework. The following code quotation

presents our approach on calling custom MATLAB functions and passing variables in MATLAB environment.

```
...
#include "engine.h"
...
Engine* ep;
...
MatlabIOHandler::MatlabIOHandler(){
    if (!(ep = engOpen(NULL))) cout << "Can't start MATLAB engine";
}

MatlabIOHandler::~MatlabIOHandler(){
    engClose(ep);
}

void MatlabIOHandler::ComputeMathcingErrorMatlabIO(Engine* ep){
        //always set cuurent workspace the folder that Matlab project function exist
        engEvalString(ep, "cd D:/../Thesis/ final_project /Matlab/");
        engEvalString(ep, "clear;"); //clear workspace

        //pass string variables
        string temp_str = mesh_name_no_context;
        mxArray *mx_mesh_name_no_context;
        if (!(mx_mesh_name_no_context = mxCreateString(temp_str.c_str())))
                std::cout << "Unable to convert to mxArray\n";
        if (engPutVariable(ep, "mesh2_name", mx_mesh_name_no_context))
                std::cout << "Unable to put into engine workspace\n";
        ...
        //also pass scalars
        mxArray *mx_max_propagation;
        if (!(mx_max_propagation = mxCreateDoubleMatrix(1, 1, mxREAL)))
                std::cout << "Unable to convert to mxArray\n";
        memcpy((void *)mxGetPr(mx_max_propagation), static_cast<void*>(&max_propagation),
         sizeof(double));
        if (engPutVariable(ep, "max_propagation", mx_max_propagation))
                std::cout << "Unable to put into engine workspace\n";

        //execute custom matlab function
        engEvalString(ep, "main_matlab_handler(mesh2_name,...,max_propagation,...);");
        mxDestroyArray(mx_mesh_name_no_context);
        ...
        mxDestroyArray(mx_max_propagation);
        ...
}
```

Listing 8.4: ComputeMathcingErrorMatlabIO

Bibliography

[1] Aleksandr Danilovich Aleksandrov and Viktor A Zalgaller. *Intrinsic geometry of surfaces*. Vol. 15. American Mathematical Society Providence, RI, 1967.

[2] Dragomir Anguelov et al. "The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces". In: *Advances in neural information processing systems* 17 (2005), pp. 33–40.

[3] Alexander M. Bronstein et al. "Partial Similarity of Shapes Using a Statistical Significance Measure". In: *IPSJ Transactions on Computer Vision and Applications* 1 (2009), pp. 105–114. DOI: 10.2197/ipsjtcva.1.105.

[4] Alexander Bronstein, Michael Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. 1st ed. Springer Publishing Company, Incorporated, 2008. ISBN: 0387733000, 9780387733005.

[5] M. Brown, R. Szeliski, and S. Winder. *Multi-Image Matching Using Multi-Scale Oriented Patches*. Tech. rep. MSR-TR-2004-133. Dec. 2004.

[6] John Canny and John Reif. "New lower bound techniques for robot motion planning problems". In: *Foundations of Computer Science, 1987., 28th Annual Symposium on*. IEEE. 1987, pp. 49–60.

[7] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976. ISBN: 9780132125895.

[8] Jindong Chen and Yijie Han. "Shortest Paths on a Polyhedron". In: *Proceedings of the Sixth Annual Symposium on Computational Geometry*. SCG '90. Berkley, California, USA: ACM, 1990, pp. 360–369. ISBN: 0-89791-362-0. DOI: 10.1145/98524.98601. URL: http://doi.acm.org/10.1145/98524.98601.

[9] O. Chum and J. Matas. "Matching with PROSAC - progressive sample consensus". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. June 2005, 220–226 vol. 1. DOI: 10.1109/CVPR.2005.221.

[10] Laurent D Cohen and Thomas Deschamps. "Grouping connected components using minimal path techniques. Application to reconstruction of vessels in 2D and 3D images". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2001, pp. II–102.

[11] Laurent D. Cohen and Ron Kimmel. "Global Minimum for Active Contour Models: A Minimal Path Approach". In: *Int. J. Comput. Vision* 24.1 (Aug. 1997), pp. 57–78. ISSN: 0920-5691. DOI: 10.1023/A:1007922224810. URL: http://dx.doi.org/10.1023/A:1007922224810.

[12] L.D. Cohen. "Multiple contour finding and perceptual grouping using minimal paths". In: *Variational and Level Set Methods in Computer Vision, 2001. Proceedings. IEEE Workshop on*. 2001, pp. 89–96. DOI: 10.1109/VLSM.2001.938886.

[13] *Computational Learning and Visual Perception Research Group*. http://109.101.234.42/code.php. Accessed: 2014-05-11.

[14] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh Vazirani. *Algorithms*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2008. ISBN: 0073523402, 9780073523408.

[15] Hongli Deng et al. "Reinforcement Matching Using Region Context". In: *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*. June 2006, pp. 11–11. DOI: 10.1109/CVPRW.2006.169.

[16] Thomas Deschamps and Laurent D. Cohen. "Fast extraction of minimal paths in 3D images and applications to virtual endoscopy". In: *Medical Image Analysis* 5 (4 2001), pp. 281–299. DOI: 10.1016/S1361-8415(01)00046-9.

[17] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[18] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: http://doi.acm.org/10.1145/358669.358692.

[19] Timothy Gatzke et al. "Curvature maps for local shape comparison". In: *Shape Modeling and Applications, 2005 International Conference*. IEEE. 2005, pp. 244–253.

[20] Allen Van Gelder. "Efficient Computation of Polygon Area and Polyhedron Volume". In: *Graphics Gems V*. Ed. by Alan W. Paeth. Academic Press, 1995, pp. 35–41.

[21] Natasha Gelfand et al. "Robust global registration". In: *Symposium on geometry processing*. Vol. 2. 3. 2005, p. 5.

[22] Roger A. Horn and Charles R. Johnson, eds. *Matrix Analysis*. New York, NY, USA: Cambridge University Press, 1986, pp. 176–180. ISBN: 0-521-30586-1.

[23] Qi-Xing Huang et al. "Non-Rigid Registration Under Isometric Deformations". In: *Computer Graphics Forum*. Vol. 27. 5. Wiley Online Library. 2008, pp. 1449–1457.

[24] *Hungarian Algorithm*. http://www.mathworks.com/matlabcentral/fileexchange/11609-hungarian-algorithm. Accessed: 2014-05-20.

[25] Jürgen Jost. *Riemannian geometry and geometric analysis*. Springer Science & Business Media, 2008.

[26] Sanjiv Kapoor. "Efficient Computation of Geodesic Shortest Paths". In: *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*. STOC '99. Atlanta, Georgia, USA: ACM, 1999, pp. 770–779. ISBN: 1-58113-067-8. DOI: 10.1145/301250.301449. URL: http://doi.acm.org/10.1145/301250.301449.

[27] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. "Blended intrinsic maps". In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 4. ACM. 2011, p. 79.

[28] Ron Kimmel and James A Sethian. "Computing geodesic paths on manifolds". In: *Proceedings of the National Academy of Sciences* 95.15 (1998), pp. 8431–8435.

[29] Marius Leordeanu and Martial Hebert. "A spectral technique for correspondence problems using pairwise constraints". In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 2. IEEE. 2005, pp. 1482–1489.

[30] Yaron Lipman and Thomas Funkhouser. "Möbius voting for surface correspondence". In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 72.

[31] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[32] Alan P. Mangan and Ross T. Whitaker. "Partitioning 3D Surface Meshes Using Watershed Segmentation". In: *IEEE Transactions on Visualization and Computer Graphics* 5.4 (Oct. 1999), pp. 308–321. ISSN: 1077-2626. DOI: 10.1109/2945.817348. URL: http://dx.doi.org/10.1109/2945.817348.

[33] Alan P. Mangan and Ross T. Whitaker. "Surface Segmentation Using Morphological Watersheds". In: *IEEE Visualization '98: Late Breaking Topics*, p. 2932.

[34] Dimas Martínez, Luiz Velho, and Paulo C Carvalho. "Computing geodesics on triangular meshes". In: *Computers & Graphics* 29.5 (2005), pp. 667–675.

[35] Alan M. McIvor, David W. Penman, and Peter T. Waltenberg. *Simple Surface Segmentation*. 1997.

[36] Donald Allan McQuarrie. *Statistical Mechanics*. Harper's Chemistry Series. New York: HarperCollins Publishing, Inc., 1976.

[37] Krystian Mikolajczyk and Cordelia Schmid. "Scale and affine invariant interest point detectors". In: *International Journal of Computer Vision* 60.1 (2004), pp. 63–86. URL: http://lear.inrialpes.fr/pubs/2004/MS04.

[38] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. "The Discrete Geodesic Problem". In: *SIAM J. Comput.* 16.4 (Aug. 1987), pp. 647–668. ISSN: 0097-5397. DOI: 10.1137/0216045. URL: http://dx.doi.org/10.1137/0216045.

[39] Niloy J Mitra et al. "Registration of point cloud data from a geometric optimization perspective". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM. 2004, pp. 22–31.

[40] G Mori, S Belongie, and J Malik. "Shape contexts enable efficient retrieval of similar shapes". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conf. on*. Vol. 1. 2001, 723–730 vol.1. DOI: 10.1109/CVPR.2001.990547.

[41] J. Munkres. "Algorithms for the Assignment and Transportation Problems". In: *Journal of the Society of Industrial and Applied Mathematics* 5.1 (Mar. 1957), pp. 32–38.

[42] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988. ISBN: 0-471-82819-X.

[43] Alexander Neubeck and Luc Van Gool. "Efficient non-maximum suppression". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 3. IEEE. 2006, pp. 850–855.

[44] Maks Ovsjanikov et al. "One Point Isometric Matching with the Heat Kernel". In: *Comput. Graph. Forum* 29.5 (July 2010), pp. 1555–1564.

[45] Birgit M Planitz, Anthony J Maeder, and John A Williams. "Intrinsic correspondence using statistical signature-based matching for 3D surfaces". In: *In: Australian Pattern Recognition Society (APRS) Workshop on Digital Image Computing (WDIC*. 2003.

[46] Konrad Polthier and Markus Schmies. "Straightest Geodesics on Polyhedral Surfaces". English. In: *Mathematical Visualization*. Ed. by Hans-Christian Hege and Konrad Polthier. Springer Berlin Heidelberg, 1998, pp. 135–150. ISBN: 978-3-642-08373-0. DOI: 10.1007/978-3-662-03567-2_11.

[47] Sandeep Pulla, Anshuman Razdan, and Gerald Farin Z. "Improved curvature estimation for watershed segmentation of 3-dimensional meshes. manuscript". In: *IEEE Trans. Visualization and Computer Graphics* 2002 (2001).

[48] Mauro R Ruggeri and Dietmar Saupe. "Isometry-invariant Matching of Point Set Surfaces." In: *3DOR*. Citeseer. 2008, pp. 17–24.

[49] C. Schmid, R. Mohr, and C. Bauckhage. "Comparing and evaluating interest points". In: *Computer Vision, 1998. Sixth International Conference on*. Jan. 1998, pp. 230–235. DOI: 10.1109/ICCV.1998.710723.

[50] James A Sethian. *A fast marching level set method for monotonically advancing fronts*. Vol. 93. 4. National Academy of Sciences, 1996, pp. 1591–1595.

[51] Heung-Yeung Shum, Martial Hebert, and Katsushi Ikeuchi. "On 3D shape similarity". In: *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*. IEEE. 1996, pp. 526–531.

[52] Dro Desire Sidibe, Philippe Montesinos, Stefan Janaqi, et al. "Fast and robust image matching using contextual information and relaxation". In: *VISAPP 07-2nd International Conference on Computer Vision Theory and Applications* (2007), pp. 68–75.

[53] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. "A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion". In: *Eurographics Symposium on Geometry Processing (SGP)*. 2009.

[54] Art Tevs et al. "Intrinsic shape matching by planned landmark sampling". In: *Computer Graphics Forum* 30.2 (2011), pp. 543–552.

[55] Art Tevs et al. "Isometric registration of ambiguous and partial data". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 1185–1192.

[56] *Toolbox Fast Marching*. `http://www.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching`. Accessed: 2014-10-01.

[57] *TOSCA high-resolution dataset*. `http://tosca.cs.technion.ac.il/book/resources_data.html`. Accessed: 2014-04-30.

[58] J.N. Tsitsiklis. "Efficient algorithms for globally optimal trajectories". In: *Automatic Control, IEEE Transactions on* 40.9 (Sept. 1995), pp. 1528–1538. ISSN: 0018-9286. DOI: `10.1109/9.412624`.

[59] T. Tung and T. Matsuyama. "Dynamic surface matching by geodesic mapping for 3D animation transfer". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. June 2010, pp. 1402–1409. DOI: `10.1109/CVPR.2010.5539806`.

[60] Tinne Tuytelaars and Luc Van Gool. "Matching widely separated views based on affine invariant regions". In: *International journal of computer vision* 59.1 (2004), pp. 61–85.

[61] VCG, *Visualization and Computer Graphics Library*. `http://vcg.isti.cnr.it/vcglib/index.html`. Accessed: 2014-04-30.

[62] Fabien Vivodtzev et al. "Hierarchical Isosurface Segmentation Based on Discrete Curvature". In: *Proceedings of the Symposium on Data Visualisation 2003*. VISSYM '03. Grenoble, France: Eurographics Association, 2003, pp. 249–258. ISBN: 1-58113-698-6. URL: `http://dl.acm.org/citation.cfm?id=769922.769950`.

[63] Richard C. Wilson and Edwin R. Hancock. "Consistent topographic surface labelling". In: *Pattern Recognition* 32.7 (1999), pp. 1211–1223. ISSN: 0031-3203. DOI: `http://dx.doi.org/10.1016/S0031-3203(98)00146-0`. URL: `http://www.sciencedirect.com/science/article/pii/S0031320398001460`.

[64] Steve Zelinka and Michael Garland. "Similarity-based surface modelling using geodesic fans". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM. 2004, pp. 204–213.

[65] Hao Zhang et al. "Deformation-Driven Shape Correspondence". In: *Computer Graphics Forum (Special Issue of Symposium on Geometry Processing 2008)* 27.5 (2008), pp. 1431–1439.